triSYCL implementation of SYCL

Generated by Doxygen 1.9.4

1 triSYCL implementation of SYCL 1

2 Todo List 3

3 Module Index 17

3.1 Modules . . . 17

4 Namespace Index 19

4.1 Namespace List . . . 19

5 Concept Index 21

5.1 Concepts . . . 21

6 Hierarchical Index 23

6.1 Class Hierarchy . . . 23

7 Class Index 33

7.1 Class List . . . 33

8 File Index 37

8.1 File List . . . 37

9 Module Documentation 43

9.1 AI Engine CGRA . . . 43

9.1.1 Detailed Description . . . 44

9.1.2 Class Documentation . . . 45

9.1.2.1 class trisycl::vendor::xilinx::acap::aie::axi_stream_switch . . . 45

9.1.2.2 struct trisycl::vendor::xilinx::acap::aie::cascade_stream . . . 68

9.1.2.3 class trisycl::vendor::xilinx::acap::aie::axi_packet . . . 72

9.1.2.4 class trisycl::vendor::xilinx::acap::aie::dma_dsl . . . 75

9.1.2.5 struct trisycl::vendor::xilinx::acap::aie::geography . . . 81

9.1.2.6 struct trisycl::vendor::xilinx::acap::aie::layout::full . . . 116

9.1.2.7 struct trisycl::vendor::xilinx::acap::aie::layout::one_pe . . . 119

9.1.2.8 struct trisycl::vendor::xilinx::acap::aie::layout::size . . . 120

9.1.2.9 struct trisycl::vendor::xilinx::acap::aie::layout::small . . . 122

9.1.2.10 struct trisycl::vendor::xilinx::acap::aie::layout::ve1152 . . . 123

9.1.2.11 struct trisycl::vendor::xilinx::acap::aie::lock_unit . . . 124

9.1.2.12 struct trisycl::vendor::xilinx::acap::aie::memory . . . 127

9.1.2.13 struct trisycl::vendor::xilinx::acap::aie::memory_base . . . 129

9.1.2.14 class trisycl::vendor::xilinx::acap::aie::memory_infrastructure . . . 131

9.1.2.15 struct trisycl::vendor::xilinx::acap::aie::program . . . 134

9.1.2.16 struct trisycl::vendor::xilinx::acap::aie::queue . . . 153

9.1.2.17 class trisycl::vendor::xilinx::acap::aie::shim_tile . . . 161

9.1.2.18 struct trisycl::vendor::xilinx::acap::aie::tile . . . 180

9.1.2.19 class trisycl::vendor::xilinx::acap::aie::tile_base . . . 232

9.1.2.20 class trisycl::vendor::xilinx::acap::aie::tile_infrastructure . . . 243
9.1.2.21 class trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure . . . 265
9.2 Extensions for Xilinx ACAP Versal . . . 296
9.2.1 Detailed Description . . . 296
9.3 Debugging support for ACAP . . . 297
9.3.1 Detailed Description . . . 297
9.3.2 Class Documentation . . . 298
9.3.2.1 class trisycl::vendor::xilinx::acap::debug::bsp_checker . . . 298
9.4 Graphics support for CGRA-like interaction . . . 300
9.4.1 Detailed Description . . . 301
9.5 LaTeX support for graphics output . . . 301
9.5.1 Detailed Description . . . 301
9.6 Data access and storage in SYCL . . . 302
9.6.1 Detailed Description . . . 303
9.6.2 Class Documentation . . . 303
9.6.2.1 class trisycl::vendor::trisycl::pipe::cout . . . 303
9.6.2.2 class trisycl::accessor . . . 305
9.6.2.3 class trisycl::detail::accessor_base . . . 319
9.6.2.4 class trisycl::detail::accessor< T, Dimensions, Mode, access::target::local > . . . 321
9.6.2.5 class trisycl::buffer . . . 323
9.6.2.6 class trisycl::detail::accessor . . . 337
9.6.2.7 class trisycl::detail::buffer . . . 346
9.6.2.8 struct trisycl::detail::buffer_base . . . 356
9.6.2.9 class trisycl::detail::buffer_waiter . . . 368
9.6.2.10 struct trisycl::image . . . 370
9.6.2.11 class trisycl::pipe . . . 370
9.6.3 Typedef Documentation . . . 374
9.6.3.1 buffer_allocator . . . 374
9.6.3.2 image_allocator . . . 375
9.6.3.3 map_allocator . . . 375
9.6.4 Function Documentation . . . 375
9.6.4.1 add_buffer_to_task() . . . 375
9.6.4.2 buffer() [1/2] . . . 376
9.6.4.3 buffer() [2/2] . . . 376
9.6.4.4 buffer_add_to_task() . . . 377
9.6.4.5 waiter() . . . 377
9.7 triSYCL extension for storage scopes . . . 378
9.7.1 Detailed Description . . . 378
9.7.2 Class Documentation . . . 378
9.7.2.1 struct trisycl::vendor::trisycl::scope::empty_device_scope . . . 378
9.7.2.2 class trisycl::vendor::trisycl::scope::device . . . 378
9.7.2.3 class trisycl::vendor::trisycl::scope::detail::device . . . 387

9.7.2.4 struct trisycl::vendor::trisycl::scope::empty_platform_scope . . . 392
9.7.2.5 class trisycl::vendor::trisycl::scope::platform . . . 392
9.7.2.6 class trisycl::vendor::trisycl::scope::detail::platform . . . 398
9.7.2.7 struct trisycl::vendor::trisycl::scope::empty_queue_scope . . . 401
9.7.2.8 class trisycl::vendor::trisycl::scope::queue . . . 402
9.7.2.9 class trisycl::vendor::trisycl::scope::detail::queue . . . 414
9.8 Xilinx vendor extensions . . . 418
9.8.1 Detailed Description . . . 419
9.8.2 Class Documentation . . . 419
9.8.2.1 struct trisycl::vendor::xilinx::partition_array . . . 419
9.8.3 Macro Definition Documentation . . . 430
9.8.3.1 _ssdm_op_SpecDataflowPipeline [1/2] . . . 430
9.8.3.2 _ssdm_op_SpecDataflowPipeline [2/2] . . . 431
9.8.3.3 _ssdm_op_SpecPipeline [1/2] . . . 431
9.8.3.4 _ssdm_op_SpecPipeline [2/2] . . . 431
9.8.3.5 _ssdm_SpecArrayPartition [1/2] . . . 431
9.8.3.6 _ssdm_SpecArrayPartition [2/2] . . . 431
9.9 Data access and storage in old version of SYCL . . . 432
9.9.1 Detailed Description . . . 432
9.9.2 Class Documentation . . . 432
9.9.2.1 class trisycl::accessor< DataType, 1, AccessMode, access::target::pipe > . . . 432
9.9.2.2 class trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe > . . . 435
9.9.2.3 class trisycl::host_accessor . . . 437
9.9.2.4 class trisycl::sycl_2_2::pipe . . . 441
9.9.2.5 struct trisycl::detail::sycl_2_2::reserve_id . . . 444
9.9.2.6 class trisycl::detail::sycl_2_2::pipe . . . 446
9.9.2.7 class trisycl::detail::sycl_2_2::pipe_accessor . . . 465
9.9.2.8 struct trisycl::sycl_2_2::pipe_reservation . . . 477
9.9.2.9 class trisycl::detail::sycl_2_2::pipe_reservation . . . 485
9.9.2.10 class trisycl::sycl_2_2::static_pipe . . . 495
9.9.3 Function Documentation . . . 498
9.9.3.1 get_pipe_detail() . . . 498
9.10 Platforms, contexts, accessors and queues . . . 499
9.10.1 Detailed Description . . . 503
9.10.2 Class Documentation . . . 503
9.10.2.1 class trisycl::facade::accessor . . . 503
9.10.2.2 class trisycl::mixin::accessor . . . 513
9.10.2.3 class trisycl::context . . . 524
9.10.2.4 class trisycl::detail::context . . . 531
9.10.2.5 class trisycl::detail::host_context . . . 534
9.10.2.6 class trisycl::device . . . 537
9.10.2.7 class trisycl::detail::device . . . 547

9.10.2.8 class trisycl::facade::device . . . 550
9.10.2.9 class trisycl::device_selector . . . 552
9.10.2.10 class trisycl::device_type_selector . . . 553
9.10.2.11 class trisycl::device_typename_selector . . . 556
9.10.2.12 class trisycl::handler . . . 557
9.10.2.13 class trisycl::kernel . . . 572
9.10.2.14 class trisycl::detail::kernel . . . 576
9.10.2.15 class trisycl::platform . . . 578
9.10.2.16 class trisycl::detail::host_platform . . . 583
9.10.2.17 class trisycl::detail::opencl_platform . . . 587
9.10.2.18 class trisycl::detail::platform . . . 591
9.10.2.19 class trisycl::queue . . . 594
9.10.3 Macro Definition Documentation . . . 604
9.10.3.1 PLATFORM_GET_INFO_STRING . . . 604
9.10.4 Typedef Documentation . . . 604
9.10.4.1 cpu_selector . . . 604
9.10.4.2 default_selector . . . 605
9.10.4.3 device_exec_capabilities . . . 605
9.10.4.4 device_fp_config . . . 605
9.10.4.5 device_queue_properties . . . 605
9.10.4.6 gpu_selector . . . 606
9.10.4.7 host_selector . . . 606
9.10.5 Enumeration Type Documentation . . . 606
9.10.5.1 context . . . 606
9.10.5.2 device . . . 607
9.10.5.3 device_type . . . 609
9.10.5.4 execution_capability . . . 610
9.10.5.5 fp_config . . . 611
9.10.5.6 global_mem_cache_type . . . 611
9.10.5.7 local_mem_type . . . 611
9.10.5.8 partition_affinity_domain . . . 612
9.10.5.9 partition_property . . . 612
9.10.5.10 platform . . . 613
9.10.6 Function Documentation . . . 614
9.10.6.1 context::get_info< info::context::devices >() . . . 614
9.10.6.2 context::get_info< info::context::platform >() . . . 614
9.10.6.3 context::get_info< info::context::reference_count >() . . . 614
9.10.6.4 get_devices() [1/2] . . . 614
9.10.6.5 get_devices() [2/2] . . . 615
9.10.6.6 platform::get_info< info::platform::extensions >() . . . 616
9.10.6.7 queue::get_info< info::queue::context >() . . . 616
9.10.6.8 queue::get_info< info::queue::device >() . . . 617

9.10.6.9 queue::get_info< info::queue::reference_count >() . . . 617
9.10.6.10 select_device() [1/2] . . . 617
9.10.6.11 select_device() [2/2] . . . 618
9.10.6.12 TRISYCL_INFO_PARAM_TRAITS() [1/7] . . . 619
9.10.6.13 TRISYCL_INFO_PARAM_TRAITS() [2/7] . . . 619
9.10.6.14 TRISYCL_INFO_PARAM_TRAITS() [3/7] . . . 619
9.10.6.15 TRISYCL_INFO_PARAM_TRAITS() [4/7] . . . 620
9.10.6.16 TRISYCL_INFO_PARAM_TRAITS() [5/7] . . . 620
9.10.6.17 TRISYCL_INFO_PARAM_TRAITS() [6/7] . . . 620
9.10.6.18 TRISYCL_INFO_PARAM_TRAITS() [7/7] . . . 620
9.10.7 Variable Documentation . . . 620
9.10.7.1 TRISYCL_WEAK_ATTRIB_SUFFIX . . . 620
9.11 Dealing with OpenCL address spaces . . . 621
9.11.1 Detailed Description . . . 623
9.11.2 Class Documentation . . . 623
9.11.2.1 struct trisycl::detail::ocl_type . . . 623
9.11.2.2 struct trisycl::detail::ocl_type< T, access::address_space::constant_space > . . . 623
9.11.2.3 struct trisycl::detail::ocl_type< T, access::address_space::global_space > . . . 624
9.11.2.4 struct trisycl::detail::ocl_type< T, access::address_space::local_space > . . . 624
9.11.2.5 struct trisycl::detail::ocl_type< T, access::address_space::private_space > . . . 625
9.11.2.6 struct trisycl::detail::address_space_array . . . 625
9.11.2.7 struct trisycl::detail::address_space_fundamental . . . 627
9.11.2.8 struct trisycl::detail::address_space_object . . . 630
9.11.2.9 struct trisycl::detail::address_space_ptr . . . 632
9.11.2.10 struct trisycl::detail::address_space_base . . . 635
9.11.2.11 struct trisycl::detail::address_space_variable . . . 637
9.11.3 Typedef Documentation . . . 640
9.11.3.1 addr_space . . . 640
9.11.3.2 constant . . . 641
9.11.3.3 constant_ptr . . . 641
9.11.3.4 generic . . . 641
9.11.3.5 global . . . 642
9.11.3.6 global_ptr . . . 642
9.11.3.7 local . . . 642
9.11.3.8 local_ptr . . . 643
9.11.3.9 multi_ptr . . . 643
9.11.3.10 priv . . . 643
9.11.3.11 private_ptr . . . 644
9.11.4 Function Documentation . . . 644
9.11.4.1 make_ptr() [1/2] . . . 644
9.11.4.2 make_ptr() [2/2] . . . 645
9.12 Helpers to do array and tuple conversion . . . 645

9.12.1 Detailed Description . . . 646
9.12.2 Class Documentation . . . 646
9.12.2.1 struct trisycl::detail::expand_to_vector . . . 646
9.12.2.2 struct trisycl::detail::expand_to_vector< V, Tuple, true > . . . 646
9.12.3 Function Documentation . . . 646
9.12.3.1 expand() [1/3] . . . 647
9.12.3.2 expand() [2/3] . . . 647
9.12.3.3 expand() [3/3] . . . 648
9.12.3.4 fill_tuple() . . . 648
9.12.3.5 tuple_to_array() . . . 649
9.12.3.6 tuple_to_array_iterate() . . . 649
9.13 Some helpers for the implementation . . . 650
9.13.1 Detailed Description . . . 651
9.13.2 Class Documentation . . . 651
9.13.2.1 struct trisycl::detail::container_element_aspect . . . 651
9.13.2.2 struct trisycl::detail::overloaded . . . 652
9.13.2.3 struct trisycl::detail::small_array_tag . . . 653
9.13.2.4 struct trisycl::detail::small_array . . . 653
9.13.2.5 struct trisycl::detail::small_array_sycl . . . 658
9.13.3 Macro Definition Documentation . . . 661
9.13.3.1 TRISYCL_BOOST_OPERATOR_VECTOR_OP . . . 661
9.13.3.2 TRISYCL_LOGICAL_OPERATOR_VECTOR_OP . . . 661
9.13.3.3 TRISYCL_OPERATOR_BASIC_TYPE_OP . . . 662
9.13.3.4 TRISYCL_UNARY_OPERATOR_VECTOR_OP . . . 662
9.13.3.5 TRISYCL_UNARY_POSTFIX_OPERATOR_VECTOR_OP . . . 662
9.13.3.6 TRISYCL_UNARY_PREFIX_OPERATOR_VECTOR_OP . . . 663
9.13.3.7 TRISYCL_UNIMPL . . . 663
9.13.4 Function Documentation . . . 663
9.13.4.1 linear_id() . . . 663
9.13.4.2 overloaded() . . . 665
9.13.4.3 unimplemented() . . . 665
9.14 Debugging and tracing support . . . 665
9.14.1 Detailed Description . . . 666
9.14.2 Class Documentation . . . 666
9.14.2.1 struct trisycl::detail::debug . . . 666
9.14.2.2 struct trisycl::detail::display_vector . . . 667
9.14.3 Function Documentation . . . 668
9.14.3.1 trace_kernel() . . . 668
9.15 Manage default configuration and types . . . 669
9.15.1 Detailed Description . . . 670
9.15.2 Macro Definition Documentation . . . 670
9.15.2.1 __SYCL_SINGLE_SOURCE__ . . . 670

9.15.2.2 BOOST_COMPUTE_CL_VERSION_1_2 . . . 670
9.15.2.3 CL_SYCL_LANGUAGE_VERSION . . . 670
9.15.2.4 TRISYCL_CL_LANGUAGE_VERSION . . . 671
9.15.2.5 TRISYCL_DEBUG_ONLY . . . 671
9.15.2.6 TRISYCL_DEVICE_ONLY . . . 671
9.15.2.7 TRISYCL_MAKE_BOOST_CIRCULARBUFFER_THREAD_SAFE . . . 671
9.15.2.8 TRISYCL_OPENCL_ONLY . . . 671
9.16 Opencl_spir_helpers . . . 672
9.16.1 Detailed Description . . . 673
9.16.2 Function Documentation . . . 673
9.16.2.1 create_parallel_for_arg() . . . 673
9.16.2.2 make_array_with_func() . . . 674
9.16.2.3 make_spir_h_item() . . . 674
9.16.2.4 make_spir_nd_range() . . . 675
9.17 Device-side runtime implementation . . . 675
9.17.1 Detailed Description . . . 676
9.17.2 Class Documentation . . . 676
9.17.2.1 class trisycl::drt::accessor . . . 676
9.17.3 Function Documentation . . . 678
9.17.3.1 serialize_accessor_arg() . . . 678
9.17.3.2 serialize_arg() . . . 679
9.17.3.3 set_kernel() . . . 680
9.17.4 Variable Documentation . . . 681
9.17.4.1 kernel_IR . . . 681
9.18 Error handling . . . 681
9.18.1 Detailed Description . . . 683
9.18.2 Class Documentation . . . 683
9.18.2.1 struct trisycl::error_handler . . . 683
9.18.2.2 struct trisycl::exception_list . . . 685
9.18.2.3 class trisycl::exception . . . 685
9.18.2.4 class trisycl::cl_exception . . . 688
9.18.2.5 struct trisycl::async_exception . . . 690
9.18.2.6 class trisycl::runtime_error . . . 691
9.18.2.7 class trisycl::kernel_error . . . 692
9.18.2.8 class trisycl::accessor_error . . . 693
9.18.2.9 class trisycl::nd_range_error . . . 694
9.18.2.10 class trisycl::event_error . . . 695
9.18.2.11 class trisycl::invalid_parameter_error . . . 696
9.18.2.12 class trisycl::device_error . . . 697
9.18.2.13 class trisycl::compile_program_error . . . 699
9.18.2.14 class trisycl::link_program_error . . . 700
9.18.2.15 class trisycl::invalid_object_error . . . 701

9.18.2.16 class trisycl::memory_allocation_error . . . 702
9.18.2.17 class trisycl::pipe_error . . . 703
9.18.2.18 class trisycl::platform_error . . . 704
9.18.2.19 class trisycl::profiling_error . . . 705
9.18.2.20 class trisycl::feature_not_supported . . . 706
9.18.2.21 class trisycl::non_cl_error . . . 707
9.18.3 Typedef Documentation . . . 708
9.18.3.1 async_handler . . . 709
9.18.3.2 exception_ptr . . . 709
9.19 Expressing parallelism through kernels . . . 709
9.19.1 Detailed Description . . . 711
9.19.2 Class Documentation . . . 711
9.19.2.1 struct trisycl::group . . . 711
9.19.2.2 struct trisycl::h_item . . . 724
9.19.2.3 class trisycl::id . . . 740
9.19.2.4 class trisycl::item . . . 742
9.19.2.5 struct trisycl::nd_item . . . 751
9.19.2.6 struct trisycl::nd_range . . . 761
9.19.2.7 struct trisycl::detail::parallel_for_iterate . . . 767
9.19.2.8 struct trisycl::detail::parallel_for_iterate< 0, Range, ParallelForFunctor, Id > . . . 767
9.19.2.9 class trisycl::range . . . 768
9.19.3 Function Documentation . . . 770
9.19.3.1 capture_arg_v() . . . 770
9.19.3.2 id() . . . 771
9.19.3.3 parallel_for() [1/4] . . . 771
9.19.3.4 parallel_for() [2/4] . . . 773
9.19.3.5 parallel_for() [3/4] . . . 774
9.19.3.6 parallel_for() [4/4] . . . 775
9.19.3.7 parallel_for_global_offset() . . . 776
9.19.3.8 parallel_for_iterate() . . . 777
9.19.3.9 parallel_for_workgroup() . . . 777
9.19.3.10 parallel_for_workitem() . . . 778
9.19.3.11 parallel_for_workitem_in_group() . . . 781
9.19.3.12 range() . . . 782
9.19.3.13 to_sycl_id() [1/3] . . . 782
9.19.3.14 to_sycl_id() [2/3] . . . 782
9.19.3.15 to_sycl_id() [3/3] . . . 783
9.19.3.16 to_tbb_range() [1/3] . . . 783
9.19.3.17 to_tbb_range() [2/3] . . . 784
9.19.3.18 to_tbb_range() [3/3] . . . 784
9.20 Vector types in SYCL . . . 784
9.20.1 Detailed Description . . . 786

9.20.2 Class Documentation . . . 786
9.20.2.1 struct trisycl::elem . . . 786
9.20.2.2 class trisycl::vec< DataType, 1 > . . . 787
9.20.2.3 class trisycl::vec< DataType, 2 > . . . 789
9.20.2.4 class trisycl::vec< DataType, 3 > . . . 793
9.20.2.5 class trisycl::vec< DataType, 4 > . . . 798
9.20.2.6 class trisycl::vec< DataType, 8 > . . . 805
9.20.2.7 class trisycl::vec< DataType, 16 > . . . 809
9.20.3 Macro Definition Documentation . . . 814
9.20.3.1 TRISYCL_DECLARE_S . . . 815
9.20.3.2 TRISYCL_DEFINE_VEC_TYPE . . . 815
9.20.3.3 TRISYCL_DEFINE_VEC_TYPE_SIZE . . . 815
9.20.3.4 TRISYCL_GEN_SWIZ2 . . . 816
9.20.3.5 TRISYCL_GEN_SWIZ3 . . . 816
9.20.3.6 TRISYCL_GEN_SWIZ4 . . . 816
9.20.4 Typedef Documentation . . . 817
9.20.4.1 __swizzled_vec__ . . . 817
9.20.5 Variable Documentation . . . 817
9.20.5.1 a . . . 817
9.20.5.2 b . . . 817
9.20.5.3 g . . . 817
9.20.5.4 r . . . 817
9.20.5.5 s0 . . . 818
9.20.5.6 s1 . . . 818
9.20.5.7 s2 . . . 818
9.20.5.8 s3 . . . 818
9.20.5.9 s4 . . . 819
9.20.5.10 s5 . . . 819
9.20.5.11 s6 . . . 819
9.20.5.12 s7 . . . 819
9.20.5.13 s8 . . . 819
9.20.5.14 s9 . . . 820
9.20.5.15 sA . . . 820
9.20.5.16 sB . . . 820
9.20.5.17 sC . . . 820
9.20.5.18 sD . . . 820
9.20.5.19 sE . . . 821
9.20.5.20 sF . . . 821
9.20.5.21 w . . . 821
9.20.5.22 x . . . 821
9.20.5.23 y . . . 821
9.20.5.24 z . . . 821

10 Namespace Documentation **823**

10.1 cl Namespace Reference . . . 823

10.2 cl::sycl Namespace Reference . . . 823

10.2.1 Detailed Description . . . 823

10.3 std Namespace Reference . . . 823

10.3.1 Function Documentation . . . 824

10.3.1.1 get() [1/2] . . . 824

10.3.1.2 get() [2/2] . . . 824

10.4 sycl Namespace Reference . . . 825

10.4.1 Detailed Description . . . 825

10.5 trisycl Namespace Reference . . . 825

10.5.1 Detailed Description . . . 831

10.5.2 Typedef Documentation . . . 831

10.5.2.1 exception_ptr_class . . . 831

10.5.2.2 function_class . . . 831

10.5.2.3 hash_class . . . 831

10.5.2.4 mutex_class . . . 832

10.5.2.5 shared_ptr_class . . . 832

10.5.2.6 string_class . . . 832

10.5.2.7 unique_ptr_class . . . 832

10.5.2.8 vector_class . . . 832

10.5.2.9 weak_ptr_class . . . 832

10.5.3 Enumeration Type Documentation . . . 832

10.5.3.1 program_state . . . 832

10.5.3.2 rounding_mode . . . 833

10.5.4 Function Documentation . . . 833

10.5.4.1 cross() . . . 833

10.5.4.2 dot() . . . 834

10.5.4.3 event::get_info< info::event::command_execution_status >() . . . 834

10.5.4.4 event::get_info< info::event::reference_count >() . . . 834

10.5.4.5 event::get_profiling_info< info::event_profiling::command_end >() . . . 834

10.5.4.6 event::get_profiling_info< info::event_profiling::command_start >() . . . 835

10.5.4.7 event::get_profiling_info< info::event_profiling::command_submit >() . . . 835

10.5.4.8 fmax() . . . 835

10.5.4.9 fmin() . . . 836

10.5.4.10 length() . . . 836

10.5.4.11 max() . . . 836

10.5.4.12 min() [1/2] . . . 837

10.5.4.13 min() [2/2] . . . 838

10.5.4.14 normalize() . . . 838

10.5.4.15 sign() [1/2] . . . 838

10.5.4.16 sign() [2/2] . . . 839

10.5.4.17 TRISYCL_MATH_WRAP() . . . 839
10.5.4.18 TRISYCL_MATH_WRAP2s() [1/3] . . . 839
10.5.4.19 TRISYCL_MATH_WRAP2s() [2/3] . . . 839
10.5.4.20 TRISYCL_MATH_WRAP2s() [3/3] . . . 839
10.5.5 Variable Documentation . . . 840
10.5.5.1 y . . . 840
10.5.5.2 z . . . 840
10.6 trisycl::access Namespace Reference . . . 840
10.6.1 Detailed Description . . . 841
10.6.2 Enumeration Type Documentation . . . 841
10.6.2.1 address_space . . . 841
10.6.2.2 fence_space . . . 842
10.6.2.3 mode . . . 842
10.6.2.4 target . . . 842
10.7 trisycl::detail Namespace Reference . . . 844
10.7.1 Detailed Description . . . 849
10.7.2 Typedef Documentation . . . 849
10.7.2.1 __swizzled_base_vec__ . . . 849
10.7.2.2 all_true . . . 849
10.7.3 Function Documentation . . . 850
10.7.3.1 add_buffer_to_task() . . . 850
10.7.3.2 assign_from_tuple() . . . 850
10.7.3.3 instantiate_kernel() . . . 850
10.7.3.4 launch_device_kernel() . . . 851
10.7.3.5 rangeify() . . . 851
10.7.3.6 set_kernel_task_marker() . . . 852
10.7.4 Variable Documentation . . . 853
10.7.4.1 alignment_v . . . 853
10.7.4.2 allow_any_dimension . . . 853
10.7.4.3 enum_count . . . 853
10.7.4.4 enum_distance . . . 853
10.7.4.5 is_range_v . . . 854
10.7.4.6 prevent_arguments_from_optimization . . . 854
10.7.4.7 underlying_value . . . 854
10.7.4.8 use_native_work_item . . . 854
10.8 trisycl::detail::spir Namespace Reference . . . 855
10.9 trisycl::detail::sycl_2_2 Namespace Reference . . . 855
10.10 trisycl::drt Namespace Reference . . . 855
10.10.1 Detailed Description . . . 856
10.11 trisycl::drt::code Namespace Reference . . . 856
10.11.1 Detailed Description . . . 856
10.12 trisycl::extension Namespace Reference . . . 856

10.13 trisycl::extension::ce Namespace Reference . . . 857
10.14 trisycl::facade Namespace Reference . . . 857
10.15 trisycl::info Namespace Reference . . . 857
10.15.1 Typedef Documentation . . . 859
10.15.1.1 queue_profiling . . . 859
10.15.2 Enumeration Type Documentation . . . 859
10.15.2.1 event . . . 859
10.15.2.2 event_command_status . . . 859
10.15.2.3 event_profiling . . . 860
10.15.2.4 queue . . . 860
10.15.3 Function Documentation . . . 861
10.15.3.1 TRISYCL_INFO_PARAM_TRAITS() [1/3] . . . 861
10.15.3.2 TRISYCL_INFO_PARAM_TRAITS() [2/3] . . . 861
10.15.3.3 TRISYCL_INFO_PARAM_TRAITS() [3/3] . . . 861
10.16 trisycl::mixin Namespace Reference . . . 861
10.17 trisycl::native Namespace Reference . . . 861
10.18 trisycl::property Namespace Reference . . . 861
10.19 trisycl::property::queue Namespace Reference . . . 862
10.20 trisycl::sycl_2_2 Namespace Reference . . . 862
10.21 trisycl::trisycl Namespace Reference . . . 862
10.21.1 Detailed Description . . . 862
10.22 trisycl::vendor Namespace Reference . . . 862
10.23 trisycl::vendor::trisycl Namespace Reference . . . 862
10.24 trisycl::vendor::trisycl::pipe Namespace Reference . . . 863
10.25 trisycl::vendor::trisycl::random Namespace Reference . . . 863
10.25.1 Detailed Description . . . 863
10.26 trisycl::vendor::trisycl::scope Namespace Reference . . . 863
10.27 trisycl::vendor::trisycl::scope::detail Namespace Reference . . . 864
10.27.1 Typedef Documentation . . . 864
10.27.1.1 storage_type_trait_t . . . 864
10.28 trisycl::vendor::xilinx Namespace Reference . . . 864
10.28.1 Variable Documentation . . . 865
10.28.1.1 dataflow . . . 865
10.28.1.2 pipeline . . . 865
10.29 trisycl::vendor::xilinx::acap Namespace Reference . . . 866
10.30 trisycl::vendor::xilinx::acap::aie Namespace Reference . . . 866
10.31 trisycl::vendor::xilinx::acap::aie::detail Namespace Reference . . . 867
10.32 trisycl::vendor::xilinx::acap::aie::layout Namespace Reference . . . 868
10.32.1 Typedef Documentation . . . 868
10.32.1.1 xc10s10 . . . 868
10.32.1.2 xc10s80 . . . 868
10.32.1.3 xcvc1902 . . . 869

10.32.1.4 xcve1152 . . . 869
10.33 trisycl::vendor::xilinx::acap::debug Namespace Reference . . . 869
10.34 trisycl::vendor::xilinx::graphics Namespace Reference . . . 869
10.34.1 Typedef Documentation . . . 869
10.34.1.1 rgb . . . 870
10.35 trisycl::vendor::xilinx::latex Namespace Reference . . . 870
10.36 trisycl::vendor::xilinx::partition Namespace Reference . . . 870
10.36.1 Detailed Description . . . 870
10.36.2 Enumeration Type Documentation . . . 870
10.36.2.1 type [1/2] . . . 870
10.36.2.2 type [2/2] . . . 871
10.37 trisycl::views Namespace Reference . . . 871
10.37.1 Variable Documentation . . . 872
10.37.1.1 enum_type . . . 872
10.37.1.2 enum_underlying_type . . . 872

11 Concept Documentation **875**
11.1 trisycl::detail::is_small_array Concept Reference . . . 875
11.1.1 Concept definition . . . 875
11.1.2 Detailed Description . . . 875

12 Class Documentation **877**
12.1 trisycl::vendor::xilinx::acap::aie::accessor< Buffer > Class Template Reference . . . 877
12.1.1 Detailed Description . . . 878
12.1.2 Member Typedef Documentation . . . 879
12.1.2.1 base [1/2] . . . 879
12.1.2.2 base [2/2] . . . 879
12.1.3 Constructor & Destructor Documentation . . . 879
12.1.3.1 accessor() [1/10] . . . 879
12.1.3.2 accessor() [2/10] . . . 880
12.1.3.3 accessor() [3/10] . . . 880
12.1.3.4 accessor() [4/10] . . . 880
12.1.3.5 accessor() [5/10] . . . 881
12.1.3.6 accessor() [6/10] . . . 881
12.1.3.7 accessor() [7/10] . . . 882
12.1.3.8 accessor() [8/10] . . . 882
12.1.3.9 accessor() [9/10] . . . 883
12.1.3.10 accessor() [10/10] . . . 883
12.2 AccessorMixIn Class Reference . . . 884
12.3 trisycl::detail::alignment< T > Class Template Reference . . . 884
12.3.1 Detailed Description . . . 885
12.3.2 Member Data Documentation . . . 885
12.3.2.1 value . . . 885

12.4 trisycl::detail::alignment<::trisycl::vec< DataType, NumElements > > Class Template Reference . 885
12.4.1 Detailed Description . . . 886
12.4.2 Member Function Documentation . . . 886
12.4.2.1 get_align() . . . 886
12.4.3 Member Data Documentation . . . 886
12.4.3.1 value . . . 886
12.5 trisycl::vendor::xilinx::graphics::application Struct Reference . . . 887
12.5.1 Detailed Description . . . 888
12.5.2 Constructor & Destructor Documentation . . . 888
12.5.2.1 ~application() [1/2] . . . 888
12.5.2.2 ~application() [2/2] . . . 889
12.5.3 Member Function Documentation . . . 889
12.5.3.1 image_grid() [1/2] . . . 889
12.5.3.2 image_grid() [2/2] . . . 889
12.5.3.3 is_done() [1/2] . . . 890
12.5.3.4 is_done() [2/2] . . . 890
12.5.3.5 is_done_barrier() [1/2] . . . 890
12.5.3.6 is_done_barrier() [2/2] . . . 890
12.5.3.7 start() [1/2] . . . 891
12.5.3.8 start() [2/2] . . . 892
12.5.3.9 update_tile_data_image() [1/2] . . . 893
12.5.3.10 update_tile_data_image() [2/2] . . . 893
12.5.3.11 wait() [1/2] . . . 894
12.5.3.12 wait() [2/2] . . . 894
12.5.4 Member Data Documentation . . . 894
12.5.4.1 initialized . . . 895
12.5.4.2 t . . . 895
12.5.4.3 w . . . 895
12.6 trisycl::vendor::xilinx::acap::aie::connection::base Struct Reference . . . 895
12.6.1 Detailed Description . . . 896
12.6.2 Constructor & Destructor Documentation . . . 896
12.6.2.1 base() [1/4] . . . 897
12.6.2.2 base() [2/4] . . . 897
12.6.2.3 base() [3/4] . . . 897
12.6.2.4 base() [4/4] . . . 897
12.6.3 Member Function Documentation . . . 897
12.6.3.1 pipe_of() [1/2] . . . 897
12.6.3.2 pipe_of() [2/2] . . . 898
12.6.4 Member Data Documentation . . . 898
12.6.4.1 p . . . 898
12.7 trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim > Struct Template Reference . . . 899
12.7.1 Detailed Description . . . 899

12.7.2 Member Data Documentation . . . 900
12.7.2.1 ele_in_each_physical_mem . . . 900
12.7.2.2 partition_dim . . . 900
12.7.2.3 partition_type . . . 901
12.8 trisycl::detail::bool_pack< ... > Struct Template Reference . . . 901
12.8.1 Detailed Description . . . 901
12.9 trisycl::detail::cache< Key, Value > Class Template Reference . . . 901
12.9.1 Detailed Description . . . 902
12.9.2 Member Typedef Documentation . . . 902
12.9.2.1 key_type . . . 902
12.9.2.2 value_type . . . 903
12.9.3 Member Function Documentation . . . 903
12.9.3.1 get_or_register() . . . 903
12.9.3.2 remove() . . . 904
12.9.4 Member Data Documentation . . . 905
12.9.4.1 c . . . 905
12.9.4.2 m . . . 906
12.10 trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t Struct Reference . . . 906
12.10.1 Detailed Description . . . 906
12.10.2 Constructor & Destructor Documentation . . . 907
12.10.2.1 cgh_t() [1/2] . . . 907
12.10.2.2 cgh_t() [2/2] . . . 907
12.10.3 Member Function Documentation . . . 907
12.10.3.1 single_task() [1/2] . . . 907
12.10.3.2 single_task() [2/2] . . . 908
12.10.4 Member Data Documentation . . . 908
12.10.4.1 ti . . . 908
12.11 trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH > Struct Template Reference . . . 909
12.11.1 Detailed Description . . . 910
12.11.2 Constructor & Destructor Documentation . . . 910
12.11.2.1 command_group() [1/2] . . . 910
12.11.2.2 command_group() [2/2] . . . 911
12.11.3 Member Function Documentation . . . 911
12.11.3.1 device_scope() [1/2] . . . 911
12.11.3.2 device_scope() [2/2] . . . 911
12.11.3.3 operator::trisycl::handler &() [1/2] . . . 911
12.11.3.4 operator::trisycl::handler &() [2/2] . . . 912
12.11.3.5 parallel_for() [1/2] . . . 912
12.11.3.6 parallel_for() [2/2] . . . 913
12.11.3.7 platform_scope() [1/2] . . . 913
12.11.3.8 platform_scope() [2/2] . . . 914

12.11.3.9 single_task() [1/2] . . . 914
12.11.3.10 single_task() [2/2] . . . 915
12.11.4 Member Data Documentation . . . 915
12.11.4.1 cgh . . . 915
12.11.4.2 d . . . 916
12.12 trisycl::vendor::xilinx::acap::aie::communicator_port Class Reference . . . 916
12.12.1 Detailed Description . . . 917
12.12.2 Constructor & Destructor Documentation . . . 917
12.12.2.1 ~communicator_port() [1/2] . . . 918
12.12.2.2 ~communicator_port() [2/2] . . . 918
12.12.3 Member Function Documentation . . . 918
12.12.3.1 operator<<() [1/2] . . . 918
12.12.3.2 operator<<() [2/2] . . . 919
12.12.3.3 operator>>() [1/2] . . . 919
12.12.3.4 operator>>() [2/2] . . . 920
12.12.3.5 read() [1/2] . . . 920
12.12.3.6 read() [2/2] . . . 921
12.12.3.7 try_read() [1/2] . . . 921
12.12.3.8 try_read() [2/2] . . . 921
12.12.3.9 try_write() [1/2] . . . 922
12.12.3.10 try_write() [2/2] . . . 922
12.12.3.11 write() [1/2] . . . 922
12.12.3.12 write() [2/2] . . . 923
12.13 trisycl::vendor::xilinx::partition::complete< PDim > Struct Template Reference . . . 923
12.13.1 Detailed Description . . . 924
12.13.2 Member Data Documentation . . . 925
12.13.2.1 partition_dim . . . 925
12.13.2.2 partition_type . . . 925
12.14 trisycl::vendor::xilinx::acap::aie::connection Struct Reference . . . 925
12.14.1 Detailed Description . . . 926
12.14.2 Constructor & Destructor Documentation . . . 926
12.14.2.1 connection() [1/2] . . . 926
12.14.2.2 connection() [2/2] . . . 927
12.14.3 Member Function Documentation . . . 927
12.14.3.1 in() [1/2] . . . 927
12.14.3.2 in() [2/2] . . . 927
12.14.3.3 out() [1/2] . . . 928
12.14.3.4 out() [2/2] . . . 928
12.14.4 Member Data Documentation . . . 928
12.14.4.1 p . . . 928
12.15 trisycl::vendor::xilinx::latex::context Class Reference . . . 928
12.15.1 Detailed Description . . . 930

12.15.2 Constructor & Destructor Documentation . . . 930
12.15.2.1 context() [1/2] . . . 930
12.15.2.2 context() [2/2] . . . 930
12.15.3 Member Function Documentation . . . 931
12.15.3.1 add() [1/2] . . . 931
12.15.3.2 add() [2/2] . . . 931
12.15.3.3 clean_node() [1/2] . . . 931
12.15.3.4 clean_node() [2/2] . . . 932
12.15.3.5 display() [1/2] . . . 932
12.15.3.6 display() [2/2] . . . 932
12.15.3.7 finalize() [1/2] . . . 933
12.15.3.8 finalize() [2/2] . . . 933
12.15.3.9 scale() [1/2] . . . 933
12.15.3.10 scale() [2/2] . . . 933
12.15.4 Member Data Documentation . . . 934
12.15.4.1 file_name . . . 934
12.15.4.2 finalized . . . 934
12.15.4.3 max_size . . . 934
12.15.4.4 out . . . 934
12.15.4.5 scaling_factor . . . 935
12.15.4.6 size . . . 935
12.16 trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch Struct Reference . 935
12.16.1 Detailed Description . . . 939
12.16.2 Member Enumeration Documentation . . . 939
12.16.2.1 master_port_layout [1/2] . . . 939
12.16.2.2 master_port_layout [2/2] . . . 941
12.16.2.3 slave_port_layout [1/2] . . . 943
12.16.2.4 slave_port_layout [2/2] . . . 945
12.16.3 Friends And Related Function Documentation . . . 947
12.16.3.1 is_axi_master [1/4] . . . 948
12.16.3.2 is_axi_master [2/4] . . . 948
12.16.3.3 is_axi_master [3/4] . . . 948
12.16.3.4 is_axi_master [4/4] . . . 948
12.16.4 Member Data Documentation . . . 949
12.16.4.1 fifo_depth . . . 949
12.16.4.2 interconnect . . . 949
12.16.4.3 latency . . . 949
12.16.4.4 m_dma_range . . . 950
12.16.4.5 m_dma_size . . . 950
12.16.4.6 m_east_range . . . 950
12.16.4.7 m_east_size . . . 951
12.16.4.8 m_fifo_range . . . 951

12.16.4.9 m_fifo_size . . . 951
12.16.4.10 m_me_range . . . 952
12.16.4.11 m_me_size . . . 952
12.16.4.12 m_north_range . . . 952
12.16.4.13 m_north_size . . . 953
12.16.4.14 m_south_range . . . 953
12.16.4.15 m_south_size . . . 953
12.16.4.16 m_tile_ctrl_range . . . 954
12.16.4.17 m_tile_ctrl_size . . . 954
12.16.4.18 m_west_range . . . 954
12.16.4.19 m_west_size . . . 955
12.16.4.20 nb_master_port . . . 955
12.16.4.21 nb_slave_port . . . 955
12.16.4.22 s_dma_range . . . 956
12.16.4.23 s_dma_size . . . 956
12.16.4.24 s_east_range . . . 956
12.16.4.25 s_east_size . . . 956
12.16.4.26 s_fifo_range . . . 957
12.16.4.27 s_fifo_size . . . 957
12.16.4.28 s_me_range . . . 957
12.16.4.29 s_me_size . . . 958
12.16.4.30 s_north_range . . . 958
12.16.4.31 s_north_size . . . 958
12.16.4.32 s_south_range . . . 959
12.16.4.33 s_south_size . . . 959
12.16.4.34 s_tile_ctrl_range . . . 959
12.16.4.35 s_tile_ctrl_size . . . 960
12.16.4.36 s_west_range . . . 960
12.16.4.37 s_west_size . . . 960
12.17 trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim > Struct Template Reference . . . 961
12.17.1 Detailed Description . . . 961
12.17.2 Member Data Documentation . . . 962
12.17.2.1 partition_dim . . . 962
12.17.2.2 partition_type . . . 962
12.17.2.3 physical_mem_num . . . 963
12.18 trisycl::trisycl::default_error_handler Struct Reference . . . 963
12.18.1 Detailed Description . . . 964
12.18.2 Member Function Documentation . . . 964
12.18.2.1 report_error() . . . 964
12.19 trisycl::vendor::xilinx::acap::aie::detail::device< Layout > Struct Template Reference . . . 965
12.19.1 Detailed Description . . . 967
12.19.2 Member Typedef Documentation . . . 968

12.19.2.1 cass [1/2] . . . 968
12.19.2.2 cass [2/2] . . . 968
12.19.2.3 cmp [1/2] . . . 968
12.19.2.4 cmp [2/2] . . . 968
12.19.2.5 csp [1/2] . . . 969
12.19.2.6 csp [2/2] . . . 969
12.19.2.7 geo [1/2] . . . 969
12.19.2.8 geo [2/2] . . . 969
12.19.2.9 sass [1/2] . . . 970
12.19.2.10 sass [2/2] . . . 970
12.19.2.11 smp [1/2] . . . 970
12.19.2.12 smp [2/2] . . . 970
12.19.2.13 ssp [1/2] . . . 971
12.19.2.14 ssp [2/2] . . . 971
12.19.3 Constructor & Destructor Documentation . . . 971
12.19.3.1 device() [1/2] . . . 971
12.19.3.2 device() [2/2] . . . 972
12.19.4 Member Function Documentation . . . 973
12.19.4.1 cascade() [1/2] . . . 973
12.19.4.2 cascade() [2/2] . . . 974
12.19.4.3 connect() [1/2] . . . 974
12.19.4.4 connect() [2/2] . . . 975
12.19.4.5 display() [1/2] . . . 976
12.19.4.6 display() [2/2] . . . 977
12.19.4.7 for_each_tile() [1/2] . . . 977
12.19.4.8 for_each_tile() [2/2] . . . 978
12.19.4.9 for_each_tile_index() [1/2] . . . 979
12.19.4.10 for_each_tile_index() [2/2] . . . 979
12.19.4.11 for_each_tile_neighborhood() [1/2] . . . 980
12.19.4.12 for_each_tile_neighborhood() [2/2] . . . 981
12.19.4.13 for_each_tile_x_index() [1/2] . . . 982
12.19.4.14 for_each_tile_x_index() [2/2] . . . 982
12.19.4.15 for_each_tile_y_index() [1/2] . . . 983
12.19.4.16 for_each_tile_y_index() [2/2] . . . 983
12.19.4.17 mem() [1/2] . . . 984
12.19.4.18 mem() [2/2] . . . 984
12.19.4.19 shim() [1/2] . . . 985
12.19.4.20 shim() [2/2] . . . 986
12.19.4.21 tile() [1/2] . . . 987
12.19.4.22 tile() [2/2] . . . 988
12.19.5 Member Data Documentation . . . 989
12.19.5.1 cs . . . 989

12.19.5.2 fiber_executor . . . 989
12.19.5.3 shims . . . 990
12.19.5.4 ti . . . 990
12.20 trisycl::vendor::xilinx::acap::aie::device< Layout > Class Template Reference . . . 990
12.20.1 Detailed Description . . . 993
12.20.2 Member Typedef Documentation . . . 994
12.20.2.1 cass [1/2] . . . 994
12.20.2.2 cass [2/2] . . . 994
12.20.2.3 cmp [1/2] . . . 994
12.20.2.4 cmp [2/2] . . . 994
12.20.2.5 csp [1/2] . . . 995
12.20.2.6 csp [2/2] . . . 995
12.20.2.7 dd [1/2] . . . 995
12.20.2.8 dd [2/2] . . . 995
12.20.2.9 facade_t [1/2] . . . 995
12.20.2.10 facade_t [2/2] . . . 996
12.20.2.11 geo [1/2] . . . 996
12.20.2.12 geo [2/2] . . . 996
12.20.2.13 sass [1/2] . . . 996
12.20.2.14 sass [2/2] . . . 996
12.20.2.15 smp [1/2] . . . 997
12.20.2.16 smp [2/2] . . . 997
12.20.2.17 ssp [1/2] . . . 997
12.20.2.18 ssp [2/2] . . . 997
12.20.3 Constructor & Destructor Documentation . . . 997
12.20.3.1 device() [1/2] . . . 997
12.20.3.2 device() [2/2] . . . 998
12.20.4 Member Function Documentation . . . 998
12.20.4.1 cascade() [1/2] . . . 998
12.20.4.2 cascade() [2/2] . . . 998
12.20.4.3 connect() [1/2] . . . 998
12.20.4.4 connect() [2/2] . . . 999
12.20.4.5 display() [1/2] . . . 1000
12.20.4.6 display() [2/2] . . . 1000
12.20.4.7 for_each_tile() [1/2] . . . 1000
12.20.4.8 for_each_tile() [2/2] . . . 1001
12.20.4.9 for_each_tile_index() [1/2] . . . 1001
12.20.4.10 for_each_tile_index() [2/2] . . . 1002
12.20.4.11 for_each_tile_neighborhood() [1/2] . . . 1002
12.20.4.12 for_each_tile_neighborhood() [2/2] . . . 1003
12.20.4.13 for_each_tile_x_index() [1/2] . . . 1003
12.20.4.14 for_each_tile_x_index() [2/2] . . . 1003

12.20.4.15 for_each_tile_y_index() [1/2] . . . 1004
12.20.4.16 for_each_tile_y_index() [2/2] . . . 1004
12.20.4.17 mem() [1/2] . . . 1005
12.20.4.18 mem() [2/2] . . . 1005
12.20.4.19 program() [1/2] . . . 1006
12.20.4.20 program() [2/2] . . . 1006
12.20.4.21 queue() [1/2] . . . 1006
12.20.4.22 queue() [2/2] . . . 1007
12.20.4.23 run() [1/4] . . . 1007
12.20.4.24 run() [2/4] . . . 1008
12.20.4.25 run() [3/4] . . . 1009
12.20.4.26 run() [4/4] . . . 1009
12.20.4.27 shim() [1/2] . . . 1010
12.20.4.28 shim() [2/2] . . . 1010
12.20.4.29 tile() [1/2] . . . 1011
12.20.4.30 tile() [2/2] . . . 1011
12.20.4.31 uniform_run() [1/2] . . . 1012
12.20.4.32 uniform_run() [2/2] . . . 1012
12.20.5 Member Data Documentation . . . 1013
12.20.5.1 size . . . 1013
12.20.5.2 x_max . . . 1013
12.20.5.3 x_min . . . 1014
12.20.5.4 x_size . . . 1014
12.20.5.5 y_max . . . 1014
12.20.5.6 y_min . . . 1014
12.20.5.7 y_size . . . 1015
12.21 trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA > Class Template Reference . . . 1015
12.21.1 Detailed Description . . . 1017
12.21.2 Member Typedef Documentation . . . 1017
12.21.2.1 dma_command [1/2] . . . 1018
12.21.2.2 dma_command [2/2] . . . 1018
12.21.3 Constructor & Destructor Documentation . . . 1018
12.21.3.1 dma() [1/2] . . . 1018
12.21.3.2 ~dma() [1/2] . . . 1019
12.21.3.3 dma() [2/2] . . . 1020
12.21.3.4 ~dma() [2/2] . . . 1020
12.21.4 Member Function Documentation . . . 1021
12.21.4.1 commit_command() [1/2] . . . 1021
12.21.4.2 commit_command() [2/2] . . . 1022
12.21.4.3 join() [1/2] . . . 1022
12.21.4.4 join() [2/2] . . . 1023

12.21.4.5 launch() [1/2] . . . 1023
12.21.4.6 launch() [2/2] . . . 1023
12.21.4.7 pop_command() [1/2] . . . 1024
12.21.4.8 pop_command() [2/2] . . . 1024
12.21.4.9 push_command() [1/2] . . . 1024
12.21.4.10 push_command() [2/2] . . . 1025
12.21.4.11 wait() [1/2] . . . 1025
12.21.4.12 wait() [2/2] . . . 1026
12.21.5 Member Data Documentation . . . 1026
12.21.5.1 c . . . 1026
12.21.5.2 data_mover . . . 1026
12.21.5.3 nb_command . . . 1027
12.21.5.4 waiting_mutex . . . 1027
12.21.5.5 waiting_room . . . 1027
12.22 trisycl::property::queue::enable_profiling Class Reference . . . 1028
12.22.1 Detailed Description . . . 1028
12.22.2 Constructor & Destructor Documentation . . . 1028
12.22.2.1 enable_profiling() . . . 1029
12.23 trisycl::detail::event Struct Reference . . . 1029
12.23.1 Detailed Description . . . 1030
12.23.2 Constructor & Destructor Documentation . . . 1030
12.23.2.1 ∼event() . . . 1030
12.23.3 Member Function Documentation . . . 1030
12.23.3.1 get() . . . 1030
12.23.3.2 get_boost_compute() . . . 1031
12.23.3.3 get_command_execution_status() . . . 1031
12.23.3.4 get_profiling_info() . . . 1031
12.23.3.5 get_reference_count() . . . 1031
12.23.3.6 is_host() . . . 1031
12.23.3.7 wait() . . . 1032
12.24 trisycl::event Class Reference . . . 1032
12.24.1 Detailed Description . . . 1034
12.24.2 Member Typedef Documentation . . . 1034
12.24.2.1 implementation_t . . . 1034
12.24.3 Constructor & Destructor Documentation . . . 1034
12.24.3.1 event() [1/3] . . . 1034
12.24.3.2 event() [2/3] . . . 1035
12.24.3.3 event() [3/3] . . . 1035
12.24.4 Member Function Documentation . . . 1035
12.24.4.1 get() . . . 1035
12.24.4.2 get_boost_compute() . . . 1036
12.24.4.3 get_info() . . . 1036

12.24.4.4 get_profiling_info() . . . 1036
12.24.4.5 get_wait_list() . . . 1036
12.24.4.6 is_host() . . . 1037
12.24.4.7 wait() [1/2] . . . 1037
12.24.4.8 wait() [2/2] . . . 1037
12.24.4.9 wait_and_throw() [1/2] . . . 1037
12.24.4.10 wait_and_throw() [2/2] . . . 1038
12.24.5 Member Data Documentation . . . 1038
12.24.5.1 implementation_t . . . 1038
12.25 trisycl::detail::fiber_pool Class Reference . . . 1038
12.25.1 Detailed Description . . . 1040
12.25.2 Member Typedef Documentation . . . 1040
12.25.2.1 future . . . 1040
12.25.3 Member Enumeration Documentation . . . 1040
12.25.3.1 sched . . . 1040
12.25.4 Constructor & Destructor Documentation . . . 1040
12.25.4.1 fiber_pool() . . . 1041
12.25.4.2 ∼fiber_pool() . . . 1042
12.25.5 Member Function Documentation . . . 1042
12.25.5.1 close() . . . 1042
12.25.5.2 join() . . . 1043
12.25.5.3 run() . . . 1044
12.25.5.4 submit() . . . 1044
12.25.6 Member Data Documentation . . . 1045
12.25.6.1 finish_line . . . 1046
12.25.6.2 joinable . . . 1046
12.25.6.3 pc_shared . . . 1046
12.25.6.4 pc_stealing . . . 1046
12.25.6.5 s . . . 1046
12.25.6.6 starting_block . . . 1047
12.25.6.7 starting_mode . . . 1047
12.25.6.8 submission . . . 1047
12.25.6.9 working_threads . . . 1047
12.26 trisycl::vendor::xilinx::acap::aie::fifo_channel Class Reference . . . 1048
12.26.1 Detailed Description . . . 1049
12.26.2 Member Typedef Documentation . . . 1049
12.26.2.1 value_type [1/2] . . . 1050
12.26.2.2 value_type [2/2] . . . 1050
12.26.3 Member Function Documentation . . . 1050
12.26.3.1 read() [1/2] . . . 1050
12.26.3.2 read() [2/2] . . . 1050
12.26.3.3 try_read() [1/2] . . . 1051

12.26.3.4 try_read() [2/2] . . . 1051
12.26.3.5 try_write() [1/2] . . . 1052
12.26.3.6 try_write() [2/2] . . . 1052
12.26.3.7 write() [1/2] . . . 1052
12.26.3.8 write() [2/2] . . . 1053
12.26.4 Member Data Documentation . . . 1053
12.26.4.1 c . . . 1053
12.26.4.2 capacity . . . 1053
12.27 trisycl::vendor::xilinx::graphics::frame_grid Struct Reference . . . 1054
12.27.1 Detailed Description . . . 1055
12.27.2 Constructor & Destructor Documentation . . . 1055
12.27.2.1 frame_grid() [1/2] . . . 1055
12.27.2.2 frame_grid() [2/2] . . . 1056
12.27.3 Member Function Documentation . . . 1057
12.27.3.1 get_frame() [1/2] . . . 1057
12.27.3.2 get_frame() [2/2] . . . 1058
12.27.3.3 set_close_action() [1/2] . . . 1058
12.27.3.4 set_close_action() [2/2] . . . 1058
12.27.4 Member Data Documentation . . . 1058
12.27.4.1 close_action . . . 1059
12.27.4.2 close_button . . . 1059
12.27.4.3 done . . . 1059
12.27.4.4 frames . . . 1059
12.27.4.5 grid . . . 1060
12.27.4.6 nx . . . 1060
12.27.4.7 ny . . . 1060
12.27.4.8 sw . . . 1060
12.28 trisycl::half Class Reference . . . 1061
12.28.1 Detailed Description . . . 1061
12.28.2 Constructor & Destructor Documentation . . . 1061
12.28.2.1 half() . . . 1061
12.28.3 Member Function Documentation . . . 1061
12.28.3.1 operator>() . . . 1061
12.29 std::hash< trisycl::buffer< T, Dimensions, Allocator > > Struct Template Reference . . . 1061
12.29.1 Detailed Description . . . 1062
12.29.2 Member Function Documentation . . . 1062
12.29.2.1 operator()() . . . 1062
12.30 std::hash< trisycl::context > Struct Reference . . . 1062
12.30.1 Detailed Description . . . 1062
12.30.2 Member Function Documentation . . . 1063
12.30.2.1 operator()() . . . 1063
12.31 std::hash< trisycl::device > Struct Reference . . . 1063

12.31.1 Detailed Description . . . 1063
12.31.2 Member Function Documentation . . . 1063
12.31.2.1 operator()() . . . 1064
12.32 std::hash< trisycl::event > Struct Reference . . . 1064
12.32.1 Detailed Description . . . 1064
12.32.2 Member Function Documentation . . . 1064
12.32.2.1 operator()() . . . 1065
12.33 std::hash< trisycl::kernel > Struct Reference . . . 1065
12.33.1 Detailed Description . . . 1065
12.33.2 Member Function Documentation . . . 1065
12.33.2.1 operator()() . . . 1066
12.34 std::hash< trisycl::platform > Struct Reference . . . 1066
12.34.1 Detailed Description . . . 1066
12.34.2 Member Function Documentation . . . 1066
12.34.2.1 operator()() . . . 1067
12.35 std::hash< trisycl::queue > Struct Reference . . . 1067
12.35.1 Detailed Description . . . 1067
12.35.2 Member Function Documentation . . . 1067
12.35.2.1 operator()() . . . 1067
12.36 trisycl::detail::host_device Class Reference . . . 1068
12.36.1 Detailed Description . . . 1069
12.36.2 Member Function Documentation . . . 1069
12.36.2.1 get() . . . 1069
12.36.2.2 get_boost_compute() . . . 1070
12.36.2.3 get_info() . . . 1070
12.36.2.4 get_platform() . . . 1070
12.36.2.5 has_extension() . . . 1071
12.36.2.6 is_accelerator() . . . 1071
12.36.2.7 is_cpu() . . . 1071
12.36.2.8 is_gpu() . . . 1072
12.36.2.9 is_host() . . . 1072
12.37 trisycl::detail::host_event Class Reference . . . 1072
12.37.1 Detailed Description . . . 1073
12.37.2 Member Function Documentation . . . 1073
12.37.2.1 get() . . . 1074
12.37.2.2 get_boost_compute() . . . 1074
12.37.2.3 get_command_execution_status() . . . 1074
12.37.2.4 get_profiling_info() . . . 1074
12.37.2.5 get_reference_count() . . . 1075
12.37.2.6 is_host() . . . 1075
12.37.2.7 wait() . . . 1075
12.38 trisycl::detail::host_queue Class Reference . . . 1076

12.38.1 Detailed Description . . . 1077
12.38.2 Member Function Documentation . . . 1077
12.38.2.1 get() . . . 1077
12.38.2.2 get_boost_compute() . . . 1077
12.38.2.3 get_context() . . . 1078
12.38.2.4 get_device() . . . 1078
12.38.2.5 is_host() . . . 1079
12.39 trisycl::vendor::xilinx::graphics::image_grid Struct Reference . . . 1079
12.39.1 Detailed Description . . . 1081
12.39.2 Constructor & Destructor Documentation . . . 1082
12.39.2.1 image_grid() [1/2] . . . 1082
12.39.2.2 image_grid() [2/2] . . . 1083
12.39.3 Member Function Documentation . . . 1084
12.39.3.1 get_palette() [1/2] . . . 1084
12.39.3.2 get_palette() [2/2] . . . 1085
12.39.3.3 is_done_barrier() [1/2] . . . 1085
12.39.3.4 is_done_barrier() [2/2] . . . 1085
12.39.3.5 on_key_press_event() [1/2] . . . 1086
12.39.3.6 on_key_press_event() [2/2] . . . 1086
12.39.3.7 submit() [1/2] . . . 1087
12.39.3.8 submit() [2/2] . . . 1088
12.39.3.9 update_tile_data_image() [1/4] . . . 1088
12.39.3.10 update_tile_data_image() [2/4] . . . 1089
12.39.3.11 update_tile_data_image() [3/4] . . . 1090
12.39.3.12 update_tile_data_image() [4/4] . . . 1091
12.39.4 Member Data Documentation . . . 1092
12.39.4.1 cv . . . 1092
12.39.4.2 dispatch_protection . . . 1093
12.39.4.3 dispatcher . . . 1093
12.39.4.4 done_barrier . . . 1093
12.39.4.5 done_has_been_sampled . . . 1093
12.39.4.6 done_snapshot . . . 1094
12.39.4.7 image_x . . . 1094
12.39.4.8 image_y . . . 1094
12.39.4.9 images . . . 1094
12.39.4.10 p . . . 1095
12.39.4.11 work_to_dispatch . . . 1095
12.39.4.12 zoom . . . 1095
12.40 trisycl::vendor::xilinx::acap::aie::connection::input Struct Reference . . . 1096
12.40.1 Detailed Description . . . 1097
12.40.2 Member Function Documentation . . . 1097
12.40.2.1 base() [1/8] . . . 1097

12.40.2.2 base() [2/8] . . . 1098
12.40.2.3 base() [3/8] . . . 1098
12.40.2.4 base() [4/8] . . . 1098
12.40.2.5 base() [5/8] . . . 1098
12.40.2.6 base() [6/8] . . . 1098
12.40.2.7 base() [7/8] . . . 1099
12.40.2.8 base() [8/8] . . . 1099
12.40.2.9 in() [1/2] . . . 1099
12.40.2.10 in() [2/2] . . . 1100
12.41 trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input_port Struct Reference . . . 1100
12.41.1 Detailed Description . . . 1101
12.41.2 Member Data Documentation . . . 1101
12.41.2.1 enabled . . . 1101
12.41.2.2 stream . . . 1101
12.42 trisycl::detail::is_range< T > Struct Template Reference . . . 1101
12.42.1 Detailed Description . . . 1102
12.43 trisycl::detail::is_range< range< Dimensions > > Struct Template Reference . . . 1103
12.43.1 Detailed Description . . . 1103
12.44 trisycl::is_wrapper< T > Struct Template Reference . . . 1104
12.44.1 Detailed Description . . . 1104
12.45 Layout Class Reference . . . 1105
12.46 trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device Struct Reference . . . 1105
12.46.1 Detailed Description . . . 1106
12.46.2 Member Typedef Documentation . . . 1106
12.46.2.1 value_t [1/2] . . . 1107
12.46.2.2 value_t [2/2] . . . 1107
12.46.3 Member Function Documentation . . . 1107
12.46.3.1 acquire() [1/2] . . . 1107
12.46.3.2 acquire() [2/2] . . . 1107
12.46.3.3 acquire_with_value() [1/2] . . . 1108
12.46.3.4 acquire_with_value() [2/2] . . . 1108
12.46.3.5 release() [1/2] . . . 1108
12.46.3.6 release() [2/2] . . . 1108
12.46.3.7 release_with_value() [1/2] . . . 1109
12.46.3.8 release_with_value() [2/2] . . . 1109
12.46.4 Member Data Documentation . . . 1109
12.46.4.1 cv . . . 1109
12.46.4.2 m . . . 1110
12.46.4.3 value . . . 1110
12.47 trisycl::vendor::xilinx::partition::none Struct Reference . . . 1110
12.47.1 Detailed Description . . . 1111

12.47.2 Member Data Documentation . . . 1111
12.47.2.1 partition_type . . . 1111
12.48 trisycl::detail::opencl_context Class Reference . . . 1111
12.48.1 Detailed Description . . . 1113
12.48.2 Constructor & Destructor Documentation . . . 1113
12.48.2.1 opencl_context() . . . 1113
12.48.2.2 ~opencl_context() . . . 1113
12.48.3 Member Function Documentation . . . 1114
12.48.3.1 get() . . . 1114
12.48.3.2 get_boost_compute() . . . 1114
12.48.3.3 get_boost_queue() . . . 1114
12.48.3.4 get_devices() . . . 1115
12.48.3.5 get_platform() . . . 1115
12.48.3.6 get_reference_count() . . . 1115
12.48.3.7 instance() . . . 1116
12.48.3.8 is_host() . . . 1116
12.48.4 Member Data Documentation . . . 1117
12.48.4.1 c . . . 1117
12.48.4.2 cache . . . 1117
12.48.4.3 q . . . 1117
12.49 trisycl::detail::opencl_device Class Reference . . . 1118
12.49.1 Detailed Description . . . 1119
12.49.2 Constructor & Destructor Documentation . . . 1119
12.49.2.1 opencl_device() . . . 1119
12.49.2.2 ~opencl_device() . . . 1120
12.49.3 Member Function Documentation . . . 1120
12.49.3.1 get() . . . 1120
12.49.3.2 get_boost_compute() . . . 1120
12.49.3.3 get_info() [1/2] . . . 1121
12.49.3.4 get_info() [2/2] . . . 1121
12.49.3.5 get_platform() . . . 1122
12.49.3.6 has_extension() . . . 1123
12.49.3.7 instance() . . . 1123
12.49.3.8 is_accelerator() . . . 1124
12.49.3.9 is_cpu() . . . 1124
12.49.3.10 is_gpu() . . . 1124
12.49.3.11 is_host() . . . 1125
12.49.4 Member Data Documentation . . . 1125
12.49.4.1 cache . . . 1125
12.49.4.2 d . . . 1125
12.50 trisycl::detail::opencl_event Class Reference . . . 1126
12.50.1 Detailed Description . . . 1127

12.50.2 Constructor & Destructor Documentation . . . 1127
12.50.2.1 opencl_event() . . . 1127
12.50.2.2 ~opencl_event() . . . 1128
12.50.3 Member Function Documentation . . . 1128
12.50.3.1 get() . . . 1128
12.50.3.2 get_boost_compute() . . . 1128
12.50.3.3 get_command_execution_status() . . . 1129
12.50.3.4 get_profiling_info() . . . 1129
12.50.3.5 get_reference_count() . . . 1129
12.50.3.6 instance() . . . 1130
12.50.3.7 is_host() . . . 1130
12.50.3.8 wait() . . . 1130
12.50.4 Member Data Documentation . . . 1131
12.50.4.1 cache . . . 1131
12.50.4.2 e . . . 1131
12.51 trisycl::detail::opencl_kernel Class Reference . . . 1131
12.51.1 Detailed Description . . . 1133
12.51.2 Constructor & Destructor Documentation . . . 1133
12.51.2.1 opencl_kernel() . . . 1133
12.51.3 Member Function Documentation . . . 1133
12.51.3.1 get() . . . 1133
12.51.3.2 get_boost_compute() . . . 1134
12.51.3.3 instance() . . . 1134
12.51.3.4 single_task() . . . 1134
12.51.3.5 TRISYCL_ParallelForKernel_RANGE() . . . 1135
12.51.4 Member Data Documentation . . . 1135
12.51.4.1 cache . . . 1135
12.51.4.2 k . . . 1135
12.52 trisycl::detail::opencl_queue Class Reference . . . 1136
12.52.1 Detailed Description . . . 1137
12.52.2 Constructor & Destructor Documentation . . . 1137
12.52.2.1 opencl_queue() . . . 1138
12.52.2.2 ~opencl_queue() . . . 1138
12.52.3 Member Function Documentation . . . 1138
12.52.3.1 get() . . . 1138
12.52.3.2 get_boost_compute() . . . 1139
12.52.3.3 get_context() . . . 1139
12.52.3.4 get_device() . . . 1139
12.52.3.5 instance() [1/2] . . . 1140
12.52.3.6 instance() [2/2] . . . 1140
12.52.3.7 is_host() . . . 1141
12.52.4 Member Data Documentation . . . 1141

12.52.4.1 cache . . . 1141
12.52.4.2 q . . . 1141
12.53 trisycl::vendor::xilinx::acap::aie::connection::output Struct Reference . . . 1142
12.53.1 Detailed Description . . . 1143
12.53.2 Member Function Documentation . . . 1143
12.53.2.1 base() [1/8] . . . 1143
12.53.2.2 base() [2/8] . . . 1144
12.53.2.3 base() [3/8] . . . 1144
12.53.2.4 base() [4/8] . . . 1144
12.53.2.5 base() [5/8] . . . 1144
12.53.2.6 base() [6/8] . . . 1144
12.53.2.7 base() [7/8] . . . 1145
12.53.2.8 base() [8/8] . . . 1145
12.53.2.9 out() [1/2] . . . 1145
12.53.2.10 out() [2/2] . . . 1146
12.54 trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::output_port Struct Reference . . . 1146
12.54.1 Detailed Description . . . 1147
12.54.2 Member Data Documentation . . . 1147
12.54.2.1 stream . . . 1147
12.55 trisycl::vendor::xilinx::graphics::palette Struct Reference . . . 1147
12.55.1 Detailed Description . . . 1149
12.55.2 Member Enumeration Documentation . . . 1149
12.55.2.1 kind [1/2] . . . 1149
12.55.2.2 kind [2/2] . . . 1149
12.55.3 Constructor & Destructor Documentation . . . 1150
12.55.3.1 palette() [1/2] . . . 1150
12.55.3.2 palette() [2/2] . . . 1150
12.55.4 Member Function Documentation . . . 1151
12.55.4.1 decrease_clip() [1/2] . . . 1151
12.55.4.2 decrease_clip() [2/2] . . . 1152
12.55.4.3 decrease_frequency() [1/2] . . . 1152
12.55.4.4 decrease_frequency() [2/2] . . . 1153
12.55.4.5 decrease_phase() [1/2] . . . 1154
12.55.4.6 decrease_phase() [2/2] . . . 1154
12.55.4.7 gray_palette() [1/2] . . . 1155
12.55.4.8 gray_palette() [2/2] . . . 1156
12.55.4.9 handle_key_press_event() [1/2] . . . 1156
12.55.4.10 handle_key_press_event() [2/2] . . . 1158
12.55.4.11 increase_clip() [1/2] . . . 1159
12.55.4.12 increase_clip() [2/2] . . . 1160
12.55.4.13 increase_frequency() [1/2] . . . 1160

12.55.4.14 increase_frequency() [2/2] . . . 1161
12.55.4.15 increase_phase() [1/2] . . . 1161
12.55.4.16 increase_phase() [2/2] . . . 1162
12.55.4.17 palettize() [1/2] . . . 1163
12.55.4.18 palettize() [2/2] . . . 1163
12.55.4.19 rainbow_palette() [1/2] . . . 1164
12.55.4.20 rainbow_palette() [2/2] . . . 1164
12.55.4.21 set() [1/2] . . . 1165
12.55.4.22 set() [2/2] . . . 1166
12.55.4.23 update() [1/2] . . . 1166
12.55.4.24 update() [2/2] . . . 1168
12.55.5 Member Data Documentation . . . 1168
12.55.5.1 clip . . . 1168
12.55.5.2 color_mapping . . . 1169
12.55.5.3 frequency_log2 . . . 1169
12.55.5.4 k . . . 1169
12.55.5.5 phase . . . 1169
12.55.5.6 size . . . 1169
12.56 trisycl::info::param_traits< T, Param > Struct Template Reference . . . 1170
12.56.1 Detailed Description . . . 1170
12.57 trisycl::detail::pooled_shared_work::pool_ctx Struct Reference . . . 1170
12.57.1 Detailed Description . . . 1171
12.57.2 Constructor & Destructor Documentation . . . 1171
12.57.2.1 pool_ctx() . . . 1171
12.57.3 Member Data Documentation . . . 1171
12.57.3.1 rqueue_ . . . 1171
12.57.3.2 rqueue_mtx_ . . . 1171
12.57.3.3 suspend_ . . . 1172
12.58 trisycl::detail::pooled_work_stealing::pool_ctx Struct Reference . . . 1172
12.58.1 Detailed Description . . . 1173
12.58.2 Constructor & Destructor Documentation . . . 1173
12.58.2.1 pool_ctx() . . . 1173
12.58.3 Member Data Documentation . . . 1173
12.58.3.1 barrier_ . . . 1173
12.58.3.2 counter_ . . . 1173
12.58.3.3 schedulers_ . . . 1174
12.58.3.4 suspend_ . . . 1174
12.58.3.5 thread_count_ . . . 1174
12.59 trisycl::detail::pooled_shared_work Class Reference . . . 1174
12.59.1 Detailed Description . . . 1176
12.59.2 Member Typedef Documentation . . . 1176
12.59.2.1 ctx . . . 1176

12.59.2.2 lqueue_type . . . 1176
12.59.2.3 rqueue_type . . . 1176
12.59.3 Constructor & Destructor Documentation . . . 1177
12.59.3.1 pooled_shared_work() [1/3] . . . 1177
12.59.3.2 pooled_shared_work() [2/3] . . . 1177
12.59.3.3 pooled_shared_work() [3/3] . . . 1177
12.59.4 Member Function Documentation . . . 1177
12.59.4.1 awakened() . . . 1177
12.59.4.2 create_pool_ctx() . . . 1178
12.59.4.3 has_ready_fibers() . . . 1178
12.59.4.4 notify() . . . 1178
12.59.4.5 operator=() [1/2] . . . 1179
12.59.4.6 operator=() [2/2] . . . 1179
12.59.4.7 pick_next() . . . 1179
12.59.4.8 suspend_until() . . . 1179
12.59.5 Member Data Documentation . . . 1180
12.59.5.1 cnd_ . . . 1180
12.59.5.2 flag_ . . . 1180
12.59.5.3 lqueue_ . . . 1180
12.59.5.4 mtx_ . . . 1181
12.59.5.5 pool_ctx_ . . . 1181
12.60 trisycl::detail::pooled_work_stealing Class Reference . . . 1181
12.60.1 Detailed Description . . . 1183
12.60.2 Member Typedef Documentation . . . 1183
12.60.2.1 ctx . . . 1183
12.60.3 Constructor & Destructor Documentation . . . 1183
12.60.3.1 pooled_work_stealing() [1/3] . . . 1183
12.60.3.2 ∼pooled_work_stealing() . . . 1184
12.60.3.3 pooled_work_stealing() [2/3] . . . 1184
12.60.3.4 pooled_work_stealing() [3/3] . . . 1184
12.60.4 Member Function Documentation . . . 1184
12.60.4.1 awakened() . . . 1184
12.60.4.2 create_pool_ctx() . . . 1185
12.60.4.3 has_ready_fibers() . . . 1185
12.60.4.4 notify() . . . 1185
12.60.4.5 operator=() [1/2] . . . 1186
12.60.4.6 operator=() [2/2] . . . 1186
12.60.4.7 pick_next() . . . 1186
12.60.4.8 steal() . . . 1187
12.60.4.9 suspend_until() . . . 1187
12.60.5 Member Data Documentation . . . 1187
12.60.5.1 cnd_ . . . 1188

12.60.5.2 flag_ . . . 1188
12.60.5.3 id_ . . . 1188
12.60.5.4 mtx_ . . . 1188
12.60.5.5 pool_ctx_ . . . 1188
12.60.5.6 rqueue_ . . . 1189
12.61 trisycl::vendor::xilinx::acap::aie::port Struct Reference . . . 1189
12.61.1 Detailed Description . . . 1189
12.62 trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich > Class Template Reference . . . 1190
12.62.1 Detailed Description . . . 1191
12.62.2 Member Typedef Documentation . . . 1192
12.62.2.1 value_type [1/2] . . . 1192
12.62.2.2 value_type [2/2] . . . 1192
12.62.3 Constructor & Destructor Documentation . . . 1192
12.62.3.1 port_receiver() [1/2] . . . 1192
12.62.3.2 port_receiver() [2/2] . . . 1192
12.62.4 Member Function Documentation . . . 1193
12.62.4.1 read() [1/2] . . . 1193
12.62.4.2 read() [2/2] . . . 1193
12.62.4.3 try_read() [1/2] . . . 1194
12.62.4.4 try_read() [2/2] . . . 1194
12.62.4.5 try_write() [1/2] . . . 1195
12.62.4.6 try_write() [2/2] . . . 1195
12.62.4.7 write() [1/2] . . . 1196
12.62.4.8 write() [2/2] . . . 1196
12.62.5 Member Data Documentation . . . 1196
12.62.5.1 axi_ss . . . 1197
12.62.5.2 c . . . 1197
12.62.5.3 capacity . . . 1197
12.62.5.4 component_name . . . 1197
12.63 trisycl::detail::program Class Reference . . . 1198
12.63.1 Detailed Description . . . 1198
12.63.2 Constructor & Destructor Documentation . . . 1199
12.63.2.1 ~program() . . . 1199
12.63.3 Member Function Documentation . . . 1199
12.63.3.1 get() . . . 1199
12.64 trisycl::drt::code::program Struct Reference . . . 1199
12.64.1 Detailed Description . . . 1200
12.64.2 Constructor & Destructor Documentation . . . 1200
12.64.2.1 program() . . . 1200
12.64.3 Member Data Documentation . . . 1200
12.64.3.1 binary . . . 1201
12.64.3.2 binary_size . . . 1201

12.64.3.3 p . . . 1201
12.65 trisycl::program Class Reference . . . 1202
12.65.1 Detailed Description . . . 1203
12.65.2 Member Typedef Documentation . . . 1203
12.65.2.1 implementation_t . . . 1203
12.65.3 Constructor & Destructor Documentation . . . 1204
12.65.3.1 program() [1/2] . . . 1204
12.65.3.2 program() [2/2] . . . 1204
12.65.4 Member Function Documentation . . . 1204
12.65.4.1 build_with_kernel_type() . . . 1204
12.65.4.2 compile_with_kernel_type() . . . 1204
12.65.4.3 get() . . . 1205
12.65.4.4 get_kernel() [1/2] . . . 1205
12.65.4.5 get_kernel() [2/2] . . . 1205
12.65.4.6 has_kernel() [1/2] . . . 1205
12.65.4.7 has_kernel() [2/2] . . . 1205
12.65.5 Friends And Related Function Documentation . . . 1205
12.65.5.1 handler . . . 1206
12.65.6 Member Data Documentation . . . 1206
12.65.6.1 implementation_t . . . 1206
12.66 trisycl::detail::property Class Reference . . . 1206
12.66.1 Detailed Description . . . 1206
12.67 trisycl::property_list Class Reference . . . 1207
12.67.1 Detailed Description . . . 1207
12.67.2 Constructor & Destructor Documentation . . . 1208
12.67.2.1 property_list() [1/2] . . . 1208
12.67.2.2 property_list() [2/2] . . . 1208
12.67.3 Member Function Documentation . . . 1208
12.67.3.1 addproperty() . . . 1209
12.67.3.2 get_property() . . . 1209
12.67.3.3 has_property() . . . 1210
12.67.3.4 TRISYCL_PROPERTY_CREATE() . . . 1210
12.68 trisycl::detail::queue Struct Reference . . . 1210
12.68.1 Detailed Description . . . 1212
12.68.2 Constructor & Destructor Documentation . . . 1212
12.68.2.1 queue() . . . 1212
12.68.2.2 ~queue() . . . 1212
12.68.3 Member Function Documentation . . . 1213
12.68.3.1 get() . . . 1213
12.68.3.2 get_boost_compute() . . . 1213
12.68.3.3 get_context() . . . 1213
12.68.3.4 get_device() . . . 1213

12.68.3.5 is_host() . . . 1214
12.68.3.6 kernel_end() . . . 1214
12.68.3.7 kernel_start() . . . 1214
12.68.3.8 wait_for_kernel_execution() . . . 1215
12.68.4 Member Data Documentation . . . 1215
12.68.4.1 finished . . . 1215
12.68.4.2 finished_mutex . . . 1215
12.68.4.3 running_kernels . . . 1216
12.69 trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter Struct Reference . . . 1216
12.69.1 Detailed Description . . . 1217
12.69.2 Member Function Documentation . . . 1217
12.69.2.1 wait() [1/2] . . . 1217
12.69.2.2 wait() [2/2] . . . 1218
12.70 trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch > Class Template Reference . . . 1218
12.70.1 Detailed Description . . . 1220
12.70.2 Member Typedef Documentation . . . 1220
12.70.2.1 dma_base [1/2] . . . 1220
12.70.2.2 dma_base [2/2] . . . 1220
12.70.3 Constructor & Destructor Documentation . . . 1220
12.70.3.1 receiving_dma() [1/2] . . . 1221
12.70.3.2 receiving_dma() [2/2] . . . 1221
12.70.4 Member Function Documentation . . . 1221
12.70.4.1 read() [1/2] . . . 1221
12.70.4.2 read() [2/2] . . . 1222
12.70.4.3 receive() [1/2] . . . 1222
12.70.4.4 receive() [2/2] . . . 1223
12.70.4.5 try_read() [1/2] . . . 1223
12.70.4.6 try_read() [2/2] . . . 1224
12.70.4.7 try_write() [1/2] . . . 1224
12.70.4.8 try_write() [2/2] . . . 1225
12.70.4.9 write() [1/2] . . . 1225
12.70.4.10 write() [2/2] . . . 1225
12.70.5 Member Data Documentation . . . 1226
12.70.5.1 axi_ss . . . 1226
12.70.5.2 fifo . . . 1226
12.71 trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion Class Reference . . . 1226
12.71.1 Detailed Description . . . 1229
12.71.2 Member Typedef Documentation . . . 1229
12.71.2.1 value_type [1/2] . . . 1229
12.71.2.2 value_type [2/2] . . . 1229
12.71.3 Constructor & Destructor Documentation . . . 1229

12.71.3.1 router_minion() [1/2] . . . 1230
12.71.3.2 ∼router_minion() [1/2] . . . 1231
12.71.3.3 router_minion() [2/2] . . . 1231
12.71.3.4 ∼router_minion() [2/2] . . . 1232
12.71.4 Member Function Documentation . . . 1233
12.71.4.1 connect_to() [1/2] . . . 1233
12.71.4.2 connect_to() [2/2] . . . 1233
12.71.4.3 get_master_port_dests() [1/2] . . . 1234
12.71.4.4 get_master_port_dests() [2/2] . . . 1234
12.71.4.5 join() [1/2] . . . 1234
12.71.4.6 join() [2/2] . . . 1235
12.71.4.7 read() [1/2] . . . 1235
12.71.4.8 read() [2/2] . . . 1235
12.71.4.9 try_read() [1/2] . . . 1236
12.71.4.10 try_read() [2/2] . . . 1236
12.71.4.11 try_write() [1/2] . . . 1237
12.71.4.12 try_write() [2/2] . . . 1237
12.71.4.13 write() [1/2] . . . 1238
12.71.4.14 write() [2/2] . . . 1238
12.71.5 Member Data Documentation . . . 1239
12.71.5.1 axi_ss . . . 1239
12.71.5.2 c . . . 1239
12.71.5.3 capacity . . . 1239
12.71.5.4 futures . . . 1240
12.71.5.5 mpl_outputs . . . 1240
12.71.5.6 outputs . . . 1240
12.72 trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch > Class Template Reference . . . 1241
12.72.1 Detailed Description . . . 1242
12.72.2 Member Typedef Documentation . . . 1242
12.72.2.1 dma_base [1/2] . . . 1242
12.72.2.2 dma_base [2/2] . . . 1242
12.72.3 Constructor & Destructor Documentation . . . 1242
12.72.3.1 sending_dma() [1/2] . . . 1243
12.72.3.2 sending_dma() [2/2] . . . 1243
12.72.4 Member Function Documentation . . . 1243
12.72.4.1 send() [1/2] . . . 1243
12.72.4.2 send() [2/2] . . . 1244
12.72.5 Member Data Documentation . . . 1244
12.72.5.1 output_port . . . 1244
12.73 trisycl::detail::shared_ptr_implementation< Parent, Implementation > Struct Template Reference . . . 1245
12.73.1 Detailed Description . . . 1246
12.73.2 Constructor & Destructor Documentation . . . 1246

12.73.2.1 shared_ptr_implementation() [1/3] . . . 1246
12.73.2.2 shared_ptr_implementation() [2/3] . . . 1247
12.73.2.3 shared_ptr_implementation() [3/3] . . . 1247
12.73.3 Member Function Documentation . . . 1247
12.73.3.1 hash() . . . 1247
12.73.4 Friends And Related Function Documentation . . . 1248
12.73.4.1 operator< . . . 1248
12.73.4.2 operator== . . . 1249
12.73.5 Member Data Documentation . . . 1249
12.73.5.1 implementation . . . 1249
12.74 trisycl::vendor::xilinx::acap::aie::port::shim Struct Reference . . . 1250
12.74.1 Detailed Description . . . 1250
12.74.2 Member Data Documentation . . . 1250
12.74.2.1 port . . . 1250
12.74.2.2 x . . . 1251
12.75 trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch Struct Reference 1251
12.75.1 Detailed Description . . . 1254
12.75.2 Member Enumeration Documentation . . . 1254
12.75.2.1 master_port_layout [1/2] . . . 1254
12.75.2.2 master_port_layout [2/2] . . . 1256
12.75.2.3 slave_port_layout [1/2] . . . 1258
12.75.2.4 slave_port_layout [2/2] . . . 1260
12.75.3 Friends And Related Function Documentation . . . 1262
12.75.3.1 is_axi_master [1/4] . . . 1262
12.75.3.2 is_axi_master [2/4] . . . 1262
12.75.3.3 is_axi_master [3/4] . . . 1262
12.75.3.4 is_axi_master [4/4] . . . 1263
12.75.4 Member Data Documentation . . . 1263
12.75.4.1 fifo_depth . . . 1263
12.75.4.2 latency . . . 1263
12.75.4.3 m_east_range . . . 1264
12.75.4.4 m_east_size . . . 1264
12.75.4.5 m_fifo_range . . . 1264
12.75.4.6 m_fifo_size . . . 1265
12.75.4.7 m_north_range . . . 1265
12.75.4.8 m_north_size . . . 1265
12.75.4.9 m_south_range . . . 1266
12.75.4.10 m_south_size . . . 1266
12.75.4.11 m_tile_ctrl_range . . . 1266
12.75.4.12 m_tile_ctrl_size . . . 1267
12.75.4.13 m_west_range . . . 1267
12.75.4.14 m_west_size . . . 1267

12.75.4.15 nb_master_port . . . 1268
12.75.4.16 nb_slave_port . . . 1268
12.75.4.17 s_east_range . . . 1268
12.75.4.18 s_east_size . . . 1269
12.75.4.19 s_fifo_range . . . 1269
12.75.4.20 s_fifo_size . . . 1269
12.75.4.21 s_north_range . . . 1270
12.75.4.22 s_north_size . . . 1270
12.75.4.23 s_south_range . . . 1270
12.75.4.24 s_south_size . . . 1271
12.75.4.25 s_tile_ctrl_range . . . 1271
12.75.4.26 s_tile_ctrl_size . . . 1271
12.75.4.27 s_west_range . . . 1272
12.75.4.28 s_west_size . . . 1272
12.76 trisycl::vendor::xilinx::acap::aie::axi_packet::shutdown_t Struct Reference . . . 1272
12.76.1 Detailed Description . . . 1272
12.77 trisycl::detail::singleton< T > Struct Template Reference . . . 1273
12.77.1 Detailed Description . . . 1273
12.77.2 Member Function Documentation . . . 1273
12.77.2.1 instance() . . . 1273
12.78 trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, StorageType > Struct Template Reference . . . 1274
12.78.1 Detailed Description . . . 1274
12.78.2 Member Typedef Documentation . . . 1274
12.78.2.1 type [1/2] . . . 1274
12.78.2.2 type [2/2] . . . 1274
12.79 trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, typename ScopedType::storage_type > Struct Template Reference . . . 1275
12.79.1 Detailed Description . . . 1275
12.79.2 Member Typedef Documentation . . . 1275
12.79.2.1 type [1/2] . . . 1275
12.79.2.2 type [2/2] . . . 1275
12.80 trisycl::detail::task Struct Reference . . . 1276
12.80.1 Detailed Description . . . 1279
12.80.2 Constructor & Destructor Documentation . . . 1279
12.80.2.1 task() . . . 1279
12.80.3 Member Function Documentation . . . 1279
12.80.3.1 add_buffer() . . . 1279
12.80.3.2 add_postlude() . . . 1280
12.80.3.3 add_prelude() . . . 1280
12.80.3.4 get_compute_buffer() . . . 1281
12.80.3.5 get_kernel() . . . 1281
12.80.3.6 get_queue() . . . 1282

12.80.3.7 notify_consumers() . . . 1282
12.80.3.8 postlude() . . . 1283
12.80.3.9 prelude() . . . 1284
12.80.3.10 register_accessor() . . . 1284
12.80.3.11 release_buffers() . . . 1285
12.80.3.12 schedule() . . . 1286
12.80.3.13 set_arg() [1/2] . . . 1287
12.80.3.14 set_arg() [2/2] . . . 1288
12.80.3.15 set_kernel() . . . 1289
12.80.3.16 wait() . . . 1289
12.80.3.17 wait_for_producers() . . . 1290
12.80.4 Member Data Documentation . . . 1290
12.80.4.1 accessors . . . 1290
12.80.4.2 buffers_in_use . . . 1291
12.80.4.3 epilogues . . . 1291
12.80.4.4 execution_ended . . . 1291
12.80.4.5 kernel . . . 1291
12.80.4.6 owner_queue . . . 1292
12.80.4.7 producer_tasks . . . 1292
12.80.4.8 prologues . . . 1292
12.80.4.9 ready . . . 1292
12.80.4.10 ready_mutex . . . 1293
12.81 trisycl::vendor::xilinx::acap::aie::port::tile Struct Reference . . . 1293
12.81.1 Detailed Description . . . 1293
12.81.2 Member Data Documentation . . . 1294
12.81.2.1 port . . . 1294
12.81.2.2 x . . . 1294
12.81.2.3 y . . . 1294
12.82 trisycl::mixin::accessor< T, Dimensions >::track_index< N > Struct Template Reference . . . 1294
12.82.1 Detailed Description . . . 1295
12.82.2 Constructor & Destructor Documentation . . . 1295
12.82.2.1 track_index() [1/2] . . . 1295
12.82.2.2 track_index() [2/2] . . . 1296
12.82.3 Member Function Documentation . . . 1296
12.82.3.1 operator[]() . . . 1296
12.82.4 Member Data Documentation . . . 1297
12.82.4.1 indices . . . 1297
12.82.4.2 mds . . . 1297
12.83 std::tuple_element< I, T > Struct Template Reference . . . 1297
12.83.1 Detailed Description . . . 1298
12.83.2 Member Typedef Documentation . . . 1298
12.83.2.1 type . . . 1298

12.84 std::tuple_element< I, trisycl::vec< T, Size > > Struct Template Reference . . . 1298
12.84.1 Detailed Description . . . 1298
12.84.2 Member Typedef Documentation . . . 1299
12.84.2.1 type . . . 1299
12.85 std::tuple_size< T > Struct Template Reference . . . 1299
12.85.1 Detailed Description . . . 1300
12.86 std::tuple_size< trisycl::vec< T, Size > > Struct Template Reference . . . 1300
12.86.1 Detailed Description . . . 1301
12.87 trisycl::vendor::xilinx::acap::aie::layout::vc1902 Struct Reference . . . 1301
12.87.1 Detailed Description . . . 1302
12.88 trisycl::detail::vec< DataType, NumElements > Class Template Reference . . . 1302
12.88.1 Detailed Description . . . 1305
12.88.2 Member Typedef Documentation . . . 1305
12.88.2.1 basic_type . . . 1305
12.88.3 Constructor & Destructor Documentation . . . 1305
12.88.3.1 vec() [1/2] . . . 1305
12.88.3.2 vec() [2/2] . . . 1306
12.88.4 Member Function Documentation . . . 1306
12.88.4.1 apply_binary_functor() . . . 1306
12.88.4.2 apply_binary_functor_impl() . . . 1307
12.88.4.3 apply_ternary_functor_impl() . . . 1307
12.88.4.4 apply_unary_functor() . . . 1308
12.88.4.5 apply_unary_functor_impl() . . . 1308
12.88.4.6 as() . . . 1309
12.88.4.7 convert() . . . 1309
12.88.4.8 flatten() [1/3] . . . 1310
12.88.4.9 flatten() [2/3] . . . 1310
12.88.4.10 flatten() [3/3] . . . 1311
12.88.4.11 flatten_to_tuple() . . . 1311
12.88.4.12 get_count() . . . 1312
12.88.4.13 get_size() . . . 1312
12.88.4.14 map() . . . 1313
12.88.4.15 swizzle() [1/2] . . . 1313
12.88.4.16 swizzle() [2/2] . . . 1314
12.88.4.17 zip() [1/2] . . . 1314
12.88.4.18 zip() [2/2] . . . 1315
12.89 trisycl::vec< typename, int > Class Template Reference . . . 1315
12.89.1 Detailed Description . . . 1315
12.90 trisycl::vendor::trisycl::random::xorshift< Nbits > Struct Template Reference . . . 1316
12.90.1 Detailed Description . . . 1317
12.90.2 Member Typedef Documentation . . . 1317
12.90.2.1 result_type [1/2] . . . 1317

12.90.2.2 result_type [2/2] . . . 1318
12.90.2.3 value_type [1/2] . . . 1318
12.90.2.4 value_type [2/2] . . . 1318
12.90.3 Constructor & Destructor Documentation . . . 1318
12.90.3.1 xorshift() [1/4] . . . 1318
12.90.3.2 xorshift() [2/4] . . . 1319
12.90.3.3 xorshift() [3/4] . . . 1319
12.90.3.4 xorshift() [4/4] . . . 1319
12.90.4 Member Function Documentation . . . 1319
12.90.4.1 max() [1/2] . . . 1319
12.90.4.2 max() [2/2] . . . 1320
12.90.4.3 min() [1/2] . . . 1320
12.90.4.4 min() [2/2] . . . 1321
12.90.4.5 operator()() [1/2] . . . 1321
12.90.4.6 operator()() [2/2] . . . 1322
12.90.5 Member Data Documentation . . . 1322
12.90.5.1 bit_size . . . 1322
12.90.5.2 initial_state . . . 1323
12.90.5.3 state . . . 1323

13 File Documentation **1325**

13.1 include/CL/sycl.hpp File Reference . . . 1325
13.1.1 Detailed Description . . . 1325
13.1.2 Macro Definition Documentation . . . 1325
13.1.2.1 TRISYCL_SYCL_NAMESPACE . . . 1326
13.2 sycl.hpp . . . 1326
13.3 include/sycl/sycl.hpp File Reference . . . 1326
13.3.1 Detailed Description . . . 1327
13.3.2 Macro Definition Documentation . . . 1327
13.3.2.1 TRISYCL_SYCL_NAMESPACE . . . 1327
13.4 sycl.hpp . . . 1327
13.5 include/triSYCL/sycl.hpp File Reference . . . 1328
13.6 sycl.hpp . . . 1329
13.7 include/triSYCL/access.hpp File Reference . . . 1331
13.8 access.hpp . . . 1331
13.9 include/triSYCL/accessor/detail/accessor_base.hpp File Reference . . . 1332
13.10 accessor_base.hpp . . . 1334
13.11 include/triSYCL/accessor/detail/local_accessor.hpp File Reference . . . 1335
13.12 local_accessor.hpp . . . 1336
13.13 include/triSYCL/address_space.hpp File Reference . . . 1337
13.13.1 Detailed Description . . . 1339
13.14 address_space.hpp . . . 1339

13.15 include/triSYCL/address_space/detail/address_space.hpp File Reference . . . 1341
13.15.1 Detailed Description . . . 1343
13.15.2 Macro Definition Documentation . . . 1343
13.15.2.1 TRISYCL_CONSTANT_AS . . . 1343
13.15.2.2 TRISYCL_GENERIC_AS . . . 1343
13.15.2.3 TRISYCL_GLOBAL_AS . . . 1343
13.15.2.4 TRISYCL_LOCAL_AS . . . 1343
13.15.2.5 TRISYCL_PRIVATE_AS . . . 1344
13.16 address_space.hpp . . . 1344
13.17 include/triSYCL/allocator.hpp File Reference . . . 1349
13.18 allocator.hpp . . . 1350
13.19 include/triSYCL/buffer.hpp File Reference . . . 1351
13.20 buffer.hpp . . . 1352
13.21 include/triSYCL/buffer/detail/buffer.hpp File Reference . . . 1359
13.22 buffer.hpp . . . 1361
13.23 include/triSYCL/buffer/detail/buffer_base.hpp File Reference . . . 1365
13.24 buffer_base.hpp . . . 1366
13.25 include/triSYCL/buffer/detail/buffer_waiter.hpp File Reference . . . 1370
13.26 buffer_waiter.hpp . . . 1372
13.27 include/triSYCL/buffer_allocator.hpp File Reference . . . 1373
13.28 buffer_allocator.hpp . . . 1374
13.29 include/triSYCL/command_group/detail/task.hpp File Reference . . . 1375
13.30 task.hpp . . . 1376
13.31 include/triSYCL/context.hpp File Reference . . . 1380
13.32 context.hpp . . . 1382
13.33 include/triSYCL/context/detail/context.hpp File Reference . . . 1385
13.34 context.hpp . . . 1386
13.35 include/triSYCL/info/context.hpp File Reference . . . 1387
13.36 context.hpp . . . 1388
13.37 include/triSYCL/context/detail/context_tail.hpp File Reference . . . 1389
13.38 context_tail.hpp . . . 1389
13.39 include/triSYCL/context/detail/host_context.hpp File Reference . . . 1390
13.40 host_context.hpp . . . 1391
13.41 include/triSYCL/context/detail/opencl_context.hpp File Reference . . . 1392
13.42 opencl_context.hpp . . . 1394
13.43 include/triSYCL/detail/alignment_helper.hpp File Reference . . . 1395
13.43.1 Detailed Description . . . 1397
13.44 alignment_helper.hpp . . . 1397
13.45 include/triSYCL/detail/all_true.hpp File Reference . . . 1398
13.46 all_true.hpp . . . 1398
13.47 include/triSYCL/detail/array_tuple_helpers.hpp File Reference . . . 1399
13.47.1 Detailed Description . . . 1400

13.48 array_tuple_helpers.hpp . . . 1400
13.49 include/triSYCL/detail/cache.hpp File Reference . . . 1402
13.50 cache.hpp . . . 1403
13.51 include/triSYCL/detail/container_element_aspect.hpp File Reference . . . 1405
13.52 container_element_aspect.hpp . . . 1405
13.53 include/triSYCL/detail/default_classes.hpp File Reference . . . 1406
13.54 default_classes.hpp . . . 1407
13.55 include/triSYCL/detail/enum.hpp File Reference . . . 1409
13.55.1 Detailed Description . . . 1410
13.56 enum.hpp . . . 1411
13.57 include/triSYCL/detail/fiber/pooled_shared_work.hpp File Reference . . . 1411
13.58 pooled_shared_work.hpp . . . 1413
13.59 include/triSYCL/detail/fiber/pooled_work_stealing.hpp File Reference . . . 1415
13.60 pooled_work_stealing.hpp . . . 1416
13.61 include/triSYCL/detail/fiber_pool.hpp File Reference . . . 1419
13.61.1 Detailed Description . . . 1420
13.62 fiber_pool.hpp . . . 1421
13.63 include/triSYCL/detail/global_config.hpp File Reference . . . 1423
13.63.1 Macro Definition Documentation . . . 1424
13.63.1.1 TRISYCL_SYCL_NAMESPACE . . . 1425
13.63.1.2 TRISYCL_WEAK_ATTRIB_PREFIX . . . 1425
13.63.1.3 TRISYCL_WEAK_ATTRIB_SUFFIX . . . 1425
13.64 global_config.hpp . . . 1425
13.65 include/triSYCL/detail/instantiate_kernel.hpp File Reference . . . 1427
13.66 instantiate_kernel.hpp . . . 1428
13.67 include/triSYCL/detail/linear_id.hpp File Reference . . . 1430
13.68 linear_id.hpp . . . 1431
13.69 include/triSYCL/detail/metaprogramming.hpp File Reference . . . 1431
13.70 metaprogramming.hpp . . . 1433
13.71 include/triSYCL/detail/overloaded.hpp File Reference . . . 1433
13.72 overloaded.hpp . . . 1434
13.73 include/triSYCL/detail/property.hpp File Reference . . . 1434
13.74 property.hpp . . . 1435
13.75 include/triSYCL/detail/ranges.hpp File Reference . . . 1435
13.75.1 Detailed Description . . . 1436
13.76 ranges.hpp . . . 1437
13.77 include/triSYCL/detail/shared_ptr_implementation.hpp File Reference . . . 1438
13.78 shared_ptr_implementation.hpp . . . 1439
13.79 include/triSYCL/detail/singleton.hpp File Reference . . . 1440
13.80 singleton.hpp . . . 1441
13.81 include/triSYCL/detail/small_array.hpp File Reference . . . 1442
13.82 small_array.hpp . . . 1444

13.83 include/triSYCL/detail/SPIR/opencl_spir.h File Reference . . . 1449
13.83.1 Macro Definition Documentation . . . 1454
13.83.1.1 __kernel_exec . . . 1454
13.83.1.2 as_char . . . 1454
13.83.1.3 as_char16 . . . 1455
13.83.1.4 as_char2 . . . 1455
13.83.1.5 as_char3 . . . 1455
13.83.1.6 as_char4 . . . 1455
13.83.1.7 as_char8 . . . 1455
13.83.1.8 as_double . . . 1455
13.83.1.9 as_double16 . . . 1456
13.83.1.10 as_double2 . . . 1456
13.83.1.11 as_double3 . . . 1456
13.83.1.12 as_double4 . . . 1456
13.83.1.13 as_double8 . . . 1456
13.83.1.14 as_float . . . 1456
13.83.1.15 as_float16 . . . 1457
13.83.1.16 as_float2 . . . 1457
13.83.1.17 as_float3 . . . 1457
13.83.1.18 as_float4 . . . 1457
13.83.1.19 as_float8 . . . 1457
13.83.1.20 as_half . . . 1457
13.83.1.21 as_half16 . . . 1458
13.83.1.22 as_half2 . . . 1458
13.83.1.23 as_half3 . . . 1458
13.83.1.24 as_half4 . . . 1458
13.83.1.25 as_half8 . . . 1458
13.83.1.26 as_int . . . 1458
13.83.1.27 as_int16 . . . 1459
13.83.1.28 as_int2 . . . 1459
13.83.1.29 as_int3 . . . 1459
13.83.1.30 as_int4 . . . 1459
13.83.1.31 as_int8 . . . 1459
13.83.1.32 as_long . . . 1459
13.83.1.33 as_long16 . . . 1460
13.83.1.34 as_long2 . . . 1460
13.83.1.35 as_long3 . . . 1460
13.83.1.36 as_long4 . . . 1460
13.83.1.37 as_long8 . . . 1460
13.83.1.38 as_short . . . 1460
13.83.1.39 as_short16 . . . 1461
13.83.1.40 as_short2 . . . 1461

13.83.1.41 as_short3 . . . 1461
13.83.1.42 as_short4 . . . 1461
13.83.1.43 as_short8 . . . 1461
13.83.1.44 as_uchar . . . 1461
13.83.1.45 as_uchar16 . . . 1462
13.83.1.46 as_uchar2 . . . 1462
13.83.1.47 as_uchar3 . . . 1462
13.83.1.48 as_uchar4 . . . 1462
13.83.1.49 as_uchar8 . . . 1462
13.83.1.50 as_uint . . . 1462
13.83.1.51 as_uint16 . . . 1463
13.83.1.52 as_uint2 . . . 1463
13.83.1.53 as_uint3 . . . 1463
13.83.1.54 as_uint4 . . . 1463
13.83.1.55 as_uint8 . . . 1463
13.83.1.56 as_ulong . . . 1463
13.83.1.57 as_ulong16 . . . 1464
13.83.1.58 as_ulong2 . . . 1464
13.83.1.59 as_ulong3 . . . 1464
13.83.1.60 as_ulong4 . . . 1464
13.83.1.61 as_ulong8 . . . 1464
13.83.1.62 as_ushort . . . 1464
13.83.1.63 as_ushort16 . . . 1465
13.83.1.64 as_ushort2 . . . 1465
13.83.1.65 as_ushort3 . . . 1465
13.83.1.66 as_ushort4 . . . 1465
13.83.1.67 as_ushort8 . . . 1465
13.83.1.68 CHAR_BIT . . . 1465
13.83.1.69 CHAR_MAX . . . 1466
13.83.1.70 CHAR_MIN . . . 1466
13.83.1.71 CLK_A . . . 1466
13.83.1.72 CLK_ADDRESS_CLAMP . . . 1466
13.83.1.73 CLK_ADDRESS_CLAMP_TO_EDGE . . . 1466
13.83.1.74 CLK_ADDRESS_MIRRORED_REPEAT . . . 1466
13.83.1.75 CLK_ADDRESS_NONE . . . 1467
13.83.1.76 CLK_ADDRESS_REPEAT . . . 1467
13.83.1.77 CLK_ARGB . . . 1467
13.83.1.78 CLK_BGRA . . . 1467
13.83.1.79 CLK_DEPTH . . . 1467
13.83.1.80 CLK_DEPTH_STENCIL . . . 1468
13.83.1.81 CLK_FILTER_LINEAR . . . 1468
13.83.1.82 CLK_FILTER_NEAREST . . . 1468

13.83.1.83 CLK_FLOAT . . . 1468
13.83.1.84 CLK_GLOBAL_MEM_FENCE . . . 1468
13.83.1.85 CLK_HALF_FLOAT . . . 1468
13.83.1.86 CLK_INTENSITY . . . 1469
13.83.1.87 CLK_LOCAL_MEM_FENCE . . . 1469
13.83.1.88 CLK_LUMINANCE . . . 1469
13.83.1.89 CLK_NORMALIZED_COORDS_FALSE . . . 1469
13.83.1.90 CLK_NORMALIZED_COORDS_TRUE . . . 1469
13.83.1.91 CLK_R . . . 1469
13.83.1.92 CLK_RA . . . 1470
13.83.1.93 CLK_RG . . . 1470
13.83.1.94 CLK_RGB . . . 1470
13.83.1.95 CLK_RGBA . . . 1470
13.83.1.96 CLK_RGBx . . . 1470
13.83.1.97 CLK_RGx . . . 1470
13.83.1.98 CLK_Rx . . . 1471
13.83.1.99 CLK_SIGNED_INT16 . . . 1471
13.83.1.100 CLK_SIGNED_INT32 . . . 1471
13.83.1.101 CLK_SIGNED_INT8 . . . 1471
13.83.1.102 CLK_SNORM_INT16 . . . 1471
13.83.1.103 CLK_SNORM_INT8 . . . 1471
13.83.1.104 CLK_UNORM_INT16 . . . 1472
13.83.1.105 CLK_UNORM_INT24 . . . 1472
13.83.1.106 CLK_UNORM_INT8 . . . 1472
13.83.1.107 CLK_UNORM_INT_101010 . . . 1472
13.83.1.108 CLK_UNORM_SHORT_555 . . . 1472
13.83.1.109 CLK_UNORM_SHORT_565 . . . 1472
13.83.1.110 CLK_UNSIGNED_INT16 . . . 1473
13.83.1.111 CLK_UNSIGNED_INT32 . . . 1473
13.83.1.112 CLK_UNSIGNED_INT8 . . . 1473
13.83.1.113 const_func . . . 1473
13.83.1.114 DBL_DIG [1/2] . . . 1473
13.83.1.115 DBL_DIG [2/2] . . . 1474
13.83.1.116 DBL_EPSILON [1/2] . . . 1474
13.83.1.117 DBL_EPSILON [2/2] . . . 1474
13.83.1.118 DBL_MANT_DIG [1/2] . . . 1474
13.83.1.119 DBL_MANT_DIG [2/2] . . . 1474
13.83.1.120 DBL_MAX [1/2] . . . 1474
13.83.1.121 DBL_MAX [2/2] . . . 1475
13.83.1.122 DBL_MAX_10_EXP [1/2] . . . 1475
13.83.1.123 DBL_MAX_10_EXP [2/2] . . . 1475
13.83.1.124 DBL_MAX_EXP [1/2] . . . 1475

13.83.1.125 DBL_MAX_EXP [2/2] . . . 1475
13.83.1.126 DBL_MIN [1/2] . . . 1475
13.83.1.127 DBL_MIN [2/2] . . . 1476
13.83.1.128 DBL_MIN_10_EXP [1/2] . . . 1476
13.83.1.129 DBL_MIN_10_EXP [2/2] . . . 1476
13.83.1.130 DBL_MIN_EXP [1/2] . . . 1476
13.83.1.131 DBL_MIN_EXP [2/2] . . . 1476
13.83.1.132 DBL_RADIX . . . 1476
13.83.1.133 FLT_DIG . . . 1477
13.83.1.134 FLT_EPSILON . . . 1477
13.83.1.135 FLT_MANT_DIG . . . 1477
13.83.1.136 FLT_MAX . . . 1477
13.83.1.137 FLT_MAX_10_EXP . . . 1477
13.83.1.138 FLT_MAX_EXP . . . 1477
13.83.1.139 FLT_MIN . . . 1478
13.83.1.140 FLT_MIN_10_EXP . . . 1478
13.83.1.141 FLT_MIN_EXP . . . 1478
13.83.1.142 FLT_RADIX . . . 1478
13.83.1.143 FP_ILOGB0 . . . 1478
13.83.1.144 FP_ILOGBNAN . . . 1478
13.83.1.145 HUGE_VAL . . . 1479
13.83.1.146 HUGE_VALF . . . 1479
13.83.1.147 INFINITY . . . 1479
13.83.1.148 INT_MAX . . . 1479
13.83.1.149 INT_MIN . . . 1479
13.83.1.150 kernel_exec . . . 1480
13.83.1.151 LONG_MAX . . . 1480
13.83.1.152 LONG_MIN . . . 1480
13.83.1.153 M_1_PI . . . 1480
13.83.1.154 M_1_PI_F . . . 1480
13.83.1.155 M_2_PI . . . 1481
13.83.1.156 M_2_PI_F . . . 1481
13.83.1.157 M_2_SQRTPI . . . 1481
13.83.1.158 M_2_SQRTPI_F . . . 1481
13.83.1.159 M_E . . . 1481
13.83.1.160 M_E_F . . . 1481
13.83.1.161 M_LN10 . . . 1482
13.83.1.162 M_LN10_F . . . 1482
13.83.1.163 M_LN2 . . . 1482
13.83.1.164 M_LN2_F . . . 1482
13.83.1.165 M_LOG10E . . . 1482
13.83.1.166 M_LOG10E_F . . . 1482

13.83.1.167 M_LOG2E . . . 1483
13.83.1.168 M_LOG2E_F . . . 1483
13.83.1.169 M_PI . . . 1483
13.83.1.170 M_PI_2 . . . 1483
13.83.1.171 M_PI_2_F . . . 1483
13.83.1.172 M_PI_4 . . . 1483
13.83.1.173 M_PI_4_F . . . 1484
13.83.1.174 M_PI_F . . . 1484
13.83.1.175 M_SQRT1_2 . . . 1484
13.83.1.176 M_SQRT1_2_F . . . 1484
13.83.1.177 M_SQRT2 . . . 1484
13.83.1.178 M_SQRT2_F . . . 1484
13.83.1.179 MAXFLOAT . . . 1485
13.83.1.180 NAN . . . 1485
13.83.1.181 readonly . . . 1485
13.83.1.182 SCHAR_MAX . . . 1485
13.83.1.183 SCHAR_MIN . . . 1485
13.83.1.184 SHRT_MAX . . . 1485
13.83.1.185 SHRT_MIN . . . 1486
13.83.1.186 UCHAR_MAX . . . 1486
13.83.1.187 UINT_MAX . . . 1486
13.83.1.188 ULONG_MAX . . . 1486
13.83.1.189 USHRT_MAX . . . 1486
13.83.2 Typedef Documentation . . . 1486
13.83.2.1 __attribute__ . . . 1487
13.83.2.2 cl_mem_fence_flags . . . 1497
13.83.2.3 intptr_t . . . 1498
13.83.2.4 uchar . . . 1498
13.83.2.5 uint . . . 1498
13.83.2.6 uintptr_t . . . 1498
13.83.2.7 ulong . . . 1498
13.83.2.8 ushort . . . 1499
13.83.3 Function Documentation . . . 1499
13.83.3.1 __attribute__() . . . 1499
13.83.3.2 printf() . . . 1510
13.83.4 Variable Documentation . . . 1510
13.83.4.1 a . . . 1510
13.83.4.2 b . . . 1510
13.83.4.3 c . . . 1510
13.83.4.4 cmp . . . 1510
13.83.4.5 color . . . 1511
13.83.4.6 coord . . . 1511

13.83.4.7 cosval . . . 1511
13.83.4.8 dst_stride . . . 1511
13.83.4.9 edge1 . . . 1511
13.83.4.10 event . . . 1511
13.83.4.11 event_list . . . 1512
13.83.4.12 exp . . . 1512
13.83.4.13 i . . . 1512
13.83.4.14 iptr . . . 1512
13.83.4.15 lo . . . 1512
13.83.4.16 mask . . . 1513
13.83.4.17 maxval . . . 1513
13.83.4.18 minval . . . 1513
13.83.4.19 n . . . 1513
13.83.4.20 num_elements . . . 1513
13.83.4.21 offset . . . 1513
13.83.4.22 p . . . 1514
13.83.4.23 p1 . . . 1514
13.83.4.24 quo . . . 1514
13.83.4.25 sampler . . . 1514
13.83.4.26 signp . . . 1514
13.83.4.27 src . . . 1515
13.83.4.28 src_stride . . . 1515
13.83.4.29 val . . . 1515
13.83.4.30 x . . . 1515
13.83.4.31 y . . . 1516
13.83.4.32 z . . . 1516
13.84 opencl_spir.h . . . 1516
13.85 include/triSYCL/detail/SPIR/opencl_spir_helpers.hpp File Reference . . . 1670
13.85.1 Detailed Description . . . 1671
13.86 opencl_spir_helpers.hpp . . . 1672
13.87 include/triSYCL/detail/SPIR/opencl_spir_req.h File Reference . . . 1673
13.87.1 Macro Definition Documentation . . . 1674
13.87.1.1 __constant . . . 1674
13.87.1.2 __generic . . . 1674
13.87.1.3 __global . . . 1674
13.87.1.4 __local . . . 1675
13.87.1.5 __private . . . 1675
13.87.1.6 const_func . . . 1675
13.87.1.7 readonly . . . 1675
13.87.2 Function Documentation . . . 1675
13.87.2.1 __attribute__() . . . 1676
13.87.2.2 printf() . . . 1687

13.88 opencl_spir_req.h . . . 1687
13.89 include/triSYCL/detail/unimplemented.hpp File Reference . . . 1689
13.90 unimplemented.hpp . . . 1690
13.91 include/triSYCL/device/detail/device_tail.hpp File Reference . . . 1691
13.92 device_tail.hpp . . . 1691
13.93 include/triSYCL/device/detail/host_device.hpp File Reference . . . 1692
13.93.1 Macro Definition Documentation . . . 1693
13.93.1.1 TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE . . . 1694
13.94 host_device.hpp . . . 1694
13.95 include/triSYCL/device/detail/opencl_device.hpp File Reference . . . 1696
13.96 opencl_device.hpp . . . 1697
13.97 include/triSYCL/device/detail/opencl_device_tail.hpp File Reference . . . 1699
13.97.1 Macro Definition Documentation . . . 1699
13.97.1.1 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE . . . 1699
13.97.1.2 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE . . . 1700
13.98 opencl_device_tail.hpp . . . 1700
13.99 include/triSYCL/device_runtime.hpp File Reference . . . 1702
13.100 device_runtime.hpp . . . 1703
13.101 include/triSYCL/device_selector.hpp File Reference . . . 1706
13.102 device_selector.hpp . . . 1707
13.103 include/triSYCL/device_selector/detail/device_selector_tail.hpp File Reference . . . 1708
13.104 device_selector_tail.hpp . . . 1709
13.105 include/triSYCL/error_handler.hpp File Reference . . . 1711
13.106 error_handler.hpp . . . 1712
13.107 include/triSYCL/event.hpp File Reference . . . 1713
13.108 event.hpp . . . 1715
13.109 include/triSYCL/event/detail/event.hpp File Reference . . . 1717
13.110 event.hpp . . . 1717
13.111 include/triSYCL/info/event.hpp File Reference . . . 1718
13.112 event.hpp . . . 1719
13.113 include/triSYCL/event/detail/host_event.hpp File Reference . . . 1720
13.114 host_event.hpp . . . 1721
13.115 include/triSYCL/event/detail/opencl_event.hpp File Reference . . . 1722
13.116 opencl_event.hpp . . . 1722
13.117 include/triSYCL/exception.hpp File Reference . . . 1724
13.118 exception.hpp . . . 1726
13.119 include/triSYCL/group.hpp File Reference . . . 1728
13.120 group.hpp . . . 1729
13.121 include/triSYCL/h_item.hpp File Reference . . . 1732
13.122 h_item.hpp . . . 1733
13.123 include/triSYCL/half.hpp File Reference . . . 1736
13.124 half.hpp . . . 1737

13.125 include/triSYCL/handler.hpp File Reference . . . 1737
13.126 handler.hpp . . . 1738
13.127 include/triSYCL/id.hpp File Reference . . . 1745
13.128 id.hpp . . . 1746
13.129 include/triSYCL/image.hpp File Reference . . . 1747
13.129.1 Detailed Description . . . 1747
13.130 image.hpp . . . 1748
13.131 include/triSYCL/info/param_traits.hpp File Reference . . . 1748
13.131.1 Macro Definition Documentation . . . 1749
13.131.1.1 TRISYCL_INFO_PARAM_TRAITS . . . 1749
13.131.1.2 TRISYCL_INFO_PARAM_TRAITS_ANY_T . . . 1749
13.132 param_traits.hpp . . . 1750
13.133 include/triSYCL/item.hpp File Reference . . . 1750
13.134 item.hpp . . . 1751
13.135 include/triSYCL/kernel.hpp File Reference . . . 1753
13.136 kernel.hpp . . . 1754
13.137 include/triSYCL/kernel/detail/kernel.hpp File Reference . . . 1756
13.137.1 Macro Definition Documentation . . . 1757
13.137.1.1 TRISYCL_ParallelForKernel_RANGE . . . 1758
13.138 kernel.hpp . . . 1758
13.139 include/triSYCL/kernel/detail/opencl_kernel.hpp File Reference . . . 1759
13.139.1 Macro Definition Documentation . . . 1760
13.139.1.1 TRISYCL_ParallelForKernel_RANGE . . . 1761
13.140 opencl_kernel.hpp . . . 1761
13.141 include/triSYCL/math.hpp File Reference . . . 1763
13.141.1 Detailed Description . . . 1765
13.141.2 Macro Definition Documentation . . . 1765
13.141.2.1 TRISYCL_MATH_WRAP . . . 1765
13.141.2.2 TRISYCL_MATH_WRAP2 . . . 1766
13.141.2.3 TRISYCL_MATH_WRAP2s . . . 1766
13.141.2.4 TRISYCL_MATH_WRAP3 . . . 1766
13.141.2.5 TRISYCL_MATH_WRAP3s . . . 1767
13.141.2.6 TRISYCL_MATH_WRAP3ss . . . 1767
13.142 math.hpp . . . 1767
13.143 include/triSYCL/nd_item.hpp File Reference . . . 1772
13.144 nd_item.hpp . . . 1773
13.145 include/triSYCL/nd_range.hpp File Reference . . . 1776
13.146 nd_range.hpp . . . 1777
13.147 include/triSYCL/opencl_types.hpp File Reference . . . 1778
13.147.1 Detailed Description . . . 1780
13.147.2 Macro Definition Documentation . . . 1780
13.147.2.1 TRISYCL_BOOST_COMPUTE_NAME . . . 1780

13.147.2.2 TRISYCL_DECLARE_CL_TYPES . . . 1780
13.147.2.3 TRISYCL_DEFINE_TYPES . . . 1780
13.147.2.4 TRISYCL_H_DEFINE_TYPE . . . 1781
13.147.2.5 TRISYCL_IS_WRAPPER_TRAIT . . . 1781
13.147.2.6 TRISYCL_SCALAR_TYPES . . . 1781
13.147.2.7 TRISYCL_SIZED_NAME . . . 1782
13.147.2.8 TRISYCL_TYPE_ACTUAL_NAME . . . 1782
13.147.2.9 TRISYCL_TYPE_CL_NAME . . . 1782
13.147.2.10 TRISYCL_TYPE_NAME . . . 1782
13.147.2.11 TRISYCL_TYPEDEF_TYPE . . . 1782
13.147.2.12 TRISYCL_WRAPPER_CLASS_2 . . . 1783
13.147.2.13 TRISYCL_WRAPPER_CLASS_3 . . . 1783
13.147.2.14 TRISYCL_WRAPPER_CLASS_4 . . . 1783
13.148 opencl_types.hpp . . . 1784
13.149 include/triSYCL/parallelism.hpp File Reference . . . 1786
13.149.1 Detailed Description . . . 1787
13.150 parallelism.hpp . . . 1787
13.151 include/triSYCL/parallelism/detail/parallelism.hpp File Reference . . . 1788
13.151.1 Detailed Description . . . 1790
13.152 parallelism.hpp . . . 1790
13.153 include/triSYCL/parallelism/detail/parallelism_tbb.hpp File Reference . . . 1796
13.153.1 Detailed Description . . . 1797
13.154 parallelism_tbb.hpp . . . 1797
13.155 include/triSYCL/platform/detail/host_platform.hpp File Reference . . . 1800
13.156 host_platform.hpp . . . 1801
13.157 include/triSYCL/platform/detail/host_platform_tail.hpp File Reference . . . 1803
13.158 host_platform_tail.hpp . . . 1803
13.159 include/triSYCL/platform/detail/opencl_platform.hpp File Reference . . . 1804
13.160 opencl_platform.hpp . . . 1805
13.161 include/triSYCL/platform/detail/opencl_platform_tail.hpp File Reference . . . 1808
13.162 opencl_platform_tail.hpp . . . 1808
13.163 include/triSYCL/platform/detail/platform_tail.hpp File Reference . . . 1809
13.164 platform_tail.hpp . . . 1810
13.165 include/triSYCL/property_list.hpp File Reference . . . 1810
13.165.1 Macro Definition Documentation . . . 1811
13.165.1.1 TRISYCL_PROPERTY_CREATE . . . 1811
13.165.1.2 TRISYCL_PROPERTY_HAS_GET . . . 1812
13.166 property_list.hpp . . . 1812
13.167 include/triSYCL/queue/detail/host_queue.hpp File Reference . . . 1813
13.168 host_queue.hpp . . . 1814
13.169 include/triSYCL/queue/detail/opencl_queue.hpp File Reference . . . 1815
13.170 opencl_queue.hpp . . . 1816

13.171 include/triSYCL/queue/detail/queue_tail.hpp File Reference . . . 1818
13.171.1 Macro Definition Documentation . . . 1818
13.171.1.1 TRISYCL_DEFAULT_QUEUE_SELECTOR . . . 1818
13.172 queue_tail.hpp . . . 1819
13.173 include/triSYCL/range.hpp File Reference . . . 1819
13.174 range.hpp . . . 1821
13.175 include/triSYCL/rounding_mode.hpp File Reference . . . 1822
13.175.1 Detailed Description . . . 1823
13.176 rounding_mode.hpp . . . 1823
13.177 include/triSYCL/pipe.hpp File Reference . . . 1824
13.178 pipe.hpp . . . 1825
13.179 include/triSYCL/sycl_2_2/pipe.hpp File Reference . . . 1826
13.180 pipe.hpp . . . 1827
13.181 include/triSYCL/sycl_2_2/pipe/detail/pipe.hpp File Reference . . . 1829
13.182 pipe.hpp . . . 1830
13.183 include/triSYCL/sycl_2_2/pipe/detail/pipe_accessor.hpp File Reference . . . 1836
13.184 pipe_accessor.hpp . . . 1837
13.185 include/triSYCL/sycl_2_2/pipe_reservation.hpp File Reference . . . 1841
13.186 pipe_reservation.hpp . . . 1842
13.187 include/triSYCL/sycl_2_2/pipe_reservation/detail/pipe_reservation.hpp File Reference . . . 1844
13.188 pipe_reservation.hpp . . . 1846
13.189 include/triSYCL/sycl_2_2/static_pipe.hpp File Reference . . . 1848
13.190 static_pipe.hpp . . . 1849
13.191 include/triSYCL/vec.hpp File Reference . . . 1851
13.191.1 Detailed Description . . . 1853
13.192 vec.hpp . . . 1853
13.193 include/triSYCL/vec/detail/vec.hpp File Reference . . . 1860
13.193.1 Detailed Description . . . 1862
13.194 vec.hpp . . . 1862
13.195 include/triSYCL/vec/detail/swiz2.hpp File Reference . . . 1866
13.196 swiz2.hpp . . . 1866
13.197 include/triSYCL/vec/detail/swiz3.hpp File Reference . . . 1867
13.198 swiz3.hpp . . . 1867
13.199 include/triSYCL/vec/detail/swiz4.hpp File Reference . . . 1868
13.200 swiz4.hpp . . . 1868
13.201 include/triSYCL/vec/detail/swiz_rgba.hpp File Reference . . . 1872
13.202 swiz_rgba.hpp . . . 1872
13.203 include/sycl/vendor/triSYCL/pipe/cout.hpp File Reference . . . 1876
13.204 cout.hpp . . . 1876
13.205 include/triSYCL/vendor/triSYCL/pipe/cout.hpp File Reference . . . 1877
13.206 cout.hpp . . . 1878
13.207 include/sycl/vendor/triSYCL/random/xorshift.hpp File Reference . . . 1878

13.208 xorshift.hpp . . . 1879
13.209 include/triSYCL/vendor/triSYCL/random/xorshift.hpp File Reference . . . 1881
13.210 xorshift.hpp . . . 1882
13.211 include/sycl/vendor/triSYCL/scope.hpp File Reference . . . 1883
13.211.1 Macro Definition Documentation . . . 1884
13.211.1.1 SYCL_VENDOR_TRISYCL_CONCEPTUAL_API . . . 1884
13.211.1.2 SYCL_VENDOR_TRISYCL_SCOPE . . . 1884
13.212 scope.hpp . . . 1884
13.213 include/triSYCL/vendor/triSYCL/scope.hpp File Reference . . . 1885
13.213.1 Macro Definition Documentation . . . 1885
13.213.1.1 SYCL_VENDOR_TRISYCL_CONCEPTUAL_API . . . 1885
13.213.1.2 SYCL_VENDOR_TRISYCL_SCOPE . . . 1885
13.214 scope.hpp . . . 1886
13.215 include/sycl/vendor/triSYCL/scope/detail/util.hpp File Reference . . . 1886
13.216 util.hpp . . . 1887
13.217 include/triSYCL/vendor/triSYCL/scope/detail/util.hpp File Reference . . . 1888
13.218 util.hpp . . . 1889
13.219 include/sycl/vendor/triSYCL/scope/platform.hpp File Reference . . . 1889
13.219.1 Macro Definition Documentation . . . 1890
13.219.1.1 SYCL_VENDOR_TRISYCL_PLATFORM_SCOPE . . . 1890
13.220 platform.hpp . . . 1890
13.221 include/sycl/vendor/triSYCL/scope/platform/detail/platform.hpp File Reference . . . 1892
13.222 platform.hpp . . . 1892
13.223 include/triSYCL/info/platform.hpp File Reference . . . 1893
13.224 platform.hpp . . . 1895
13.225 include/triSYCL/platform.hpp File Reference . . . 1896
13.226 platform.hpp . . . 1898
13.227 include/triSYCL/platform/detail/platform.hpp File Reference . . . 1900
13.228 platform.hpp . . . 1901
13.229 include/triSYCL/vendor/triSYCL/scope/platform.hpp File Reference . . . 1903
13.229.1 Macro Definition Documentation . . . 1904
13.229.1.1 SYCL_VENDOR_TRISYCL_PLATFORM_SCOPE . . . 1904
13.230 platform.hpp . . . 1904
13.231 include/triSYCL/vendor/triSYCL/scope/platform/detail/platform.hpp File Reference . . . 1906
13.232 platform.hpp . . . 1907
13.233 include/sycl/vendor/Xilinx/acap.hpp File Reference . . . 1907
13.233.1 Detailed Description . . . 1908
13.234 acap.hpp . . . 1908
13.235 include/triSYCL/vendor/Xilinx/acap.hpp File Reference . . . 1908
13.235.1 Detailed Description . . . 1909
13.236 acap.hpp . . . 1909
13.237 include/sycl/vendor/Xilinx/acap/aie.hpp File Reference . . . 1910

13.237.1 Detailed Description . . . 1911
13.238 aie.hpp . . . 1911
13.239 include/triSYCL/vendor/Xilinx/acap/aie.hpp File Reference . . . 1911
13.239.1 Detailed Description . . . 1912
13.240 aie.hpp . . . 1912
13.241 include/sycl/vendor/Xilinx/acap/aie/accessor.hpp File Reference . . . 1913
13.241.1 Detailed Description . . . 1914
13.242 accessor.hpp . . . 1914
13.243 include/triSYCL/accessor.hpp File Reference . . . 1915
13.244 accessor.hpp . . . 1916
13.245 include/triSYCL/accessor/facade/accessor.hpp File Reference . . . 1922
13.246 accessor.hpp . . . 1923
13.247 include/triSYCL/accessor/mixin/accessor.hpp File Reference . . . 1925
13.248 accessor.hpp . . . 1927
13.249 include/triSYCL/buffer/detail/accessor.hpp File Reference . . . 1929
13.250 accessor.hpp . . . 1930
13.251 include/triSYCL/vendor/Xilinx/acap/aie/accessor.hpp File Reference . . . 1933
13.251.1 Detailed Description . . . 1934
13.252 accessor.hpp . . . 1935
13.253 include/sycl/vendor/Xilinx/acap/aie/axi_stream_switch.hpp File Reference . . . 1935
13.253.1 Detailed Description . . . 1937
13.254 axi_stream_switch.hpp . . . 1937
13.255 include/triSYCL/vendor/Xilinx/acap/aie/axi_stream_switch.hpp File Reference . . . 1945
13.255.1 Detailed Description . . . 1947
13.256 axi_stream_switch.hpp . . . 1947
13.257 include/sycl/vendor/Xilinx/acap/aie/cascade_stream.hpp File Reference . . . 1955
13.257.1 Detailed Description . . . 1956
13.258 cascade_stream.hpp . . . 1957
13.259 include/triSYCL/vendor/Xilinx/acap/aie/cascade_stream.hpp File Reference . . . 1958
13.259.1 Detailed Description . . . 1960
13.260 cascade_stream.hpp . . . 1960
13.261 include/sycl/vendor/Xilinx/acap/aie/connection.hpp File Reference . . . 1961
13.262 connection.hpp . . . 1963
13.263 include/triSYCL/vendor/Xilinx/acap/aie/connection.hpp File Reference . . . 1968
13.264 connection.hpp . . . 1970
13.265 include/sycl/vendor/triSYCL/scope/device.hpp File Reference . . . 1975
13.265.1 Macro Definition Documentation . . . 1975
13.265.1.1 SYCL_VENDOR_TRISYCL_DEVICE_SCOPE . . . 1976
13.266 device.hpp . . . 1976
13.267 include/sycl/vendor/triSYCL/scope/device/detail/device.hpp File Reference . . . 1978
13.268 device.hpp . . . 1978
13.269 include/sycl/vendor/Xilinx/acap/aie/device.hpp File Reference . . . 1979

13.269.1 Detailed Description . . . 1981
13.270 device.hpp . . . 1981
13.271 include/sycl/vendor/Xilinx/acap/aie/device/detail/device.hpp File Reference . . . 1984
13.271.1 Detailed Description . . . 1985
13.272 device.hpp . . . 1986
13.273 include/triSYCL/device.hpp File Reference . . . 1989
13.274 device.hpp . . . 1990
13.275 include/triSYCL/device/detail/device.hpp File Reference . . . 1994
13.276 device.hpp . . . 1995
13.277 include/triSYCL/device/facade/device.hpp File Reference . . . 1997
13.278 device.hpp . . . 1998
13.279 include/triSYCL/info/device.hpp File Reference . . . 1998
13.280 device.hpp . . . 2001
13.281 include/triSYCL/vendor/triSYCL/scope/device.hpp File Reference . . . 2004
13.281.1 Macro Definition Documentation . . . 2005
13.281.1.1 SYCL_VENDOR_TRISYCL_DEVICE_SCOPE . . . 2005
13.282 device.hpp . . . 2006
13.283 include/triSYCL/vendor/triSYCL/scope/device/detail/device.hpp File Reference . . . 2008
13.284 device.hpp . . . 2009
13.285 include/triSYCL/vendor/Xilinx/acap/aie/device.hpp File Reference . . . 2010
13.285.1 Detailed Description . . . 2011
13.286 device.hpp . . . 2012
13.287 include/triSYCL/vendor/Xilinx/acap/aie/device/detail/device.hpp File Reference . . . 2015
13.287.1 Detailed Description . . . 2017
13.288 device.hpp . . . 2017
13.289 include/sycl/vendor/Xilinx/acap/aie/dma.hpp File Reference . . . 2020
13.289.1 Detailed Description . . . 2023
13.290 dma.hpp . . . 2023
13.291 include/triSYCL/vendor/Xilinx/acap/aie/dma.hpp File Reference . . . 2027
13.291.1 Detailed Description . . . 2029
13.292 dma.hpp . . . 2029
13.293 include/sycl/vendor/Xilinx/acap/aie/geography.hpp File Reference . . . 2033
13.293.1 Detailed Description . . . 2035
13.294 geography.hpp . . . 2035
13.295 include/triSYCL/vendor/Xilinx/acap/aie/geography.hpp File Reference . . . 2045
13.295.1 Detailed Description . . . 2047
13.296 geography.hpp . . . 2047
13.297 include/sycl/vendor/Xilinx/acap/aie/layout.hpp File Reference . . . 2057
13.297.1 Detailed Description . . . 2058
13.298 layout.hpp . . . 2058
13.299 include/triSYCL/vendor/Xilinx/acap/aie/layout.hpp File Reference . . . 2059
13.299.1 Detailed Description . . . 2060

13.300 layout.hpp . . . 2060
13.301 include/sycl/vendor/Xilinx/acap/aie/layout/full.hpp File Reference . . . 2061
13.301.1 Detailed Description . . . 2062
13.302 full.hpp . . . 2062
13.303 include/triSYCL/vendor/Xilinx/acap/aie/layout/full.hpp File Reference . . . 2063
13.303.1 Detailed Description . . . 2064
13.304 full.hpp . . . 2065
13.305 include/sycl/vendor/Xilinx/acap/aie/layout/one_pe.hpp File Reference . . . 2065
13.305.1 Detailed Description . . . 2067
13.306 one_pe.hpp . . . 2067
13.307 include/triSYCL/vendor/Xilinx/acap/aie/layout/one_pe.hpp File Reference . . . 2068
13.307.1 Detailed Description . . . 2070
13.308 one_pe.hpp . . . 2070
13.309 include/sycl/vendor/Xilinx/acap/aie/layout/size.hpp File Reference . . . 2070
13.309.1 Detailed Description . . . 2072
13.310 size.hpp . . . 2072
13.311 include/triSYCL/vendor/Xilinx/acap/aie/layout/size.hpp File Reference . . . 2073
13.311.1 Detailed Description . . . 2075
13.312 size.hpp . . . 2075
13.313 include/sycl/vendor/Xilinx/acap/aie/layout/small.hpp File Reference . . . 2075
13.313.1 Detailed Description . . . 2077
13.314 small.hpp . . . 2077
13.315 include/triSYCL/vendor/Xilinx/acap/aie/layout/small.hpp File Reference . . . 2078
13.315.1 Detailed Description . . . 2080
13.316 small.hpp . . . 2080
13.317 include/sycl/vendor/Xilinx/acap/aie/layout/versal.hpp File Reference . . . 2080
13.317.1 Detailed Description . . . 2082
13.318 versal.hpp . . . 2082
13.319 include/triSYCL/vendor/Xilinx/acap/aie/layout/versal.hpp File Reference . . . 2083
13.319.1 Detailed Description . . . 2085
13.320 versal.hpp . . . 2085
13.321 include/sycl/vendor/Xilinx/acap/aie/lock.hpp File Reference . . . 2086
13.321.1 Detailed Description . . . 2088
13.322 lock.hpp . . . 2088
13.323 include/triSYCL/vendor/Xilinx/acap/aie/lock.hpp File Reference . . . 2089
13.323.1 Detailed Description . . . 2092
13.324 lock.hpp . . . 2092
13.325 include/sycl/vendor/Xilinx/acap/aie/memory.hpp File Reference . . . 2093
13.325.1 Detailed Description . . . 2095
13.326 memory.hpp . . . 2096
13.327 include/triSYCL/vendor/Xilinx/acap/aie/memory.hpp File Reference . . . 2097
13.327.1 Detailed Description . . . 2099

13.328 memory.hpp . . . 2099
13.329 include/sycl/vendor/Xilinx/acap/aie/memory_base.hpp File Reference . . . 2100
13.329.1 Detailed Description . . . 2102
13.330 memory_base.hpp . . . 2102
13.331 include/triSYCL/vendor/Xilinx/acap/aie/memory_base.hpp File Reference . . . 2103
13.331.1 Detailed Description . . . 2105
13.332 memory_base.hpp . . . 2105
13.333 include/sycl/vendor/Xilinx/acap/aie/memory_infrastructure.hpp File Reference . . . 2106
13.333.1 Detailed Description . . . 2108
13.334 memory_infrastructure.hpp . . . 2108
13.335 include/triSYCL/vendor/Xilinx/acap/aie/memory_infrastructure.hpp File Reference . . . 2109
13.335.1 Detailed Description . . . 2111
13.336 memory_infrastructure.hpp . . . 2111
13.337 include/sycl/vendor/Xilinx/acap/aie/program.hpp File Reference . . . 2112
13.337.1 Detailed Description . . . 2113
13.338 program.hpp . . . 2114
13.339 include/triSYCL/program.hpp File Reference . . . 2117
13.340 program.hpp . . . 2118
13.341 include/triSYCL/program/detail/program.hpp File Reference . . . 2120
13.342 program.hpp . . . 2121
13.343 include/triSYCL/vendor/Xilinx/acap/aie/program.hpp File Reference . . . 2121
13.343.1 Detailed Description . . . 2123
13.344 program.hpp . . . 2123
13.345 include/sycl/vendor/triSYCL/scope/queue.hpp File Reference . . . 2126
13.345.1 Macro Definition Documentation . . . 2127
13.345.1.1 SYCL_VENDOR_TRISYCL_QUEUE_SCOPE . . . 2127
13.346 queue.hpp . . . 2128
13.347 include/sycl/vendor/triSYCL/scope/queue/detail/queue.hpp File Reference . . . 2130
13.348 queue.hpp . . . 2131
13.349 include/sycl/vendor/Xilinx/acap/aie/queue.hpp File Reference . . . 2132
13.349.1 Detailed Description . . . 2133
13.350 queue.hpp . . . 2134
13.351 include/triSYCL/info/queue.hpp File Reference . . . 2135
13.352 queue.hpp . . . 2136
13.353 include/triSYCL/property/queue.hpp File Reference . . . 2137
13.354 queue.hpp . . . 2138
13.355 include/triSYCL/queue.hpp File Reference . . . 2138
13.356 queue.hpp . . . 2139
13.357 include/triSYCL/queue/detail/queue.hpp File Reference . . . 2144
13.358 queue.hpp . . . 2145
13.359 include/triSYCL/vendor/triSYCL/scope/queue.hpp File Reference . . . 2147
13.359.1 Macro Definition Documentation . . . 2148

13.359.1.1 SYCL_VENDOR_TRISYCL_QUEUE_SCOPE . . . 2148
13.360 queue.hpp . . . 2149
13.361 include/triSYCL/vendor/triSYCL/scope/queue/detail/queue.hpp File Reference . . . 2151
13.362 queue.hpp . . . 2152
13.363 include/triSYCL/vendor/Xilinx/acap/aie/queue.hpp File Reference . . . 2153
13.363.1 Detailed Description . . . 2155
13.364 queue.hpp . . . 2155
13.365 include/sycl/vendor/Xilinx/acap/aie/shim_tile.hpp File Reference . . . 2157
13.365.1 Detailed Description . . . 2158
13.366 shim_tile.hpp . . . 2158
13.367 include/triSYCL/vendor/Xilinx/acap/aie/shim_tile.hpp File Reference . . . 2160
13.367.1 Detailed Description . . . 2162
13.368 shim_tile.hpp . . . 2162
13.369 include/sycl/vendor/Xilinx/acap/aie/tile.hpp File Reference . . . 2164
13.369.1 Detailed Description . . . 2166
13.370 tile.hpp . . . 2166
13.371 include/triSYCL/vendor/Xilinx/acap/aie/tile.hpp File Reference . . . 2171
13.371.1 Detailed Description . . . 2173
13.372 tile.hpp . . . 2173
13.373 include/sycl/vendor/Xilinx/acap/aie/tile_base.hpp File Reference . . . 2178
13.373.1 Detailed Description . . . 2180
13.374 tile_base.hpp . . . 2180
13.375 include/triSYCL/vendor/Xilinx/acap/aie/tile_base.hpp File Reference . . . 2181
13.375.1 Detailed Description . . . 2184
13.376 tile_base.hpp . . . 2184
13.377 include/sycl/vendor/Xilinx/acap/aie/tile_infrastructure.hpp File Reference . . . 2186
13.377.1 Detailed Description . . . 2188
13.378 tile_infrastructure.hpp . . . 2188
13.379 include/sycl/vendor/Xilinx/acap/aie/tile_infrastructure/detail/tile_infrastructure.hpp File Reference . . . 2191
13.379.1 Detailed Description . . . 2193
13.380 tile_infrastructure.hpp . . . 2193
13.381 include/triSYCL/vendor/Xilinx/acap/aie/tile_infrastructure.hpp File Reference . . . 2197
13.381.1 Detailed Description . . . 2199
13.382 tile_infrastructure.hpp . . . 2199
13.383 include/triSYCL/vendor/Xilinx/acap/aie/tile_infrastructure/detail/tile_infrastructure.hpp File Reference . . . 2202
13.383.1 Detailed Description . . . 2204
13.384 tile_infrastructure.hpp . . . 2204
13.385 include/sycl/vendor/Xilinx/acap/debug.hpp File Reference . . . 2208
13.385.1 Detailed Description . . . 2209
13.386 debug.hpp . . . 2209
13.387 include/triSYCL/detail/debug.hpp File Reference . . . 2210
13.387.1 Macro Definition Documentation . . . 2212

13.387.1.1 TRISYCL_DUMP . . . 2212
13.387.1.2 TRISYCL_DUMP_T . . . 2212
13.387.1.3 TRISYCL_INTERNAL_DUMP . . . 2212
13.388 debug.hpp . . . 2213
13.389 include/triSYCL/vendor/Xilinx/acap/debug.hpp File Reference . . . 2215
13.389.1 Detailed Description . . . 2216
13.390 debug.hpp . . . 2217
13.391 include/sycl/vendor/Xilinx/acap/tests/processor_array.cpp File Reference . . . 2218
13.391.1 Function Documentation . . . 2218
13.391.1.1 main() . . . 2218
13.392 processor_array.cpp . . . 2218
13.393 include/triSYCL/vendor/Xilinx/acap/tests/processor_array.cpp File Reference . . . 2220
13.393.1 Function Documentation . . . 2220
13.393.1.1 main() . . . 2220
13.394 processor_array.cpp . . . 2220
13.395 include/sycl/vendor/Xilinx/config.hpp File Reference . . . 2222
13.395.1 Detailed Description . . . 2222
13.395.2 Macro Definition Documentation . . . 2222
13.395.2.1 TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS . . . 2222
13.395.2.2 TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER . . . 2223
13.396 config.hpp . . . 2223
13.397 include/triSYCL/vendor/Xilinx/config.hpp File Reference . . . 2223
13.397.1 Detailed Description . . . 2224
13.397.2 Macro Definition Documentation . . . 2225
13.397.2.1 TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS . . . 2225
13.397.2.2 TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER . . . 2225
13.398 config.hpp . . . 2225
13.399 include/sycl/vendor/Xilinx/fpga.hpp File Reference . . . 2226
13.400 fpga.hpp . . . 2226
13.401 include/triSYCL/vendor/Xilinx/fpga.hpp File Reference . . . 2226
13.402 fpga.hpp . . . 2227
13.403 include/sycl/vendor/Xilinx/fpga/opt_decorate_func.hpp File Reference . . . 2228
13.404 opt_decorate_func.hpp . . . 2228
13.405 include/triSYCL/vendor/Xilinx/fpga/opt_decorate_func.hpp File Reference . . . 2229
13.406 opt_decorate_func.hpp . . . 2230
13.407 include/sycl/vendor/Xilinx/fpga/partition_array.hpp File Reference . . . 2231
13.408 partition_array.hpp . . . 2232
13.409 include/triSYCL/vendor/Xilinx/fpga/partition_array.hpp File Reference . . . 2235
13.410 partition_array.hpp . . . 2236
13.411 include/sycl/vendor/Xilinx/fpga/ssdm_inst.hpp File Reference . . . 2239
13.412 ssdm_inst.hpp . . . 2239
13.413 include/triSYCL/vendor/Xilinx/fpga/ssdm_inst.hpp File Reference . . . 2240

13.414 ssdm_inst.hpp 2241
13.415 include/sycl/vendor/Xilinx/graphics.hpp File Reference 2241
13.416 graphics.hpp 2242
13.417 include/triSYCL/vendor/Xilinx/graphics.hpp File Reference 2251
13.418 graphics.hpp 2252
13.419 include/sycl/vendor/Xilinx/latex.hpp File Reference 2260
13.420 latex.hpp 2261
13.421 include/triSYCL/vendor/Xilinx/latex.hpp File Reference 2263
13.422 latex.hpp 2264

Index **2267**

Chapter 1

triSYCL implementation of SYCL

This is the main triSYCL C++ header file to experiment with the SYCL specification.

It declares everything into the `trisycl` namespace instead of `sycl` for SYCL 2020 or `cl::sycl` for SYCL 1.2.1 to express triSYCL specific extensions and to be used in full or partially with other SYCL implementations without naming conflict.

You can use sycl/sycl.hpp or CL/sycl.hpp headers instead to use this implementation with the usual `sycl` or `cl::sycl` namespaces.

For more information about SYCL: http://www.khronos.org/sycl/

For more information on this project and to access to the source of this file, look at https://github.com/triSYCL/triSYCL

The Doxygen version of the implementation itself is in http://trisycl.github.io/triSYCL/Doxygen/triSYCL/html and http://trisycl.github.io/triSYCL/Doxygen/triSYCL/triSYCL-implementation-refman.pdf

Ronan at keryell dot FR

Copyright 2014–2015 Advanced Micro Devices, Inc.

Copyright 2015–2022 Xilinx, Inc.

Copyright 2022–2023 Advanced Micro Devices, Inc.

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Chapter 2

Todo List

Member __global

avoid this hack

File address_space.hpp

Add the alias ..._ptr<T> = ...<T *>

Member const_func

avoid this hack

File opencl_spir_helpers.hpp

implement group and nd_item and to add nd_item to create_parallel_for_arg

File parallelism.hpp

Refactor this file

Namespace trisycl::access

This values should be normalized to allow separate compilation with different implementations?

Class trisycl::accessor< DataType, Dimensions, AccessMode, Target >

Implement it for images according so section 3.3.4.5

Member trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (buffer< DataType, Dimensions, Allocator > &target_buffer, handler &command_group_handler)

Add template allocator type in all the accessor constructors in the specification or just use a more opaque Buffer type?

fix specification where access mode should be target instead

Member trisycl::accessor< DataType, Dimensions, AccessMode, Target >::begin () const

Factor out these in a template helper

Add these functions to the specification

The fact that the lambda capture make a const copy of the accessor is not yet elegantly managed... The issue is that begin()/end() dispatch is made according to the accessor constness and not from the array member constness...

The issue is that the end may not be known if it is implemented by a raw OpenCL cl_mem... So only provide on the device the iterators related to the start? Actually the accessor needs to know a part of the shape to have the multidimentional addressing. So this only require a size_t more...

try to solve it by using some enable_if on array constness?

Member trisycl::accessor< DataType, Dimensions, AccessMode, Target >::dimensionality

in the specification: store the dimension for user request

Member trisycl::accessor< DataType, Dimensions, AccessMode, Target >::get_count () const

Move on https://cvs.khronos.org/bugzilla/show_bug.cgi?id=15564 and https://cvs.khronos.org/bugzilla/show_bug.cgi?id=14404

Member trisycl::accessor< DataType, Dimensions, AccessMode, Target >::get_pointer () const

Should it be named data() instead?

Member trisycl::accessor< DataType, Dimensions, AccessMode, Target >::get_range () const

Move on https://cvs.khronos.org/bugzilla/show_bug.cgi?id=15564 and https://cvs.khronos.org/bugzilla/show_bug.cgi?id=14404

Member trisycl::accessor< DataType, Dimensions, AccessMode, Target >::get_size () const

Move on https://cvs.khronos.org/bugzilla/show_bug.cgi?id=15564 and https://cvs.khronos.org/bugzilla/show_bug.cgi?id=14404

It is incompatible with buffer get_size() in the spec

Member trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator* ()

Add in the specification

Member trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator* () const

Add in the specification?

Add the concept of 0-dim buffer and accessor for scalar and use an implicit conversion to value_type reference to access the value with the accessor?

Member trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator[] (const nd_item< dimensionality > &index) const

Add in the specification because used by HPC-GPU slide 22

Member trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator[] (const nd_item< dimensionality > &index)

Add in the specification because used by HPC-GPU slide 22

Member trisycl::buffer (FIB begin, FIE end) -> buffer< typename std::iterator_traits< FIB >::value_type, 1 >

Use some standard concepts

Add this to SYCL Next

Member trisycl::buffer (auto &host_data) -> buffer< std::ranges::range_value_t< decltype(host_data)>, 1 >

Use some standard concepts

Add this to SYCL Next

Class trisycl::buffer< T, Dimensions, Allocator >

There is a naming inconsistency in the specification between buffer and accessor on T versus datatype

Finish allocator implementation

Think about the need of an allocator when constructing a buffer from other buffers

Update the specification to have a non-const allocator for const buffer? Or do we rely on rebind_alloc<T>. But does this work with astate-full allocator?

Add constructors from arrays so that in C++17 the range and type can be infered from the constructor

Add constructors from array_ref

Member trisycl::buffer< T, Dimensions, Allocator >::buffer (cl_mem mem_object, queue from_queue, event available_event={}, Allocator allocator={})

To be implemented

Improve the specification to allow CLHPP objects too

Member trisycl::buffer< T, Dimensions, Allocator >::buffer (shared_ptr_class< T > host_data, const range< Dimensions > &buffer_range, Allocator allocator={})

add this mutex-less constructor to the specification

Member trisycl::buffer< T, Dimensions, Allocator >::buffer (buffer< T, Dimensions, Allocator > &b, const id< Dimensions > &base_index, const range< Dimensions > &sub_range, Allocator allocator={})

To be implemented

Member trisycl::buffer< T, Dimensions, Allocator >::buffer (shared_ptr_class< T > &host_data, const range< Dimensions > &buffer_range, trisycl::mutex_class &m, Allocator allocator={})

update the specification to replace the pointer by a reference and provide the constructor with and without a mutex

Member trisycl::buffer< T, Dimensions, Allocator >::buffer (const T *host_data, const range< Dimensions > &r, Allocator allocator={})

Actually this is redundant.

Member trisycl::buffer< T, Dimensions, Allocator >::buffer (InputIterator start_iterator, InputIterator end_iterator, Allocator allocator={})

Implement the copy back at buffer destruction

Generalize this for n-D and provide column-major and row-major initialization

a reason to have this nD is that set_final_data(weak_ptr_class<T> & finalData) is actually doing this linearization anyway

Allow read-only buffer construction too

update the specification to deal with forward iterators instead and rewrite back only when it is non const and output iterator at least

Allow initialization from ranges and collections à la STL

Member trisycl::buffer< T, Dimensions, Allocator >::buffer (buffer< T, Dimensions, Allocator > &b, const id< Dimensions > &base_index, const range< Dimensions > &sub_range, Allocator allocator={})

Update the specification to replace index by id

Member trisycl::buffer< T, Dimensions, Allocator >::get_access (handler &command_group_handler)

Do we need for an accessor to increase the reference count of a buffer object? It does make more sense for a host-side accessor.

Implement the modes and targets

Member trisycl::buffer< T, Dimensions, Allocator >::get_access ()

More elegant solution

Implement the modes

Member trisycl::buffer< T, Dimensions, Allocator >::get_range () const

rename to the equivalent from array_ref proposals? Such as size() in http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0009r2.html

Member trisycl::buffer< T, Dimensions, Allocator >::get_size () const

rename to something else. In http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/p0122r0.pdf it is named bytes() for example

Member trisycl::buffer< T, Dimensions, Allocator >::is_read_only () const

Add to specification

Member trisycl::buffer< T, Dimensions, Allocator >::use_count () const

Add to the specification, useful for validation

Class trisycl::context

The implementation is quite minimal for now.

Member trisycl::context::get_devices () const

To be implemented

Member trisycl::context::get_info () const

To be implemented

Member trisycl::context::get_platform () const

To be implemented

Class trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >

Implement a real local accessor handling several work-group running in parallel

Class trisycl::detail::accessor< T, Dimensions, Mode, Target >

Use the access::mode

Member trisycl::detail::accessor< T, Dimensions, Mode, Target >::accessor (std::shared_ptr< detail::buffer< T, Dimensions > > target_buffer)

fix the specification to rename target that shadows template parm

Member trisycl::detail::accessor< T, Dimensions, Mode, Target >::accessor (std::shared_ptr< detail::buffer< T, Dimensions > > target_buffer, handler &command_group_handler)

fix the specification to rename target that shadows template parm

Member trisycl::detail::accessor< T, Dimensions, Mode, Target >::is_read_access () const

to move in the access::mode enum class and add to the specification ?

Strangely, it is not really constexpr because it is not a static method...

Member trisycl::detail::accessor< T, Dimensions, Mode, Target >::is_write_access () const

to move in the access::mode enum class and add to the specification ?

Strangely, it is not really constexpr because it is not a static method...

Member trisycl::detail::accessor< T, Dimensions, Mode, Target >::register_accessor ()

Double-check with the C++ committee on this issue.

Member trisycl::detail::address_space_array< T, AS >::address_space_array (std::initializer_list< std::remove_extent_t< T > > list)

Extend to more than 1 dimension

Class trisycl::detail::address_space_base< T, AS >

Verify/improve to deal with const/volatile?

Member trisycl::detail::address_space_base< T, AS >::opencl_type

Add to the specification

Member trisycl::detail::address_space_base< T, AS >::type

Add to the specification

Class trisycl::detail::address_space_fundamental< T, AS >

Verify/improve to deal with const/volatile?

Class trisycl::detail::address_space_object< T, AS >

Verify/improve to deal with const/volatile?

what about T having some final methods?

Member trisycl::detail::address_space_object< T, AS >::opencl_type

Add to the specification

Member trisycl::detail::address_space_variable< T, AS >::opencl_type

Add to the specification

Member trisycl::detail::buffer< T, Dimensions >::alloc

Implement user-provided allocator

Member trisycl::detail::buffer< T, Dimensions >::buffer (const range< Dimensions > &r)

Lazily allocate memory since it might not be used on host

Member trisycl::detail::buffer< T, Dimensions >::buffer (const T *host_data, const range< Dimensions > &r)

Clarify the semantics in the spec. What happens if the host change the host_data after buffer creation?

Member trisycl::detail::buffer< T, Dimensions >::call_update_buffer_state (trisycl::context ctx, access::mode mode, size_t size, DataType *data, std::enable_if_t<!std::is_const< BaseType >::value > *=0)

Use `if constexpr` when it is available with C++17

Member trisycl::detail::buffer< T, Dimensions >::get_destructor_future ()

Make the function private again

Member trisycl::detail::buffer< T, Dimensions >::~buffer ()

To implement and deal with reference counting buffer(buffer<T, Dimensions> b, index<Dimensions> base_↩ index, range<Dimensions> sub_range)

Allow CLHPP objects too?

Member trisycl::detail::buffer_add_to_task (BufferDetail buf, handler *command_group_handler, bool is↩ _write_mode)

To remove with some refactoring

Member trisycl::detail::buffer_base::buffer_base ()

Use lazy allocation for the context tracking set

Member trisycl::detail::context::get_devices () const =0

virtual cannot be templated template <info::context Param> typename info::param_traits<info::context, Param>::type get_info() const = 0;

Class trisycl::detail::host_context

The implementation is quite minimal for now. :-)

Member trisycl::detail::host_context::get_devices () const override

To be implemented

Class trisycl::detail::host_device

The implementation is quite minimal for now. :-)

Member trisycl::detail::host_device::get_platform () const override

To be implemented

Member trisycl::detail::host_device::has_extension (const string_class &extension) const override

To be implemented

Member trisycl::detail::host_platform::has_extension (const string_class &extension) const override

To be implemented

Member trisycl::detail::opencl_context::get_devices () const override

To be implemented

Member trisycl::detail::opencl_context::get_platform () const override

To be implemented

Member trisycl::detail::opencl_kernel::get () const override

Test error and throw. Externalize this feature in Boost.Compute?

Improve the spec to deprecate C OpenCL host API and move to C++ instead to avoid this ugly ownership management

Member trisycl::detail::opencl_kernel::single_task (std::shared_ptr< detail::task > task, std::shared_ptr< detail::queue > q) override

Remove either task or q

Member trisycl::detail::opencl_queue::instance (const trisycl::device &d)

Check with SYCL committee what is the expected behaviour here about the context. Is this a new context everytime, or always the same for a given device?

Member trisycl::detail::parallel_for (nd_range< Dimensions > r, ParallelForFunctor f)

Add an OpenMP implementation

Deal with incomplete work-groups

Implement with parallel_for_workgroup()/parallel_for_workitem()

Member trisycl::detail::parallel_for_workitem (const group< Dimensions > &g, ParallelForFunctor f)

Better type the functor

Member trisycl::detail::queue::~queue ()

Update according spec since queue destruction is non blocking

Member trisycl::detail::shared_ptr_implementation< Parent, Implementation >::operator< (const Parent &lhs, const Parent &rhs)

Add this to the spec

Member trisycl::detail::sycl_2_2::pipe< T >::write (const T &value, bool blocking=false)

provide a && version

Member trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::pipe_accessor (const std::shared_ptr< detail::sycl_2_2::pipe< T > > &p)

Use pipe_exception instead

Member trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::write (const value_type &value) const

provide a && version

Member trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::assume_validity ()

Throw exception instead

Member trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::commit ()

Add to the specification that for simplicity a reservation can be commited several times but only the first one is taken into account

Member trisycl::detail::task::buffers_in_use

Use a set to check that some buffers are not used many times at least on writing

Member trisycl::detail::task::get_kernel ()

Specify this error in the spec

Member trisycl::detail::task::schedule (std::function< void(void)> f)

This is an issue if there is an exception in the kernel

Member trisycl::detail::vec< DataType, NumElements >::zip (::trisycl::vec< T, size > y, ::trisycl::vec< T, size > z, F f) const

this should be generalized to any arity

Member trisycl::device::device (const device_selector &ds)

Make it non-explicit in the specification?

Member trisycl::device::get_info () const

Member trisycl::device::type () const

Present in Boost.Compute, to be added to the specification

Member trisycl::device_selector::select_device () const

Remove this from specification

Class trisycl::device_type_selector

To be added to the specification

Class trisycl::device_typename_selector< DeviceType >

To be added to the specification

Member trisycl::drt::code::program::p

Deal with several programs

Member trisycl::drt::kernel_IR

use std::byte when moving to C++17

Member trisycl::error_handler::default_handler

add this concept to the specification?

Member trisycl::error_handler::report_error (exception &error)=0

Add "virtual void" to the specification

Class trisycl::exception_list

Do we need to define it in SYCL or can we rely on plain C++17 one?

Member trisycl::exception_ptr

Do we need this instead of reusing directly the one from C++11?

Member trisycl::facade::accessor< AccessorMixIn >::begin ()

Add these functions to the specification

Do we need this in detail::accessor too or only in accessor?

Factor out these in a template helper

The issue is that the end may not be known if it is implemented by a raw OpenCL cl_mem... So only provide on the device the iterators related to the start? Actually the accessor needs to know a part of the shape to have the multidimentional addressing. So this only require a size_t more...

try to solve it by using some enable_if on array constness?

The fact that the lambda capture make a const copy of the accessor is not yet elegantly managed... The issue is that begin()/end() dispatch is made according to the accessor constness and not from the array member constness...

Member trisycl::facade::accessor< AccessorMixIn >::get_pointer ()

Implement the various pointer address spaces

Member trisycl::facade::accessor< AccessorMixIn >::operator* () const

Add the concept of 0-dim buffer and accessor for scalar and use an implicit conversion to value_type reference to access the value with the accessor?

Add to the specification?

Member trisycl::facade::accessor< AccessorMixIn >::operator* ()

Add to the specification

Member trisycl::facade::accessor< AccessorMixIn >::operator[] (const nd_item< mixin::rank()> &index)

Add to the specification because used by HPC-GPU slide 22

Member trisycl::facade::accessor< AccessorMixIn >::operator[] (const nd_item< mixin::rank()> &index) const

Add to the specification because used by HPC-GPU slide 22

Member trisycl::group< Dimensions >::get_group_range () const

Fix this comment and the specification

Member trisycl::group< Dimensions >::get_local_range () const

Add to the specification

Member trisycl::group< Dimensions >::get_local_range (int dimension) const

Add to the specification

Member trisycl::group< Dimensions >::get_nd_range () const

Also provide this access to the current nd_range

Member trisycl::group< Dimensions >::get_offset (int dimension) const

Add to the specification

Member trisycl::group< Dimensions >::get_offset () const

Add to the specification

Member trisycl::group< Dimensions >::group (const nd_range< Dimensions > &ndr)

This should be private since it is only used by the triSYCL implementation

Member trisycl::group< Dimensions >::group (const id< Dimensions > &i, const nd_range< Dimensions > &ndr)

This should be private somehow, but it is used by the validation infrastructure

Member trisycl::group< Dimensions >::group ()=default

Make most of them protected, reserved to implementation

Member trisycl::group< Dimensions >::operator[] (int dimension)

In this implementation it is not const because the group<> is written in the parallel_for iterators. To fix according to the specification

Member trisycl::h_item< Dimensions >::get_item () const

Add to the specification

Member trisycl::h_item< Dimensions >::h_item (nd_range< Dimensions > ndr)

This is for the triSYCL implementation which is expected to call set_global() and set_local() later. This should be hidden to the user.

Member trisycl::h_item< Dimensions >::h_item (id< Dimensions > global_index, nd_range< Dimensions > ndr)

This is for validation purpose. Hide this to the programmer somehow

Member trisycl::h_item< Dimensions >::h_item ()=default

Make most of them protected, reserved to implementation

Member trisycl::handler::parallel_for (SYCLrange num_work_items, kernel sycl_kernel)

Add in the spec a version taking a kernel and a functor, to have host fall-back

Member trisycl::handler::parallel_for (nd_range< Dimensions >, kernel syclKernel)

Add in the spec a version taking a kernel and a functor, to have host fall-back

To be implemented

Member trisycl::handler::schedule_parallel_for_kernel (Kernel k, const range< N > &num_work_items)

Add host fall-back execution for parallel_for_kernel

Member trisycl::handler::set_arg (int arg_index, accessor< DataType, Dimensions, Mode, Target > &&acc_obj)

Update the specification to use a ref && to the accessor instead?

It is not that clean to have set_arg() associated to a command handler. Rethink the specification?

It seems more logical to have these methods on kernel instead

Member trisycl::handler::set_args (Ts &&... args)

Update the specification to add this function according to https://cvs.khronos.org/bugzilla/show_bug.cgi?id=15978 proposal

Member trisycl::handler::single_task (kernel sycl_kernel)

Add in the spec a version taking a kernel and a functor, to have host fall-back

Class trisycl::image< Dimensions >

implement image

Member trisycl::info::context

Should be unsigned int to be consistent with others?

Member trisycl::info::device

Should be unsigned int?

Member trisycl::info::device_type

there is no accelerator_selector and custom_accelerator

Add opencl to the specification

To be moved in the specification from platform to device

Member trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image2d_max_width, size_t)

To be implemented, return always void.

Member trisycl::item< Dimensions, with_offset >::item ()=default

Make most of them protected, reserved to implementation

Member trisycl::item< Dimensions, with_offset >::set (id< Dimensions > Index)

Move to private and add friends

Class trisycl::kernel

To be implemented

Check specification

Member trisycl::make_ptr (T *pointer)

Implement the case with a plain pointer

Member trisycl::map_allocator

: implement and clarify the specification. It looks like it is not really an allocator according the current spec

Member trisycl::mixin::accessor< T, Dimensions >::get_count () const

Cache it since it is const?

Move these kinds of functions into a mixin between buffers and accessors?

Member trisycl::mixin::accessor< T, Dimensions >::get_range () const

Cache it since it is const?

Member trisycl::mixin::accessor< T, Dimensions >::get_size () const

Cache it since it is const?

rename to something else. In http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/p0122r0.pdf it is named bytes() for example

Member trisycl::nd_item< Dimensions >::get_item () const

Add to the specification

Member trisycl::nd_item< Dimensions >::nd_item ()=default

Make most of them protected, reserved to implementation

Member trisycl::nd_item< Dimensions >::nd_item (id< Dimensions > global_index, nd_range< Dimensions > ndr)

This is for validation purpose. Hide this to the programmer somehow

Member trisycl::nd_item< Dimensions >::nd_item (nd_range< Dimensions > ndr)

This is for the triSYCL implementation which is expected to call set_global() and set_local() later. This should be hidden to the user.

Class trisycl::nd_range< Dimensions >

add copy constructors in the specification

Member trisycl::nd_range< Dimensions >::get_offset () const

get_offset() is lacking in the specification

Class trisycl::non_cl_error

Exceptions are named error in C++

Clean implementation

Add to the specification

Class trisycl::pipe< Name, T, MinCapacity, SizeT >

Bike-shed the naming & API according to current C++ proposals or libraries about static sized FIFO

Member trisycl::pipe< Name, T, MinCapacity, SizeT >::bit_ceil_at_least_2 (std::size_t value)

At some point use C++20 std::bit_ceil instead of this DIY function

Class trisycl::platform

triSYCL Implementation

Member trisycl::platform::get () const

Define a SYCL exception for this

Member trisycl::platform::get_info () const

Add to the specification

Class trisycl::program

To be implemented

Class trisycl::queue

All the queue methods should return a queue& instead of void to it is possible to chain opoerations

The implementation is quite minimal for now. :-)

Member trisycl::queue::queue (const boost::compute::command_queue &q, async_handler ah=nullptr)

Deal with handler

Member trisycl::queue::submit (Handler_Functor cgf)

Add in the spec an implicit conversion of event to queue& so it is possible to chain operations on the queue

Class trisycl::range< Dimensions >

add to the specification some way to specify an offset?

add to the specification this default parameter value?

use std::size_t Dimensions instead of int Dimensions in the specification?

Member trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::pipe_reservation (detail::sycl_2_2::pipe_reservation< accessor_detail > &&pr)

Make it private and add required friends

Namespace trisycl::trisycl

Refactor when updating to latest specification

Member trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::scope_storage

For now it is allocated on the host in this CPU emulation but a device compiler and runtime can create this on a real device

For now it is allocated on the host in this CPU emulation but a device compiler and runtime can create this on a real device

Member trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::scope_storage

For now it is allocated on the host in this CPU emulation but a device compiler and run time can create this on a real device

For now it is allocated on the host in this CPU emulation but a device compiler and run time can create this on a real device

Member trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::scope_storage

For now it is allocated on the host in this CPU emulation but a device compiler and run time can create this on a real device

For now it is allocated on the host in this CPU emulation but a device compiler and run time can create this on a real device

Member trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::write (const axi_packet &v) override

separate debug from shutdown case

separate debug from shutdown case

Clean up the API to separate data from signaling

Clean up the API to separate data from signaling

Member trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::translate_port

This makes sense only for core tile

This makes sense only for core tile

Class trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >

Remove this global infrastructure and distribute it on the tile_infrastructure itself

Remove this global infrastructure and distribute it on the tile_infrastructure itself

Member trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::cascade_stream_pipes [geo↩ ::x_size ∗geo::y_size+1]

Use a data type with 384 bits

Use a data type with 384 bits

Member trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::cascade ()

To remove?

To remove?

Member trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::connect (SrcPort &&src, DstPort &&dst)

Refactor, make the difference between user & physical ports

Refactor, make the difference between user & physical ports

To deprecate and implement in another way?

To deprecate and implement in another way?

Member trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::cs

remove and distribute it instead

remove and distribute it instead

Member trisycl::vendor::xilinx::acap::aie::device< Layout >::cascade ()

To remove?

To remove?

Member trisycl::vendor::xilinx::acap::aie::device< Layout >::connect (SrcPort &&src, DstPort &&dst)

Refactor, make the difference between user & physical ports

Refactor, make the difference between user & physical ports

To deprecate and implement in another way?

To deprecate and implement in another way?

Class trisycl::vendor::xilinx::acap::aie::fifo_channel

factorize out router minion

factorize out router minion

Member trisycl::vendor::xilinx::acap::aie::fifo_channel::capacity

check with hardware team for the value

check with hardware team for the value

Member trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch::nb_master_port

factorize with a mix-in the following between master and slave

factorize with a mix-in the following between master and slave

Member trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch::is_axi_master (master_port_layout)

Factorize out Use a hidden friend to have introspection for master_port_layout

Factorize out Use a hidden friend to have introspection for master_port_layout

Member trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch::latency

factorize the 2 following values with core tiles? Default number of registers on default router paths

factorize the 2 following values with core tiles? Default number of registers on default router paths

Member trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch::m_tile_ctrl_range

Factorize out this range code with core tile

Factorize out this range code with core tile

Member trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch::nb_master_port

factorize with a mix-in the following between master and slave

factorize with a mix-in the following between master and slave

Class trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >

factorize with fifo_port

factorize with fifo_port

Member trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::capacity

check with hardware team for the value

check with hardware team for the value

Member trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::run (Invocable &&f)

Factorize out the 2 run functions

Factorize out the 2 run functions

Class trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >

Make a real queue implementation instead of this mock-up

Make a real queue implementation instead of this mock-up

Member trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::aie_d

Why using a reference since now a device is a shared_ptr?

Why using a reference since now a device is a shared_ptr?

Member trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::submit () const

SUBMIT

SUBMIT

Class trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >

Factorize out the FIFO port

Factorize out the FIFO port

Member trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::try_write (const axi_packet &v) override

think about it again...

think about it again...

Class trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >

Factorize out the FIFO port

Factorize out the FIFO port

Class trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >

Assume only AIE Shim PL tiles for now

Assume only AIE Shim PL tiles for now

Member trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::input (spl p)

Factorize some code between core tile and shim tile

Factorize some code between core tile and shim tile

Member trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::y_coordinate

Remove? Keep the vertical coordinate for debug purpose

Remove? Keep the vertical coordinate for debug purpose

Class trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >

(Andrew Gozillon) FYI, until Intel fix the way they handle base classes in the SYCL compiler frontend this tile_base can never have non-default constructed data (on the device side implementation, the host should be fine to have whatever). As the hack I have in place just now only outlines a call to the base classes default constructor, it might be feasible to work around that though by calling a base class "fill" function in the tile

constructor or tweaking the way the run function of a tile is outlined so that it outlines it as a copy of the full object rather than components of the object and reconstructing it inside the kernel (although you'll fall into the standard layout trap as you'll be passing a full object to the kernel invocation and it'd take some reworking of the Tile's main generation).

(Andrew Gozillon) FYI, until Intel fix the way they handle base classes in the SYCL compiler frontend this tile_base can never have non-default constructed data (on the device side implementation, the host should be fine to have whatever). As the hack I have in place just now only outlines a call to the base classes default constructor, it might be feasible to work around that though by calling a base class "fill" function in the tile constructor or tweaking the way the run function of a tile is outlined so that it outlines it as a copy of the full object rather than components of the object and reconstructing it inside the kernel (although you'll fall into the standard layout trap as you'll be passing a full object to the kernel invocation and it'd take some reworking of the Tile's main generation).

Member trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_west_column ()

Need something more modern than Boost.Operator to implement trisycl::id<p> Need something more modern than Boost.Operator to implement trisycl::id

Member trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::program

can probably be removed

can probably be removed

Member trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::single_task (Work &&f)

In a device implementation we should have a real tile_handler type making sense on the device instead of just ∗this

In a device implementation we should have a real tile_handler type making sense on the device instead of just ∗this

Class trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim >

Deal with multi-dimension array. Now, since we can only deal with 1-dim, PDim is set to 1 by default.

Deal with multi-dimension array. Now, since we can only deal with 1-dim, PDim is set to 1 by default.

Class trisycl::vendor::xilinx::partition::complete< PDim >

Deal with multi-dimension array. Now, since we can only deal with 1-dim, PDim is set to 1 by default.

Deal with multi-dimension array. Now, since we can only deal with 1-dim, PDim is set to 1 by default.

Class trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim >

Deal with multi-dimension array. Now, since we can only deal with 1-dim, PDim is set to 1 by default.

Deal with multi-dimension array. Now, since we can only deal with 1-dim, PDim is set to 1 by default.

Class trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >

Deal with multi-dimension array.

Deal with multi-dimension array.

Member trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_array (const SomeContainer &src)

Find a way to specialize this with a safer implementation when the size of src is at least constexpr

Member trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_array (std::initializer_list< SourceBasicType > l)

Find a way to specialize this with a safer implementation when the size of src is at least constexpr This does not work... static_assert(l.size() == Size);

Member trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_array (const SomeContainer &src)

Find a way to specialize this with a safer implementation when the size of src is at least constexpr

Member trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_array (std::initializer_list< SourceBasicType > l)

Find a way to specialize this with a safer implementation when the size of src is at least constexpr This does not work... static_assert(l.size() == Size);

Member TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE (name, result)

To be implemented

Member TRISYCL_ParallelForKernel_RANGE (N)

Remove either task or q

Think to a cleaner solution

Think to a cleaner solution

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Extensions for Xilinx ACAP Versal . . . 296
AI Engine CGRA . . . 43
Debugging support for ACAP . . . 297
Graphics support for CGRA-like interaction . . . 300
LaTeX support for graphics output . . . 301
Data access and storage in SYCL . . . 302
triSYCL extension for storage scopes . . . 378
Xilinx vendor extensions . . . 418
Data access and storage in old version of SYCL . . . 432
Platforms, contexts, accessors and queues . . . 499
Dealing with OpenCL address spaces . . . 621
Helpers to do array and tuple conversion . . . 645
Some helpers for the implementation . . . 650
Debugging and tracing support . . . 665
Manage default configuration and types . . . 669
Opencl_spir_helpers . . . 672
Device-side runtime implementation . . . 675
Error handling . . . 681
Expressing parallelism through kernels . . . 709
Vector types in SYCL . . . 784

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

cl . . . 823
cl::sycl
The official SYCL 1.2.1 specification exposes the API in the cl::sycl namespace . . . 823
std . . . 823
sycl
Expose the SYCL API directly in the sycl namespace . . . 825
trisycl
The vector type to be used as SYCL vector . . . 825
trisycl::access
Describe the type of access by kernels . . . 840
trisycl::detail
Some global triSYCL configuration . . . 844
trisycl::detail::spir . . . 855
trisycl::detail::sycl_2_2 . . . 855
trisycl::drt
The device-side runtime . . . 855
trisycl::drt::code
The binary code of the kernels . . . 856
trisycl::extension . . . 856
trisycl::extension::ce . . . 857
trisycl::facade . . . 857
trisycl::info . . . 857
trisycl::mixin . . . 861
trisycl::native . . . 861
trisycl::property . . . 861
trisycl::property::queue . . . 862
trisycl::sycl_2_2 . . . 862
trisycl::trisycl . . . 862
trisycl::vendor . . . 862
trisycl::vendor::trisycl . . . 862
trisycl::vendor::trisycl::pipe . . . 863
trisycl::vendor::trisycl::random
Random generators . . . 863
trisycl::vendor::trisycl::scope . . . 863
trisycl::vendor::trisycl::scope::detail . . . 864

trisycl::vendor::xilinx 864
trisycl::vendor::xilinx::acap 866
trisycl::vendor::xilinx::acap::aie 866
trisycl::vendor::xilinx::acap::aie::detail 867
trisycl::vendor::xilinx::acap::aie::layout 868
trisycl::vendor::xilinx::acap::debug 869
trisycl::vendor::xilinx::graphics 869
trisycl::vendor::xilinx::latex 870
trisycl::vendor::xilinx::partition
Kind of array partition 870
trisycl::views 871

Chapter 5

Concept Index

5.1 Concepts

Here is a list of all concepts with brief descriptions:

trisycl::detail::is_small_array
Concept describing a class inheriting from small_array 875

Chapter 6

Hierarchical Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

trisycl::mixin::accessor< T, Dimensions > . . . 499
  trisycl::facade::accessor< mixin::accessor< T, Dimensions > > . . . 499
    trisycl::detail::accessor< T, Dimensions, Mode, access::target::host_buffer > . . . 302
      trisycl::detail::accessor< T, Dimensions, Mode, access::target::local > . . . 302
    trisycl::detail::accessor< T, Dimensions, Mode, Target > . . . 302
trisycl::mixin::accessor< T, 1 > . . . 499
  trisycl::detail::buffer< T, Dimensions > . . . 302
trisycl::detail::accessor_base . . . 302
  trisycl::detail::accessor< T, Dimensions, Mode, access::target::host_buffer > . . . 302
  trisycl::detail::accessor< T, Dimensions, Mode, Target > . . . 302
AccessorMixIn . . . 884
  trisycl::facade::accessor< AccessorMixIn > . . . 499
trisycl::detail::address_space_base< T, AS > . . . 621
  trisycl::detail::address_space_object< T, AS > . . . 621
  trisycl::detail::address_space_variable< T, AS > . . . 621
    trisycl::detail::address_space_array< T, AS > . . . 621
    trisycl::detail::address_space_fundamental< T, AS > . . . 621
      trisycl::detail::address_space_ptr< T, AS > . . . 621
boost::fibers::algo::algorithm
  trisycl::detail::pooled_shared_work . . . 1174
  trisycl::detail::pooled_work_stealing . . . 1181
trisycl::detail::alignment< T > . . . 884
trisycl::detail::alignment<::trisycl::vec< DataType, NumElements > > . . . 885
trisycl::vendor::xilinx::graphics::application . . . 887
Gtk::ApplicationWindow
  trisycl::vendor::xilinx::graphics::frame_grid . . . 1054
    trisycl::vendor::xilinx::graphics::image_grid . . . 1079
    trisycl::vendor::xilinx::graphics::image_grid . . . 1079
  trisycl::vendor::xilinx::graphics::frame_grid . . . 1054
std::array
  trisycl::detail::small_array< DataType, ::trisycl::vec< DataType, NumElements >, NumElements > . . . 650
    trisycl::detail::vec< DataType, 1 > . . . 1302
      trisycl::vec< DataType, 1 > . . . 784

trisycl::detail::vec< DataType, 16 > . . . 1302
trisycl::vec< DataType, 16 > . . . 784
trisycl::detail::vec< DataType, 2 > . . . 1302
trisycl::vec< DataType, 2 > . . . 784
trisycl::detail::vec< DataType, 3 > . . . 1302
trisycl::vec< DataType, 3 > . . . 784
trisycl::detail::vec< DataType, 4 > . . . 1302
trisycl::vec< DataType, 4 > . . . 784
trisycl::detail::vec< DataType, 8 > . . . 1302
trisycl::vec< DataType, 8 > . . . 784
trisycl::detail::vec< DataType, NumElements > . . . 1302
trisycl::detail::small_array< std::size_t, id< 1 >, Dims > . . . 650
trisycl::detail::small_array_sycl< std::size_t, id< 1 >, 1 > . . . 650
trisycl::id< Dimensions > . . . 709
trisycl::detail::small_array< std::size_t, range< 1 >, Dims > . . . 650
trisycl::detail::small_array_sycl< std::size_t, range< 1 >, 1 > . . . 650
trisycl::range< Dimensions > . . . 709
trisycl::detail::small_array< std::size_t, id< Dimensions >, Dims > . . . 650
trisycl::detail::small_array_sycl< std::size_t, id< Dimensions >, Dimensions > . . . 650
trisycl::id< Dimensions > . . . 709
trisycl::id< rank()> . . . 709
trisycl::id< 1 > . . . 709
trisycl::detail::small_array< std::size_t, range< Dimensions >, Dims > . . . 650
trisycl::detail::small_array_sycl< std::size_t, range< Dimensions >, Dimensions > . . . 650
trisycl::range< rank()> . . . 709
trisycl::range< 1 > . . . 709
trisycl::detail::small_array< BasicType, FinalType, Dims > . . . 650
trisycl::detail::small_array_sycl< BasicType, FinalType, Dims > . . . 650
trisycl::vendor::xilinx::acap::aie::axi_packet . . . 43
trisycl::vendor::xilinx::acap::aie::connection::base . . . 895
trisycl::vendor::xilinx::acap::aie::connection::input . . . 1096
trisycl::vendor::xilinx::acap::aie::connection::input . . . 1096
trisycl::vendor::xilinx::acap::aie::connection::output . . . 1142
trisycl::vendor::xilinx::acap::aie::connection::output . . . 1142
trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim > . . . 899
trisycl::detail::bool_pack<... > . . . 901
trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography > . . . 297
trisycl::detail::cache< Key, Value > . . . 901
trisycl::detail::cache< cl_command_queue, trisycl::detail::opencl_queue > . . . 901
trisycl::detail::cache< cl_context, trisycl::detail::opencl_context > . . . 901
trisycl::detail::cache< cl_device_id, trisycl::detail::opencl_device > . . . 901
trisycl::detail::cache< cl_event, trisycl::detail::opencl_event > . . . 901
trisycl::detail::cache< cl_kernel, trisycl::detail::opencl_kernel > . . . 901
trisycl::detail::cache< cl_platform_id, trisycl::detail::opencl_platform > . . . 901
trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography > . . . 43
trisycl::vendor::xilinx::acap::aie::cascade_stream< trisycl::vendor::xilinx::acap::aie::geography > . . . 43
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t . . . 906
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH > . . . 909
trisycl::vendor::xilinx::partition::complete< PDim > . . . 923
trisycl::vendor::xilinx::acap::aie::connection . . . 925
trisycl::detail::container_element_aspect< T > . . . 650
trisycl::detail::container_element_aspect< Accessor::value_type > . . . 650
trisycl::drt::accessor< Accessor > . . . 675
trisycl::detail::container_element_aspect< Buffer::value_type > . . . 650
trisycl::accessor< Buffer::value_type, Buffer::rank(), access::mode::read_write, access::target::host↩
_buffer > . . . 302

trisycl::vendor::xilinx::acap::aie::accessor< Buffer > . . . 877
trisycl::vendor::xilinx::acap::aie::accessor< Buffer > . . . 877
trisycl::detail::container_element_aspect< DataType > . . . 650
trisycl::accessor< DataType, Dimensions, access::mode::read_write, access::target::host_buffer > . . . 302
trisycl::host_accessor< DataType, Dimensions, AccessMode > . . . 432
trisycl::accessor< DataType, Dimensions, AccessMode, Target > . . . 302
trisycl::detail::context . . . 499
trisycl::detail::host_context . . . 499
trisycl::detail::opencl_context . . . 1111
trisycl::vendor::xilinx::latex::context . . . 928
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch . . . 935
trisycl::vendor::trisycl::pipe::cout . . . 302
trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim > . . . 961
trisycl::detail::debug< T > . . . 665
trisycl::detail::debug< accessor< T, Dimensions, Mode, access::target::local > > . . . 665
trisycl::detail::accessor< T, Dimensions, Mode, access::target::local > . . . 302
trisycl::detail::debug< accessor< T, Dimensions, Mode, Target > > . . . 665
trisycl::detail::accessor< T, Dimensions, Mode, access::target::host_buffer > . . . 302
trisycl::detail::accessor< T, Dimensions, Mode, Target > . . . 302
trisycl::detail::debug< axi_stream_switch< axi_ss_geo > > . . . 665
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< axi_ss_geo > . . . 43
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< axi_ss_geo > . . . 43
trisycl::detail::debug< axi_stream_switch< AXIStreamGeography > > . . . 665
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography > . . . 43
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography > . . . 43
trisycl::detail::debug< buffer< T, 1 > > . . . 665
trisycl::detail::buffer< T, Dimensions > . . . 302
detail::debug< buffer< T, 1, buffer_allocator< std::remove_const_t< T > > > > . . . **??**
trisycl::buffer< T, Dimensions, Allocator > . . . 302
detail::debug< buffer_waiter< T, 1, buffer_allocator< std::remove_const_t< T > > > > . . . **??**
trisycl::detail::buffer_waiter< T, Dimensions, Allocator > . . . 302
trisycl::detail::debug< communicator_port > . . . 665
trisycl::vendor::xilinx::acap::aie::communicator_port . . . 916
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion . . . 1226
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion . . . 1226
trisycl::vendor::xilinx::acap::aie::fifo_channel . . . 1048
trisycl::vendor::xilinx::acap::aie::fifo_channel . . . 1048
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich > . . . 1190
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich > . . . 1190
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch > . . . 1218
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch > . . . 1218
trisycl::vendor::xilinx::acap::aie::communicator_port . . . 916
trisycl::detail::debug< detail::event > . . . 665
trisycl::detail::event . . . 1029
trisycl::detail::host_event . . . 1072
trisycl::detail::opencl_event . . . 1126
trisycl::detail::debug< detail::kernel > . . . 665
trisycl::detail::kernel . . . 499
trisycl::detail::opencl_kernel . . . 1131
trisycl::detail::debug< detail::program > . . . 665
trisycl::detail::program . . . 1198
trisycl::detail::debug< detail::queue > . . . 665
trisycl::detail::queue . . . 1210
trisycl::detail::host_queue . . . 1076

trisycl::detail::opencl_queue . . . 1136
trisycl::detail::debug< detail::sycl_2_2::pipe_accessor< DataType, AccessMode, Target > > . . . 665
trisycl::detail::sycl_2_2::pipe_accessor< DataType, AccessMode, access::target::blocking_pipe > . . . 432
trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe > . . . 432
trisycl::detail::sycl_2_2::pipe_accessor< DataType, AccessMode, access::target::pipe > . . . 432
trisycl::accessor< DataType, 1, AccessMode, access::target::pipe > . . . 432
trisycl::detail::debug< detail::sycl_2_2::pipe_accessor< T, AccessMode, Target > > . . . 665
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target > . . . 432
trisycl::detail::debug< detail::sycl_2_2::pipe_reservation< PipeAccessor > > . . . 665
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor > . . . 432
trisycl::detail::debug< handler > . . . 665
trisycl::handler . . . 499
trisycl::detail::debug< host_queue > . . . 665
trisycl::detail::host_queue . . . 1076
trisycl::detail::debug< opencl_kernel > . . . 665
trisycl::detail::opencl_kernel . . . 1131
trisycl::detail::debug< opencl_queue > . . . 665
trisycl::detail::opencl_queue . . . 1136
trisycl::detail::debug< pipe< T > > . . . 665
trisycl::detail::sycl_2_2::pipe< T > . . . 432
trisycl::sycl_2_2::pipe< T > . . . 432
trisycl::detail::debug< pipe< value_type > > . . . 665
trisycl::detail::sycl_2_2::pipe< value_type > . . . 432
trisycl::detail::debug< queue > . . . 665
trisycl::queue . . . 499
trisycl::detail::debug< receiving_dma< AXIStreamSwitch > > . . . 665
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, receiving_dma< AXIStreamSwitch > > . . . 1015
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch > . . . 1218
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch > . . . 1218
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, receiving_dma< AXIStreamSwitch > > . . . 1015
trisycl::detail::debug< sending_dma< AXIStreamSwitch > > . . . 665
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, sending_dma< AXIStreamSwitch > > . . . 1015
trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch > . . . 1241
trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch > . . . 1241
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, sending_dma< AXIStreamSwitch > > . . . 1015
trisycl::detail::debug< SpecializedDMA > . . . 665
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA > . . . 1015
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA > . . . 1015
trisycl::detail::debug< static_pipe< data_type, Capacity > > . . . 665
trisycl::sycl_2_2::static_pipe< data_type, stream_latency > . . . 432
trisycl::detail::debug< static_pipe< int, Capacity > > . . . 665
trisycl::sycl_2_2::static_pipe< int, 4 > . . . 432
trisycl::detail::debug< static_pipe< T, Capacity > > . . . 665
trisycl::sycl_2_2::static_pipe< T, Capacity > . . . 432
trisycl::detail::debug< task > . . . 665
trisycl::detail::task . . . 1276
trisycl::detail::device . . . 499
trisycl::detail::host_device . . . 1068
trisycl::detail::opencl_device . . . 1118
trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform > . . . 378
trisycl::vendor::xilinx::acap::aie::detail::device< Layout > . . . 965
trisycl::device_selector . . . 499

trisycl::device_type_selector . . . 499
trisycl::device_typename_selector< DeviceType > . . . 499
trisycl::detail::display_vector< T > . . . 665
trisycl::detail::display_vector< ::trisycl::vec< DataType, NumElements > > . . . 665
trisycl::detail::small_array< DataType, ::trisycl::vec< DataType, NumElements >, NumElements > . . . 650
trisycl::detail::display_vector< FinalType > . . . 665
trisycl::detail::small_array< BasicType, FinalType, Dims > . . . 650
trisycl::detail::display_vector< id< 1 > > . . . 665
trisycl::detail::small_array< std::size_t, id< 1 >, Dims > . . . 650
trisycl::detail::display_vector< id< Dimensions > > . . . 665
trisycl::detail::small_array< std::size_t, id< Dimensions >, Dims > . . . 650
trisycl::detail::display_vector< range< 1 > > . . . 665
trisycl::detail::small_array< std::size_t, range< 1 >, Dims > . . . 650
trisycl::detail::display_vector< range< Dimensions > > . . . 665
trisycl::detail::small_array< std::size_t, range< Dimensions >, Dims > . . . 650
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef > . . . 43
trisycl::elem . . . 784
trisycl::vendor::trisycl::scope::empty_device_scope . . . 378
trisycl::vendor::trisycl::scope::empty_platform_scope . . . 378
trisycl::vendor::trisycl::scope::empty_queue_scope . . . 378
std::enable_shared_from_this
trisycl::detail::accessor< T, Dimensions, Mode, access::target::host_buffer > . . . 302
trisycl::detail::accessor< T, Dimensions, Mode, Target > . . . 302
trisycl::detail::buffer_base . . . 302
trisycl::detail::buffer< T, Dimensions > . . . 302
trisycl::detail::task . . . 1276
boost::equality_comparable
trisycl::nd_range< Dimensions > . . . 709
trisycl::nd_range< 1 > . . . 709
trisycl::group< Dimensions > . . . 709
trisycl::h_item< Dimensions > . . . 709
trisycl::item< Dimensions, with_offset > . . . 709
trisycl::nd_item< Dimensions > . . . 709
trisycl::nd_range< Dimensions > . . . 709
trisycl::error_handler . . . 681
trisycl::trisycl::default_error_handler . . . 963
boost::euclidean_ring_operators
trisycl::detail::small_array< DataType, ::trisycl::vec< DataType, NumElements >, NumElements > . . . 650
trisycl::detail::small_array< std::size_t, id< 1 >, Dims > . . . 650
trisycl::detail::small_array< std::size_t, range< 1 >, Dims > . . . 650
trisycl::detail::small_array< std::size_t, id< Dimensions >, Dims > . . . 650
trisycl::detail::small_array< std::size_t, range< Dimensions >, Dims > . . . 650
trisycl::detail::small_array< BasicType, FinalType, Dims > . . . 650
trisycl::exception . . . 681
trisycl::async_exception . . . 681
trisycl::cl_exception . . . 681
trisycl::device_error . . . 681
trisycl::compile_program_error . . . 681
trisycl::feature_not_supported . . . 681
trisycl::invalid_object_error . . . 681
trisycl::link_program_error . . . 681
trisycl::memory_allocation_error . . . 681
trisycl::platform_error . . . 681
trisycl::profiling_error . . . 681
trisycl::runtime_error . . . 681
trisycl::accessor_error . . . 681

trisycl::event_error . . . 681
trisycl::invalid_parameter_error . . . 681
trisycl::kernel_error . . . 681
trisycl::nd_range_error . . . 681
trisycl::non_cl_error . . . 681
trisycl::pipe_error . . . 681
trisycl::detail::expand_to_vector< V, Tuple, expansion > . . . 645
trisycl::detail::expand_to_vector< V, Tuple, true > . . . 645
std::false_type
trisycl::detail::is_range< T > . . . 1101
trisycl::is_wrapper< T > . . . 1104
trisycl::detail::fiber_pool . . . 1038
trisycl::vendor::xilinx::acap::aie::layout::full . . . 43
trisycl::vendor::xilinx::acap::aie::layout::size< 50, 8 > . . . 43
trisycl::vendor::xilinx::acap::aie::layout::vc1902 . . . 1301
trisycl::vendor::xilinx::acap::aie::layout::vc1902 . . . 1301
trisycl::vendor::xilinx::acap::aie::layout::size< 17, 3 > . . . 43
trisycl::vendor::xilinx::acap::aie::layout::ve1152 . . . 43
trisycl::vendor::xilinx::acap::aie::layout::ve1152 . . . 43
trisycl::vendor::xilinx::acap::aie::layout::size< 50, 8 > . . . 43
trisycl::vendor::xilinx::acap::aie::layout::size< 17, 3 > . . . 43
trisycl::vendor::xilinx::acap::aie::layout::one_pe . . . 43
trisycl::vendor::xilinx::acap::aie::layout::one_pe . . . 43
trisycl::vendor::xilinx::acap::aie::layout::size< X_size, Y_size > . . . 43
trisycl::vendor::xilinx::acap::aie::layout::size< X_size, Y_size > . . . 43
trisycl::vendor::xilinx::acap::aie::layout::small . . . 43
trisycl::vendor::xilinx::acap::aie::layout::small . . . 43
std::future
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter . . . 1216
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter . . . 1216
trisycl::half . . . 1061
std::hash< trisycl::buffer< T, Dimensions, Allocator > > . . . 1061
std::hash< trisycl::context > . . . 1062
std::hash< trisycl::device > . . . 1063
std::hash< trisycl::event > . . . 1064
std::hash< trisycl::kernel > . . . 1065
std::hash< trisycl::platform > . . . 1066
std::hash< trisycl::queue > . . . 1067
trisycl::image< Dimensions > . . . 302
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input_port . . . 1100
std::integral_constant
std::tuple_size< T > . . . 1299
std::tuple_size< trisycl::vec< T, Size > > . . . 1300
Layout . . . 1105
trisycl::vendor::xilinx::acap::aie::geography< Layout > . . . 43
trisycl::vendor::xilinx::acap::aie::geography< Layout > . . . 43
trisycl::vendor::xilinx::acap::aie::lock_unit . . . 43
trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device . . . 1105
trisycl::vendor::xilinx::acap::aie::memory_base . . . 43
trisycl::vendor::xilinx::acap::aie::memory< AIE, X, Y > . . . 43
trisycl::vendor::xilinx::acap::aie::memory< AIE, X, Y > . . . 43
trisycl::vendor::xilinx::acap::aie::memory_infrastructure . . . 43
trisycl::vendor::xilinx::partition::none . . . 1110
trisycl::detail::ocl_type< T, AS > . . . 621
trisycl::detail::ocl_type< T, access::address_space::constant_space > . . . 621
trisycl::detail::ocl_type< T, access::address_space::global_space > . . . 621
trisycl::detail::ocl_type< T, access::address_space::local_space > . . . 621

trisycl::detail::ocl_type< T, access::address_space::private_space > . . . 621
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::output_port . . . 1146
trisycl::vendor::xilinx::graphics::palette . . . 1147
trisycl::detail::parallel_for_iterate< level, Range, ParallelForFunctor, Id > . . . 709
trisycl::detail::parallel_for_iterate< 0, Range, ParallelForFunctor, Id > . . . 709
trisycl::info::param_traits< T, Param > . . . 1170
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType > . . . 418
trisycl::pipe< Name, T, MinCapacity, SizeT > . . . 302
trisycl::sycl_2_2::pipe_reservation< PipeAccessor > . . . 432
trisycl::detail::platform . . . 499
  trisycl::detail::host_platform . . . 499
  trisycl::detail::opencl_platform . . . 499
trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage > . . . 378
trisycl::detail::pooled_shared_work::pool_ctx . . . 1170
trisycl::detail::pooled_work_stealing::pool_ctx . . . 1172
trisycl::vendor::xilinx::acap::aie::port . . . 1189
trisycl::drt::code::program . . . 1199
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory > . . . 43
trisycl::detail::property . . . 1206
  trisycl::property::queue::enable_profiling . . . 1028
trisycl::property_list . . . 1207
  trisycl::queue . . . 499
trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage > . . . 378
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice > . . . 43
trisycl::detail::sycl_2_2::reserve_id< T > . . . 432
detail::shared_ptr_implementation< buffer< T, 1, buffer_allocator< std::remove_const_t< T > > >, detail↩
::buffer_waiter< T, 1, buffer_allocator< std::remove_const_t< T > > > >
  trisycl::buffer< T, Dimensions, Allocator > . . . 302
detail::shared_ptr_implementation< buffer_waiter< T, 1, buffer_allocator< std::remove_const_t< T > >
  >, detail::buffer< T, 1 > > . . . **??**
  trisycl::detail::buffer_waiter< T, Dimensions, Allocator > . . . 302
trisycl::vendor::xilinx::acap::aie::port::shim . . . 1250
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch . . . 1251
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice > . . . 43
trisycl::vendor::xilinx::acap::aie::shim_tile< trisycl::vendor::xilinx::acap::aie::detail::device > . . . 43
trisycl::vendor::xilinx::acap::aie::axi_packet::shutdown_t . . . 1272
trisycl::detail::singleton< T > . . . 1273
trisycl::detail::singleton< host_context > . . . 1273
  trisycl::detail::host_context . . . 499
trisycl::detail::singleton< host_device > . . . 1273
  trisycl::detail::host_device . . . 1068
trisycl::detail::singleton< host_event > . . . 1273
  trisycl::detail::host_event . . . 1072
trisycl::detail::singleton< host_platform > . . . 1273
  trisycl::detail::host_platform . . . 499
trisycl::detail::small_array_tag . . . 650
  trisycl::detail::small_array< DataType, ::trisycl::vec< DataType, NumElements >, NumElements > . . . 650
  trisycl::detail::small_array< std::size_t, id< 1 >, Dims > . . . 650
  trisycl::detail::small_array< std::size_t, range< 1 >, Dims > . . . 650
  trisycl::detail::small_array< std::size_t, id< Dimensions >, Dims > . . . 650
  trisycl::detail::small_array< std::size_t, range< Dimensions >, Dims > . . . 650
  trisycl::detail::small_array< BasicType, FinalType, Dims > . . . 650
trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, StorageType > . . . 1274
trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, typename ScopedType::storage↩
  _type > . . . 1275
trisycl::vendor::xilinx::acap::aie::port::tile . . . 1293

trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program > 43
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y > 43
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y > 43
trisycl::vendor::xilinx::acap::aie::tile_base< trisycl::vendor::xilinx::acap::aie::program > 43
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography > 43
boost::totally_ordered
trisycl::detail::shared_ptr_implementation< accessor< DataType, Dimensions,(std::is_const_v< DataType > ? access::mode::read :access::mode::read_write), access::target::global_buffer >, detail::accessor< DataType, Dimensions,(std::is_const_v< DataType > ? access::mode::read :access::mode::read_write), access::target::global_buffer > > 1245
trisycl::accessor< DataType, Dimensions, AccessMode, Target > 302
trisycl::detail::shared_ptr_implementation< context, detail::context > 1245
trisycl::context 499
trisycl::detail::shared_ptr_implementation< device, detail::device > 1245
trisycl::facade::device< device, detail::device > 499
trisycl::device 499
trisycl::detail::shared_ptr_implementation< event, detail::event > 1245
trisycl::event 1032
trisycl::detail::shared_ptr_implementation< UserFacing, ImplementationDetail > 1245
trisycl::facade::device< UserFacing, ImplementationDetail > 499
trisycl::detail::shared_ptr_implementation< accessor< DataType, Dimensions, AccessMode, Target >, detail::accessor< DataType, Dimensions, AccessMode, Target > > 1245
trisycl::accessor< DataType, Dimensions, access::mode::read_write, access::target::host_buffer > 302
trisycl::detail::shared_ptr_implementation< kernel, detail::kernel > 1245
trisycl::kernel 499
trisycl::detail::shared_ptr_implementation< platform, detail::platform > 1245
trisycl::platform 499
trisycl::detail::shared_ptr_implementation< program, detail::program > 1245
trisycl::program 1202
trisycl::detail::shared_ptr_implementation< queue, detail::queue > 1245
trisycl::queue 499
trisycl::detail::shared_ptr_implementation< pipe< T >, detail::sycl_2_2::pipe< T > > 1245
trisycl::sycl_2_2::pipe< T > 432
trisycl::detail::shared_ptr_implementation< static_pipe< T, Capacity >, detail::sycl_2_2::pipe< T > >1245
trisycl::sycl_2_2::static_pipe< T, Capacity > 432
trisycl::detail::shared_ptr_implementation< device< empty_device_scope, ::trisycl::vendor::trisycl::scope::empty_platform_scope >, detail::device< empty_device_scope, ::trisycl::vendor::trisycl::scope::empty_platform_scope > > 1245
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform > 378
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform > 378
trisycl::detail::shared_ptr_implementation< platform< empty_platform_scope >, detail::platform< empty_platform_scope > > 1245
trisycl::vendor::trisycl::scope::platform< PlatformStorage > 378
trisycl::vendor::trisycl::scope::platform< PlatformStorage > 378
trisycl::detail::shared_ptr_implementation< queue< Device, empty_queue_scope >, detail::queue< Device, empty_queue_scope > > 1245
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage > 378
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage > 378
trisycl::detail::shared_ptr_implementation< accessor< Buffer::value_type, Dimensions, AccessMode, Target >, detail::accessor< Buffer::value_type, Dimensions, AccessMode, Target > > . . . 1245
trisycl::accessor< Buffer::value_type, Buffer::rank(), access::mode::read_write, access::target::host_buffer > 302
trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > > 1245
trisycl::facade::device< device< Layout >, detail::device< Layout > > 499
trisycl::vendor::xilinx::acap::aie::device< Layout > 990
trisycl::vendor::xilinx::acap::aie::device< Layout > 990

trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_↩
infrastructure< Geography > > . . . 1245
trisycl::facade::device< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography
> > . . . 499
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography > . . . 43
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography > . . . 43
trisycl::detail::shared_ptr_implementation< static_pipe< data_type, Capacity >, detail::sycl_2_2↩
::pipe< data_type > > . . . 1245
trisycl::sycl_2_2::static_pipe< data_type, stream_latency > . . . 432
trisycl::detail::shared_ptr_implementation< static_pipe< int, Capacity >, detail::sycl_2_2::pipe< int >
> . . . 1245
trisycl::sycl_2_2::static_pipe< int, 4 > . . . 432
trisycl::detail::shared_ptr_implementation< tile_infrastructure< trisycl::vendor::xilinx::acap::aie↩
::geography >, detail::tile_infrastructure< trisycl::vendor::xilinx::acap::aie::geography >
> . . . 1245
trisycl::facade::device< tile_infrastructure< trisycl::vendor::xilinx::acap::aie::geography >, detail↩
::tile_infrastructure< trisycl::vendor::xilinx::acap::aie::geography > > . . . 499
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< trisycl::vendor::xilinx::acap::aie::geography
> . . . 43
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< trisycl::vendor::xilinx::acap::aie::geography
> . . . 43
trisycl::detail::shared_ptr_implementation< tile_infrastructure< geo >, detail::tile_infrastructure< geo
> > . . . 1245
trisycl::facade::device< tile_infrastructure< geo >, detail::tile_infrastructure< geo > > . . . 499
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< geo > . . . 43
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< geo > . . . 43
trisycl::detail::shared_ptr_implementation< Parent, Implementation > . . . 1245
trisycl::mixin::accessor< T, Dimensions >::track_index< N > . . . 1294
std::true_type
trisycl::detail::is_range< range< Dimensions > > . . . 1103
std::tuple_element< I, T > . . . 1297
std::tuple_element< I, trisycl::vec< T, Size > > . . . 1298
ocl_type::type
trisycl::detail::address_space_object< T, AS > . . . 621
trisycl::vec< typename, int > . . . 1315
trisycl::vec< int, 2 > . . . 1315
std::vector
trisycl::exception_list . . . 681
trisycl::vendor::trisycl::random::xorshift< Nbits > . . . 1316
bool . . . **??**
friend . . . **??**
shared_ptr< detail::accessor< Buffer::value_type, Dimensions, AccessMode, Target > > . . . **??**
shared_ptr< detail::accessor< DataType, Dimensions, AccessMode, Target > > . . . **??**
shared_ptr< detail::accessor< DataType, Dimensions,(std::is_const_v< DataType > ? access::mode↩
::read :access::mode::read_write), access::target::global_buffer > > . . . **??**
shared_ptr< detail::context > . . . **??**
shared_ptr< detail::device > . . . **??**
shared_ptr< detail::device< empty_device_scope, ::trisycl::vendor::trisycl::scope::empty_platform_scope
> > . . . **??**
shared_ptr< detail::device< Layout > > . . . **??**
shared_ptr< detail::event > . . . **??**
shared_ptr< detail::kernel > . . . **??**
shared_ptr< detail::platform > . . . **??**
shared_ptr< detail::platform< empty_platform_scope > > . . . **??**
shared_ptr< detail::program > . . . **??**
shared_ptr< detail::queue > . . . **??**
shared_ptr< detail::queue< Device, empty_queue_scope > > . . . **??**
shared_ptr< detail::sycl_2_2::pipe< data_type > > . . . **??**

shared_ptr< detail::sycl_2_2::pipe< int > > . . . . . ??
shared_ptr< detail::sycl_2_2::pipe< T > > . . . . . ??
shared_ptr< detail::tile_infrastructure< geo > > . . . . . ??
shared_ptr< detail::tile_infrastructure< Geography > > . . . . . ??
shared_ptr< detail::tile_infrastructure< trisycl::vendor::xilinx::acap::aie::geography > > . . . . . ??
shared_ptr< ImplementationDetail > . . . . . ??
shared_ptr< trisycl::detail::sycl_2_2::pipe< DataType > > . . . . . ??
static constexpr bool . . . . . ??
static constexpr size_t . . . . . ??
trisycl::detail::Ts . . . . . ??
trisycl::detail::overloaded< Ts > 650

Chapter 7

Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

trisycl::vendor::xilinx::acap::aie::accessor< Buffer >
Create a SYCL-like device view of an AI Engine CGRA with some layout . . . 877
AccessorMixIn . . . 884
trisycl::detail::alignment< T > . . . 884
trisycl::detail::alignment<::trisycl::vec< DataType, NumElements > > . . . 885
trisycl::vendor::xilinx::graphics::application
A graphics application running in a separate thread to display images in a grid of tiles . . . 887
trisycl::vendor::xilinx::acap::aie::connection::base
Base class of connection port . . . 895
trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim >
Represent a block partition . . . 899
trisycl::detail::bool_pack<... > . . . 901
trisycl::detail::cache< Key, Value >
A simple thread safe cache mechanism to cache std::shared_ptr of values indexed by keys . . . 901
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t . . . 906
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >
The command group redefining some member functions to be able to pass the scope and so on to the kernel . . . 909
trisycl::vendor::xilinx::acap::aie::communicator_port
Abstract interface for a communication port . . . 916
trisycl::vendor::xilinx::partition::complete< PDim >
Represent a complete partition . . . 923
trisycl::vendor::xilinx::acap::aie::connection
A connection between some AXI Stream Switch . . . 925
trisycl::vendor::xilinx::latex::context . . . 928
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch . . . 935
trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim >
Represent a cyclic partition . . . 961
detail::debug< buffer< T, 1, buffer_allocator< std::remove_const_t< T > > > > . . . ??
detail::debug< buffer_waiter< T, 1, buffer_allocator< std::remove_const_t< T > > > > . . . ??
trisycl::trisycl::default_error_handler . . . 963
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >
Create a SYCL-like device view of an AI Engine CGRA with some layout . . . 965
trisycl::vendor::xilinx::acap::aie::device< Layout >
Create a SYCL-like device view of an AI Engine CGRA with some layout . . . 990

trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >
Generic DMA . . . 1015
trisycl::property::queue::enable_profiling . . . 1028
trisycl::detail::event . . . 1029
trisycl::event
SYCL event . . . 1032
trisycl::detail::fiber_pool . . . 1038
trisycl::vendor::xilinx::acap::aie::fifo_channel
A FIFO channel connection . . . 1048
trisycl::vendor::xilinx::graphics::frame_grid
An application window displaying a grid of tiles . . . 1054
trisycl::half . . . 1061
std::hash< trisycl::buffer< T, Dimensions, Allocator > > . . . 1061
std::hash< trisycl::context > . . . 1062
std::hash< trisycl::device > . . . 1063
std::hash< trisycl::event > . . . 1064
std::hash< trisycl::kernel > . . . 1065
std::hash< trisycl::platform > . . . 1066
std::hash< trisycl::queue > . . . 1067
trisycl::detail::host_device
SYCL host device . . . 1068
trisycl::detail::host_event . . . 1072
trisycl::detail::host_queue
Some implementation details about the SYCL queue . . . 1076
trisycl::vendor::xilinx::graphics::image_grid
An application window displaying a grid of tiled images . . . 1079
trisycl::vendor::xilinx::acap::aie::connection::input
Abstract an input port . . . 1096
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input_port . . . 1100
trisycl::detail::is_range< T >
A type trait to check if a type it a sycl::range or not . . . 1101
trisycl::detail::is_range< range< Dimensions > >
Return true if template instantiating match a sycl::range . . . 1103
trisycl::is_wrapper< T > . . . 1104
Layout . . . 1105
trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device
The individual locking system . . . 1105
trisycl::vendor::xilinx::partition::none
Represent a none partition . . . 1110
trisycl::detail::opencl_context
SYCL OpenCL context . . . 1111
trisycl::detail::opencl_device
SYCL OpenCL device . . . 1118
trisycl::detail::opencl_event . . . 1126
trisycl::detail::opencl_kernel
An abstraction of the OpenCL kernel . . . 1131
trisycl::detail::opencl_queue
Some implementation details about the SYCL queue . . . 1136
trisycl::vendor::xilinx::acap::aie::connection::output
Abstract an output port . . . 1142
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::output_port . . . 1146
trisycl::vendor::xilinx::graphics::palette
Color palette to project a linear data space to an RGB color space . . . 1147
trisycl::info::param_traits< T, Param >
Implement a meta-function from (T, value) to T' to express the return type value of an OpenCL function of kind (T, value) . . . 1170
trisycl::detail::pooled_shared_work::pool_ctx
Shared storage among the working threads . . . 1170

trisycl::detail::pooled_work_stealing::pool_ctx
Shared storage among the working threads . . . 1172
trisycl::detail::pooled_shared_work . . . 1174
trisycl::detail::pooled_work_stealing . . . 1181
trisycl::vendor::xilinx::acap::aie::port
Abstraction of a communication port . . . 1189
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >
A router input port directing to an AIE core input or a BLI input . . . 1190
trisycl::detail::program . . . 1198
trisycl::drt::code::program
A binary program containing some kernels . . . 1199
trisycl::program
SYCL program . . . 1202
trisycl::detail::property . . . 1206
trisycl::property_list
SYCL property list . . . 1207
trisycl::detail::queue
Some implementation details about the SYCL queue . . . 1210
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter
Provide a wait() function that actually can throw an exception instead of ignoring it by the std↩
::future::wait() . . . 1216
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >
Receiving DMA . . . 1218
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion
A router input port with routing skills . . . 1226
trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >
Sending DMA . . . 1241
trisycl::detail::shared_ptr_implementation< Parent, Implementation >
Provide an implementation as shared_ptr with total ordering and hashing to be used with algorithms and in (un)ordered containers . . . 1245
detail::shared_ptr_implementation< buffer_waiter< T, 1, buffer_allocator< std::remove_const_t< T > > >, detail::buffer< T, 1 >
??
trisycl::vendor::xilinx::acap::aie::port::shim
Abstraction of a communication port inside a shim tile . . . 1250
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch . . . 1251
trisycl::vendor::xilinx::acap::aie::axi_packet::shutdown_t
Marker type used to signal network shutdown . . . 1272
trisycl::detail::singleton< T >
Provide a singleton factory . . . 1273
trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, StorageType > . . . 1274
trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, typename ScopedType::storage_type >
1275
trisycl::detail::task
The abstraction to represent SYCL tasks executing inside command_group . . . 1276
trisycl::vendor::xilinx::acap::aie::port::tile
Abstraction of a communication port inside a tile . . . 1293
trisycl::mixin::accessor< T, Dimensions >::track_index< N >
Proxy object to transform an expression like accessor[i1][i2][i3] into the implementation mdp-san(i1,i2,i3) one index at a time . . . 1294
std::tuple_element< I, T >
Each tuple element type is the same, the one of any small_array heir element . . . 1297
std::tuple_element< I, trisycl::vec< T, Size > >
The std::tuple_element specialization for vec . . . 1298
std::tuple_size< T >
Export the dimension as its tuple size . . . 1299
std::tuple_size< trisycl::vec< T, Size > >
The std::tuple_size specialization for vec . . . 1300

trisycl::vendor::xilinx::acap::aie::layout::vc1902
Describe the layout of a Versal VC1902 device 1301
trisycl::detail::vec< DataType, NumElements >
Forward declaration to use in the detail class 1302
trisycl::vec< typename, int >
Forward declaration to use in the detail class 1315
trisycl::vendor::trisycl::random::xorshift< Nbits >
A possible implementation of the xorshift RNGs from 1316

Chapter 8

File Index

8.1 File List

Here is a list of all files with brief descriptions:

include/CL/sycl.hpp
: This is the main SYCL 1.2.1 interoperability header to expose triSYCL into the `cl::sycl` namespace . . . 1325

include/sycl/sycl.hpp
: This is the SYCL extension header to expose triSYCL directly into the sycl namespace as expected by SYCL 2020 . . . 1326

include/sycl/vendor/triSYCL/scope.hpp . . . 1883

include/sycl/vendor/triSYCL/pipe/cout.hpp . . . 1876

include/sycl/vendor/triSYCL/random/xorshift.hpp . . . 1878

include/sycl/vendor/triSYCL/scope/device.hpp . . . 1975

include/sycl/vendor/triSYCL/scope/platform.hpp . . . 1889

include/sycl/vendor/triSYCL/scope/queue.hpp . . . 2126

include/sycl/vendor/triSYCL/scope/detail/util.hpp . . . 1886

include/sycl/vendor/triSYCL/scope/device/detail/device.hpp . . . 1978

include/sycl/vendor/triSYCL/scope/platform/detail/platform.hpp . . . 1892

include/sycl/vendor/triSYCL/scope/queue/detail/queue.hpp . . . 2130

include/sycl/vendor/Xilinx/acap.hpp
: Experimenting with CGRA & processor arrays such as ACAP/AI Engine . . . 1907

include/sycl/vendor/Xilinx/config.hpp
: Control the configuration of some implementation of Xilinx-related features . . . 2222

include/sycl/vendor/Xilinx/fpga.hpp . . . 2226

include/sycl/vendor/Xilinx/graphics.hpp . . . 2241

include/sycl/vendor/Xilinx/latex.hpp . . . 2260

include/sycl/vendor/Xilinx/acap/aie.hpp
: Experimenting with CGRA & processor arrays such as ACAP/AI Engine . . . 1910

include/sycl/vendor/Xilinx/acap/debug.hpp
: Some debugging utilities for ACAP . . . 2208

include/sycl/vendor/Xilinx/acap/aie/accessor.hpp
: SYCL accessor for AIE tile device . . . 1913

include/sycl/vendor/Xilinx/acap/aie/axi_stream_switch.hpp
: Model of an AXI stream switch . . . 1935

include/sycl/vendor/Xilinx/acap/aie/cascade_stream.hpp
: The cascade stream infrastructure between AI Engine tiles . . . 1955

include/sycl/vendor/Xilinx/acap/aie/connection.hpp . . . 1961

include/sycl/vendor/Xilinx/acap/aie/device.hpp
: SYCL-friendly device view of AIE . . . 1979

include/sycl/vendor/Xilinx/acap/aie/dma.hpp
Model of a tile DMA . . . 2020
include/sycl/vendor/Xilinx/acap/aie/geography.hpp
Geographic information about the AI Engine CGRA . . . 2033
include/sycl/vendor/Xilinx/acap/aie/layout.hpp
Some examples of AI Engine array layouts . . . 2057
include/sycl/vendor/Xilinx/acap/aie/lock.hpp
The lock mechanism used by some AI Engine tiles . . . 2086
include/sycl/vendor/Xilinx/acap/aie/memory.hpp
The basic AI Engine heterogeneous Memory Module (i.e . . . 2093
include/sycl/vendor/Xilinx/acap/aie/memory_base.hpp
The basic AI Engine homogeneous Memory Module, with the common infrastructure to all the memory modules (i.e . . . 2100
include/sycl/vendor/Xilinx/acap/aie/memory_infrastructure.hpp
The basic AI Engine homogeneous memory infrastructure, i.e . . . 2106
include/sycl/vendor/Xilinx/acap/aie/program.hpp
Model of an AI Engine program, that weaves the program of each tile with the memory of each tile for a given device . . . 2112
include/sycl/vendor/Xilinx/acap/aie/queue.hpp
SYCL-friendly queue for AIE cooperative program including all the tiles . . . 2132
include/sycl/vendor/Xilinx/acap/aie/shim_tile.hpp
Model of a shim tile between the AI array and the rest of the circuit . . . 2157
include/sycl/vendor/Xilinx/acap/aie/tile.hpp
The basic AI Engine heterogeneous tile that dependent of x & y coordinates but also from the collective program run on it . . . 2164
include/sycl/vendor/Xilinx/acap/aie/tile_base.hpp
The basic AI Engine homogeneous tile, with common content to all the tiles (i.e . . . 2178
include/sycl/vendor/Xilinx/acap/aie/tile_infrastructure.hpp
The basic AI Engine homogeneous tile, with the common infrastructure to all the tiles, i.e . . . 2186
include/sycl/vendor/Xilinx/acap/aie/device/detail/device.hpp
SYCL-friendly device view of AIE . . . 1984
include/sycl/vendor/Xilinx/acap/aie/layout/full.hpp
Layout for a AI Engine array with 5∗6 PE . . . 2061
include/sycl/vendor/Xilinx/acap/aie/layout/one_pe.hpp
Layout for a AI Engine array with 1 PE only . . . 2065
include/sycl/vendor/Xilinx/acap/aie/layout/size.hpp
Flexible layout for a AI Engine array with any 2D size . . . 2070
include/sycl/vendor/Xilinx/acap/aie/layout/small.hpp
Layout for a AI Engine array with a few PE only . . . 2075
include/sycl/vendor/Xilinx/acap/aie/layout/versal.hpp
Layout for some Xilinx Versal products . . . 2080
include/sycl/vendor/Xilinx/acap/aie/tile_infrastructure/detail/tile_infrastructure.hpp
The basic AI Engine homogeneous tile, with the common infrastructure to all the tiles, i.e . . . 2191
include/sycl/vendor/Xilinx/acap/tests/processor_array.cpp . . . 2218
include/sycl/vendor/Xilinx/fpga/opt_decorate_func.hpp . . . 2228
include/sycl/vendor/Xilinx/fpga/partition_array.hpp . . . 2231
include/sycl/vendor/Xilinx/fpga/ssdm_inst.hpp . . . 2239
include/triSYCL/access.hpp . . . 1331
include/triSYCL/accessor.hpp . . . 1915
include/triSYCL/address_space.hpp
Implement OpenCL address spaces in SYCL with C++-style . . . 1337
include/triSYCL/allocator.hpp . . . 1349
include/triSYCL/buffer.hpp . . . 1351
include/triSYCL/buffer_allocator.hpp . . . 1373
include/triSYCL/context.hpp . . . 1380
include/triSYCL/device.hpp . . . 1989
include/triSYCL/device_runtime.hpp . . . 1702
include/triSYCL/device_selector.hpp . . . 1706

include/triSYCL/error_handler.hpp . . . 1711
include/triSYCL/event.hpp . . . 1713
include/triSYCL/exception.hpp . . . 1724
include/triSYCL/group.hpp . . . 1728
include/triSYCL/h_item.hpp . . . 1732
include/triSYCL/half.hpp . . . 1736
include/triSYCL/handler.hpp . . . 1737
include/triSYCL/id.hpp . . . 1745
include/triSYCL/image.hpp
OpenCL SYCL image class . . . 1747
include/triSYCL/item.hpp . . . 1750
include/triSYCL/kernel.hpp . . . 1753
include/triSYCL/math.hpp
Implement a wrapper around OpenCL math operations Joan.Thibault AT ens-rennes POINT fr
This file is distributed under the University of Illinois Open Source License . . . 1763
include/triSYCL/nd_item.hpp . . . 1772
include/triSYCL/nd_range.hpp . . . 1776
include/triSYCL/opencl_types.hpp
TriSYCL wrapper for OpenCL types . . . 1778
include/triSYCL/parallelism.hpp
Implement parallel constructions to launch kernels . . . 1786
include/triSYCL/pipe.hpp . . . 1824
include/triSYCL/platform.hpp . . . 1896
include/triSYCL/program.hpp . . . 2117
include/triSYCL/property_list.hpp . . . 1810
include/triSYCL/queue.hpp . . . 2138
include/triSYCL/range.hpp . . . 1819
include/triSYCL/rounding_mode.hpp
Define rounding modes used in SYCL . . . 1822
include/triSYCL/sycl.hpp . . . 1328
include/triSYCL/vec.hpp
Implement the small vector class . . . 1851
include/triSYCL/accessor/detail/accessor_base.hpp . . . 1332
include/triSYCL/accessor/detail/local_accessor.hpp . . . 1335
include/triSYCL/accessor/facade/accessor.hpp . . . 1922
include/triSYCL/accessor/mixin/accessor.hpp . . . 1925
include/triSYCL/address_space/detail/address_space.hpp
Implement OpenCL address spaces in SYCL with C++-style . . . 1341
include/triSYCL/buffer/detail/accessor.hpp . . . 1929
include/triSYCL/buffer/detail/buffer.hpp . . . 1359
include/triSYCL/buffer/detail/buffer_base.hpp . . . 1365
include/triSYCL/buffer/detail/buffer_waiter.hpp . . . 1370
include/triSYCL/command_group/detail/task.hpp . . . 1375
include/triSYCL/context/detail/context.hpp . . . 1385
include/triSYCL/context/detail/context_tail.hpp . . . 1389
include/triSYCL/context/detail/host_context.hpp . . . 1390
include/triSYCL/context/detail/opencl_context.hpp . . . 1392
include/triSYCL/detail/alignment_helper.hpp
A simple trait to help generate the appropriate alignment data for a SYCL type when specified appropriately . . . 1395
include/triSYCL/detail/all_true.hpp . . . 1398
include/triSYCL/detail/array_tuple_helpers.hpp
Some helpers to do array-tuple conversions . . . 1399
include/triSYCL/detail/cache.hpp . . . 1402
include/triSYCL/detail/container_element_aspect.hpp . . . 1405
include/triSYCL/detail/debug.hpp . . . 2210
include/triSYCL/detail/default_classes.hpp . . . 1406

include/triSYCL/detail/enum.hpp
Some enum-related helpers . . . 1409
include/triSYCL/detail/fiber_pool.hpp
A Boost.Fiber pool implemented on a std::thread pool . . . 1419
include/triSYCL/detail/global_config.hpp . . . 1423
include/triSYCL/detail/instantiate_kernel.hpp . . . 1427
include/triSYCL/detail/linear_id.hpp . . . 1430
include/triSYCL/detail/metaprogramming.hpp . . . 1431
include/triSYCL/detail/overloaded.hpp . . . 1433
include/triSYCL/detail/property.hpp . . . 1434
include/triSYCL/detail/ranges.hpp
Some range-related helpers . . . 1435
include/triSYCL/detail/shared_ptr_implementation.hpp . . . 1438
include/triSYCL/detail/singleton.hpp . . . 1440
include/triSYCL/detail/small_array.hpp . . . 1442
include/triSYCL/detail/unimplemented.hpp . . . 1689
include/triSYCL/detail/fiber/pooled_shared_work.hpp . . . 1411
include/triSYCL/detail/fiber/pooled_work_stealing.hpp . . . 1415
include/triSYCL/detail/SPIR/opencl_spir.h . . . 1449
include/triSYCL/detail/SPIR/opencl_spir_helpers.hpp
Adds helper functions to populate Id's and Range's with OpenCL SPIR intrinsics data: e.g . . . 1670
include/triSYCL/detail/SPIR/opencl_spir_req.h . . . 1673
include/triSYCL/device/detail/device.hpp . . . 1994
include/triSYCL/device/detail/device_tail.hpp . . . 1691
include/triSYCL/device/detail/host_device.hpp . . . 1692
include/triSYCL/device/detail/opencl_device.hpp . . . 1696
include/triSYCL/device/detail/opencl_device_tail.hpp . . . 1699
include/triSYCL/device/facade/device.hpp . . . 1997
include/triSYCL/device_selector/detail/device_selector_tail.hpp . . . 1708
include/triSYCL/event/detail/event.hpp . . . 1717
include/triSYCL/event/detail/host_event.hpp . . . 1720
include/triSYCL/event/detail/opencl_event.hpp . . . 1722
include/triSYCL/info/context.hpp . . . 1387
include/triSYCL/info/device.hpp . . . 1998
include/triSYCL/info/event.hpp . . . 1718
include/triSYCL/info/param_traits.hpp . . . 1748
include/triSYCL/info/platform.hpp . . . 1893
include/triSYCL/info/queue.hpp . . . 2135
include/triSYCL/kernel/detail/kernel.hpp . . . 1756
include/triSYCL/kernel/detail/opencl_kernel.hpp . . . 1759
include/triSYCL/parallelism/detail/parallelism.hpp
Implement the detail of the parallel constructions to launch kernels . . . 1788
include/triSYCL/parallelism/detail/parallelism_tbb.hpp
Implement the detail of the parallel constructions to launch kernels using Intel Threading Building Blocks (TBB) . . . 1796
include/triSYCL/platform/detail/host_platform.hpp . . . 1800
include/triSYCL/platform/detail/host_platform_tail.hpp . . . 1803
include/triSYCL/platform/detail/opencl_platform.hpp . . . 1804
include/triSYCL/platform/detail/opencl_platform_tail.hpp . . . 1808
include/triSYCL/platform/detail/platform.hpp . . . 1900
include/triSYCL/platform/detail/platform_tail.hpp . . . 1809
include/triSYCL/program/detail/program.hpp . . . 2120
include/triSYCL/property/queue.hpp . . . 2137
include/triSYCL/queue/detail/host_queue.hpp . . . 1813
include/triSYCL/queue/detail/opencl_queue.hpp . . . 1815
include/triSYCL/queue/detail/queue.hpp . . . 2144
include/triSYCL/queue/detail/queue_tail.hpp . . . 1818
include/triSYCL/sycl_2_2/pipe.hpp . . . 1826

include/triSYCL/sycl_2_2/pipe_reservation.hpp . . . 1841
include/triSYCL/sycl_2_2/static_pipe.hpp . . . 1848
include/triSYCL/sycl_2_2/pipe/detail/pipe.hpp . . . 1829
include/triSYCL/sycl_2_2/pipe/detail/pipe_accessor.hpp . . . 1836
include/triSYCL/sycl_2_2/pipe_reservation/detail/pipe_reservation.hpp . . . 1844
include/triSYCL/vec/detail/swiz2.hpp . . . 1866
include/triSYCL/vec/detail/swiz3.hpp . . . 1867
include/triSYCL/vec/detail/swiz4.hpp . . . 1868
include/triSYCL/vec/detail/swiz_rgba.hpp . . . 1872
include/triSYCL/vec/detail/vec.hpp
Implement the detail base SYCL vector class . . . 1860
include/triSYCL/vendor/triSYCL/scope.hpp . . . 1885
include/triSYCL/vendor/triSYCL/pipe/cout.hpp . . . 1877
include/triSYCL/vendor/triSYCL/random/xorshift.hpp . . . 1881
include/triSYCL/vendor/triSYCL/scope/device.hpp . . . 2004
include/triSYCL/vendor/triSYCL/scope/platform.hpp . . . 1903
include/triSYCL/vendor/triSYCL/scope/queue.hpp . . . 2147
include/triSYCL/vendor/triSYCL/scope/detail/util.hpp . . . 1888
include/triSYCL/vendor/triSYCL/scope/device/detail/device.hpp . . . 2008
include/triSYCL/vendor/triSYCL/scope/platform/detail/platform.hpp . . . 1906
include/triSYCL/vendor/triSYCL/scope/queue/detail/queue.hpp . . . 2151
include/triSYCL/vendor/Xilinx/acap.hpp
Experimenting with CGRA & processor arrays such as ACAP/AI Engine . . . 1908
include/triSYCL/vendor/Xilinx/config.hpp
Control the configuration of some implementation of Xilinx-related features . . . 2223
include/triSYCL/vendor/Xilinx/fpga.hpp . . . 2226
include/triSYCL/vendor/Xilinx/graphics.hpp . . . 2251
include/triSYCL/vendor/Xilinx/latex.hpp . . . 2263
include/triSYCL/vendor/Xilinx/acap/aie.hpp
Experimenting with CGRA & processor arrays such as ACAP/AI Engine . . . 1911
include/triSYCL/vendor/Xilinx/acap/debug.hpp
Some debugging utilities for ACAP . . . 2215
include/triSYCL/vendor/Xilinx/acap/aie/accessor.hpp
SYCL accessor for AIE tile device . . . 1933
include/triSYCL/vendor/Xilinx/acap/aie/axi_stream_switch.hpp
Model of an AXI stream switch . . . 1945
include/triSYCL/vendor/Xilinx/acap/aie/cascade_stream.hpp
The cascade stream infrastructure between AI Engine tiles . . . 1958
include/triSYCL/vendor/Xilinx/acap/aie/connection.hpp . . . 1968
include/triSYCL/vendor/Xilinx/acap/aie/device.hpp
SYCL-friendly device view of AIE . . . 2010
include/triSYCL/vendor/Xilinx/acap/aie/dma.hpp
Model of a tile DMA . . . 2027
include/triSYCL/vendor/Xilinx/acap/aie/geography.hpp
Geographic information about the AI Engine CGRA . . . 2045
include/triSYCL/vendor/Xilinx/acap/aie/layout.hpp
Some examples of AI Engine array layouts . . . 2059
include/triSYCL/vendor/Xilinx/acap/aie/lock.hpp
The lock mechanism used by some AI Engine tiles . . . 2089
include/triSYCL/vendor/Xilinx/acap/aie/memory.hpp
The basic AI Engine heterogeneous Memory Module (i.e . . . 2097
include/triSYCL/vendor/Xilinx/acap/aie/memory_base.hpp
The basic AI Engine homogeneous Memory Module, with the common infrastructure to all the memory modules (i.e . . . 2103
include/triSYCL/vendor/Xilinx/acap/aie/memory_infrastructure.hpp
The basic AI Engine homogeneous memory infrastructure, i.e . . . 2109

include/triSYCL/vendor/Xilinx/acap/aie/program.hpp
Model of an AI Engine program, that weaves the program of each tile with the memory of each tile for a given device . . . 2121
include/triSYCL/vendor/Xilinx/acap/aie/queue.hpp
SYCL-friendly queue for AIE cooperative program including all the tiles . . . 2153
include/triSYCL/vendor/Xilinx/acap/aie/shim_tile.hpp
Model of a shim tile between the AI array and the rest of the circuit . . . 2160
include/triSYCL/vendor/Xilinx/acap/aie/tile.hpp
The basic AI Engine heterogeneous tile that dependent of x & y coordinates but also from the collective program run on it . . . 2171
include/triSYCL/vendor/Xilinx/acap/aie/tile_base.hpp
The basic AI Engine homogeneous tile, with common content to all the tiles (i.e . . . 2181
include/triSYCL/vendor/Xilinx/acap/aie/tile_infrastructure.hpp
The basic AI Engine homogeneous tile, with the common infrastructure to all the tiles, i.e . . . 2197
include/triSYCL/vendor/Xilinx/acap/aie/device/detail/device.hpp
SYCL-friendly device view of AIE . . . 2015
include/triSYCL/vendor/Xilinx/acap/aie/layout/full.hpp
Layout for a AI Engine array with 5∗6 PE . . . 2063
include/triSYCL/vendor/Xilinx/acap/aie/layout/one_pe.hpp
Layout for a AI Engine array with 1 PE only . . . 2068
include/triSYCL/vendor/Xilinx/acap/aie/layout/size.hpp
Flexible layout for a AI Engine array with any 2D size . . . 2073
include/triSYCL/vendor/Xilinx/acap/aie/layout/small.hpp
Layout for a AI Engine array with a few PE only . . . 2078
include/triSYCL/vendor/Xilinx/acap/aie/layout/versal.hpp
Layout for some Xilinx Versal products . . . 2083
include/triSYCL/vendor/Xilinx/acap/aie/tile_infrastructure/detail/tile_infrastructure.hpp
The basic AI Engine homogeneous tile, with the common infrastructure to all the tiles, i.e . . . 2202
include/triSYCL/vendor/Xilinx/acap/tests/processor_array.cpp . . . 2220
include/triSYCL/vendor/Xilinx/fpga/opt_decorate_func.hpp . . . 2229
include/triSYCL/vendor/Xilinx/fpga/partition_array.hpp . . . 2235
include/triSYCL/vendor/Xilinx/fpga/ssdm_inst.hpp . . . 2240

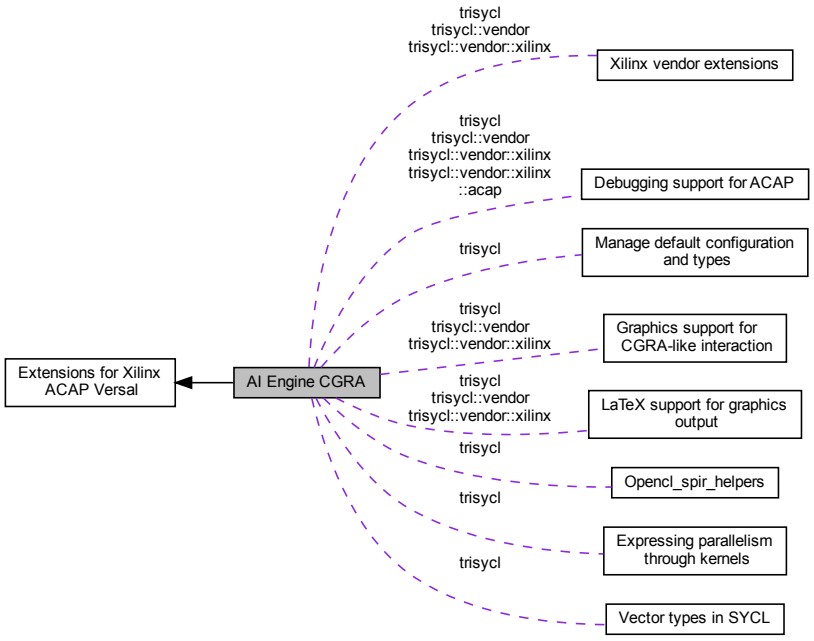
Chapter 9

Module Documentation

9.1 AI Engine CGRA

Extensions to support explicit AI Engine system-wide programming in C++.

Collaboration diagram for AI Engine CGRA:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::detail
- namespace trisycl::vendor::xilinx::acap::aie::layout

Classes

- class trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >

  An AXI stream switch parameterized with some geographic layout. More...
- struct trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >

  Implementation of the cascade stream infrastructure between AI Engine tiles. More...
- class trisycl::vendor::xilinx::acap::aie::axi_packet
- class trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >

  Implement a small DSL to chain DMA & locking operations like allowed by the hardware. More...
- struct trisycl::vendor::xilinx::acap::aie::geography< Layout >

  Some geographic information about the AI Engine array. More...
- struct trisycl::vendor::xilinx::acap::aie::layout::full

  Describe the layout of a "big" AI Engine array. More...
- struct trisycl::vendor::xilinx::acap::aie::layout::one_pe

  Describe the layout of a tiny AI Engine with only 1 tile. More...
- struct trisycl::vendor::xilinx::acap::aie::layout::size< X_size, Y_size >

  Fully specify the size of an AI Engine layout. More...
- struct trisycl::vendor::xilinx::acap::aie::layout::small

  Describe the layout of a small AI Engine array with only a few PE. More...
- struct trisycl::vendor::xilinx::acap::aie::layout::ve1152
- struct trisycl::vendor::xilinx::acap::aie::lock_unit

  The lock infrastructure used by AI Engine memory modules and shim tiles. More...
- struct trisycl::vendor::xilinx::acap::aie::memory< AIE, X, Y >

  The AI Engine Memory Module infrastructure of a tile. More...
- struct trisycl::vendor::xilinx::acap::aie::memory_base

  The AI Engine Memory Module infrastructure of a tile common to all the tiles. More...
- class trisycl::vendor::xilinx::acap::aie::memory_infrastructure

  The AI Engine memory infrastructure common to all the tiles. More...
- struct trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >

  Define an AI Engine CGRA program with its code and memory per core. More...
- struct trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >

  The AI Engine minimal queue. More...
- class trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >

  The AI Engine shim tile infrastructure. More...
- struct trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >

  The AI Engine tile infrastructure defining the program of a tile. More...
- class trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >

  The AI Engine tile infrastructure common to all the tiles. More...
- class trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >

  The AI Engine tile infrastructure common to all the tiles. More...
- class trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >

  The AI Engine tile infrastructure common to all the tiles. More...

9.1.1 Detailed Description

Extensions to support explicit AI Engine system-wide programming in C++.

9.1.2 Class Documentation

9.1.2.1 class trisycl::vendor::xilinx::acap::aie::axi_stream_switch

template<typename AXIStreamGeography>
class trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >

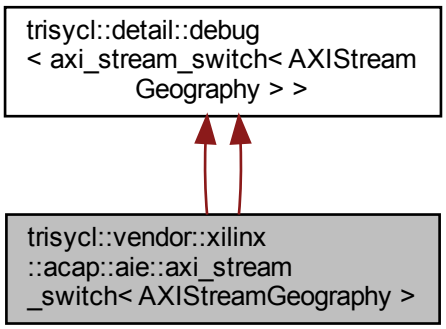
An AXI stream switch parameterized with some geographic layout.

Parameters

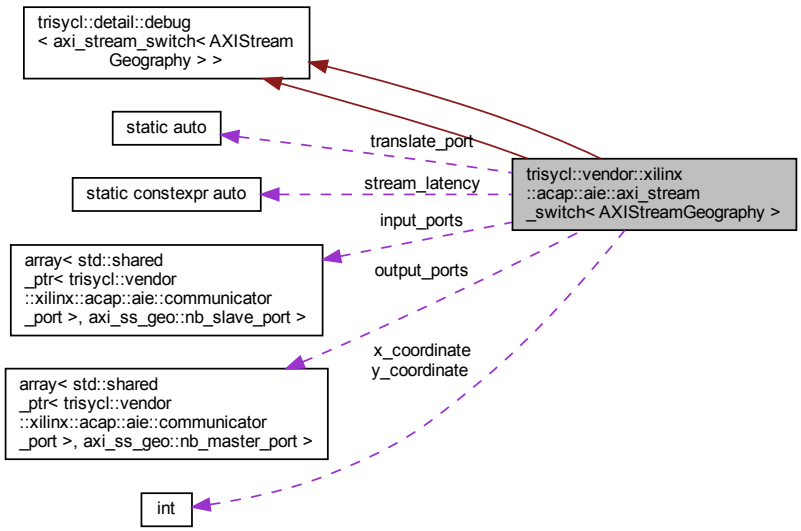
in	*AxiStreamGeography*	is the geography class used to instantiate the AXI stream switch.

Definition at line 51 of file axi_stream_switch.hpp.

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >:



Public Types

- using axi_ss_geo = AXIStreamGeography
- using mpl = typename axi_ss_geo::master_port_layout
- using spl = typename axi_ss_geo::slave_port_layout
- using data_type = std::uint32_t
- using axi_ss_geo = AXIStreamGeography
- using mpl = typename axi_ss_geo::master_port_layout
- using spl = typename axi_ss_geo::slave_port_layout
- using data_type = std::uint32_t

Public Member Functions

- auto & in_connection (spl sp)

  Access the input connection behind an input port.
- auto & out_connection (mpl mp)

  Access the output connection behind an output port.
- void connect (spl sp, mpl mp)

  Configure a connection of the shim AXI stream switch.
- auto & input (spl p)

  Get the input router port of the AXI stream switch.
- auto & output (mpl p)

  Get the input router port of the AXI stream switch.
- axi_stream_switch ()

  Connect the internals of the AXI stream switch.
- void start (int x, int y, ::trisycl::detail::fiber_pool &fiber_executor)

  Start the tile infrastructure associated to the AIE device.
- int tikz_edge_angle (const std::string_view &port_name, const std::initializer_list< std::tuple< std::string_↩ view, int > > &desc) const

Compute the edge angle for a port name according a list of description associating a port name prefix and an associated angle in degree.

- void display_routing_configuration (latex::context &c) const

  Display the internal switch connections with routing arrows perpendicular to the borders.

- template<typename FG , typename FL , typename... EnumRanges>
  auto display_border (const std::string_view &node_attribute, FG &&global_coordinate_function, FL &&local↩
  _coordinate_function, EnumRanges &&... enum_ranges) const

  Display a border of the AXI stream switch.

- template<typename GTC >
  void display (latex::context &c, const vec< int, 2 > &core_size, const GTC &get_tikz_coordinate) const

  Display the AXI stream switch.

- auto & in_connection (spl sp)

  Access the input connection behind an input port.

- auto & out_connection (mpl mp)

  Access the output connection behind an output port.

- void connect (spl sp, mpl mp)

  Configure a connection of the shim AXI stream switch.

- auto & input (spl p)

  Get the input router port of the AXI stream switch.

- auto & output (mpl p)

  Get the input router port of the AXI stream switch.

- axi_stream_switch ()

  Connect the internals of the AXI stream switch.

- void start (int x, int y, ::trisycl::detail::fiber_pool &fiber_executor)

  Start the tile infrastructure associated to the AIE device.

- int tikz_edge_angle (const std::string_view &port_name, const std::initializer_list< std::tuple< std::string_↩
  view, int > > &desc) const

  Compute the edge angle for a port name according a list of description associating a port name prefix and an associated angle in degree.

- void display_routing_configuration (latex::context &c) const

  Display the internal switch connections with routing arrows perpendicular to the borders.

- template<typename FG , typename FL , typename... EnumRanges>
  auto display_border (const std::string_view &node_attribute, FG &&global_coordinate_function, FL &&local↩
  _coordinate_function, EnumRanges &&... enum_ranges) const

  Display a border of the AXI stream switch.

- template<typename GTC >
  void display (latex::context &c, const vec< int, 2 > &core_size, const GTC &get_tikz_coordinate) const

  Display the AXI stream switch.

Static Public Member Functions

- static router_minion * get_as_router_minion (communicator_port *rp)

  Get the router_minion behind a communicator_port.

- static vec< int, 2 > display_size ()

  Compute the size of the graphics representation of the switch.

- static router_minion * get_as_router_minion (communicator_port *rp)

  Get the router_minion behind a communicator_port.

- static vec< int, 2 > display_size ()

  Compute the size of the graphics representation of the switch.

Public Attributes

- std::array< std::shared_ptr< communicator_port >, axi_ss_geo::nb_slave_port > input_ports

  The input ports used to send information to the switch.
- std::array< std::shared_ptr< communicator_port >, axi_ss_geo::nb_master_port > output_ports

  The input ports used to send information to the switch.
- int x_coordinate

  Keep the horizontal coordinate for debug purpose.
- int y_coordinate

  Keep the vertical coordinate for debug purpose.

Static Public Attributes

- static auto translate_port

  Validate the user port number and translate it to the physical port number.
- static constexpr auto stream_latency = 4

Additional Inherited Members

9.1.2.1.1 Member Typedef Documentation

9.1.2.1.1.1 axi_ss_geo [1/2] template<typename AXIStreamGeography >

```
using trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::axi_ss_geo
= AXIStreamGeography
```

Definition at line 56 of file axi_stream_switch.hpp.

9.1.2.1.1.2 axi_ss_geo [2/2] template<typename AXIStreamGeography >

```
using trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::axi_ss_geo
= AXIStreamGeography
```

Definition at line 56 of file axi_stream_switch.hpp.

9.1.2.1.1.3 data_type [1/2] template<typename AXIStreamGeography >

```
using trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::data_type =
std::uint32_t
```

Definition at line 334 of file axi_stream_switch.hpp.

9.1.2.1.1.4 data_type [2/2] template<typename AXIStreamGeography >

```
using trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::data_type =
std::uint32_t
```

Definition at line 334 of file axi_stream_switch.hpp.

9.1.2.1.1.5 mpl [1/2] template<typename AXIStreamGeography >

```
using trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::mpl =
typename axi_ss_geo::master_port_layout
```

Definition at line 57 of file axi_stream_switch.hpp.

9.1.2.1.1.6 mpl [2/2] template<typename AXIStreamGeography >

```
using trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::mpl =
typename axi_ss_geo::master_port_layout
```

Definition at line 57 of file axi_stream_switch.hpp.

9.1.2.1.1.7 spl [1/2] template<typename AXIStreamGeography >

```
using trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::spl =
typename axi_ss_geo::slave_port_layout
```

Definition at line 58 of file axi_stream_switch.hpp.

9.1.2.1.1.8 spl [2/2] template<typename AXIStreamGeography >

```
using trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::spl =
typename axi_ss_geo::slave_port_layout
```

Definition at line 58 of file axi_stream_switch.hpp.

9.1.2.1.2 Constructor & Destructor Documentation

9.1.2.1.2.1 axi_stream_switch() [1/2] template<typename AXIStreamGeography >

```
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::axi_stream_switch
( )  [inline]
```

Connect the internals of the AXI stream switch.

Definition at line 412 of file axi_stream_switch.hpp.

```
00412                     {
00413     // \todo DMA
00414
00415     // \todo CTRL
00416   }
```

9.1.2.1.2.2 axi_stream_switch() [2/2] template<typename AXIStreamGeography >

```
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::axi_stream_switch
( )  [inline]
```

Connect the internals of the AXI stream switch.

Definition at line 412 of file axi_stream_switch.hpp.

```
00412                     {
00413     // \todo DMA
00414
00415     // \todo CTRL
00416   }
```

9.1.2.1.3 Member Function Documentation

9.1.2.1.3.1 connect() [1/2] template<typename AXIStreamGeography >

```
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::connect (
            spl sp,
            mpl mp )  [inline]
```

Configure a connection of the shim AXI stream switch.

Parameters

in	*sp*	is the input port number
in	*mp*	is the output port number

Exceptions

trisycl::runtime_error	if the port number is invalid

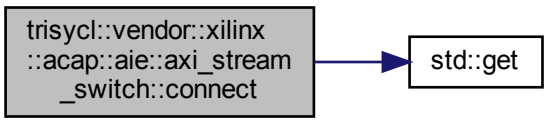
Definition at line 325 of file axi_stream_switch.hpp.

```
00325                                {
00326     TRISYCL_DUMP_T("connect " « this
00327                    « " input:  " « magic_enum::enum_name(sp)
00328                    « " to output:  " « magic_enum::enum_name(mp));
00329     get_as_router_minion(in_connection(sp).get())
00330       ->connect_to(mp, out_connection(mp));
00331   }
```

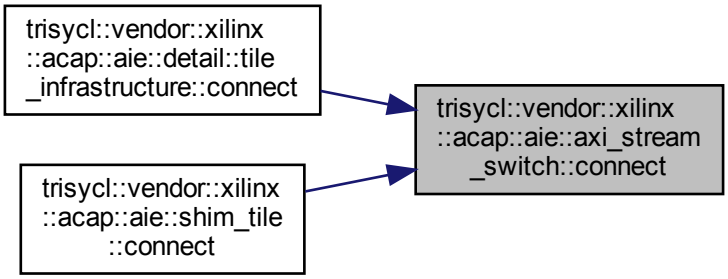
References std::get(), and TRISYCL_DUMP_T.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::connect(), and trisycl::vendor::xilinx::acap::ai

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.1.3.2 connect() [2/2] template<typename AXIStreamGeography >

```
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::connect (
            spl sp,
            mpl mp )  [inline]
```

Configure a connection of the shim AXI stream switch.

Parameters

in	*sp*	is the input port number
in	*mp*	is the output port number

Exceptions

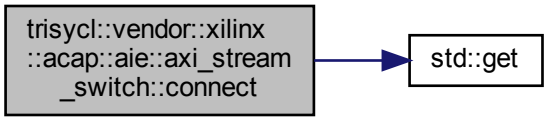
trisycl::runtime_error	if the port number is invalid

Definition at line 325 of file axi_stream_switch.hpp.

```
00325                                 {
00326     TRISYCL_DUMP_T("connect " « this
00327                    « " input:  " « magic_enum::enum_name(sp)
00328                    « " to output:  " « magic_enum::enum_name(mp));
00329     get_as_router_minion(in_connection(sp).get())
00330       ->connect_to(mp, out_connection(mp));
00331   }
```

References std::get(), and TRISYCL_DUMP_T.

Here is the call graph for this function:



9.1.2.1.3.3 display() [1/2]

```
template<typename AXIStreamGeography >
template<typename GTC >
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::display (
            latex::context & c,
            const vec< int, 2 > & core_size,
            const GTC & get_tikz_coordinate ) const  [inline]
```

Display the AXI stream switch.

Definition at line 567 of file axi_stream_switch.hpp.

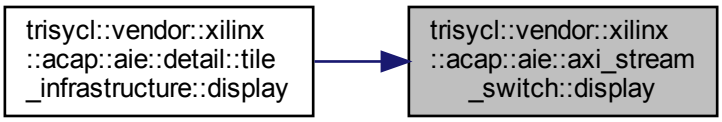
```
00573                                                                {
00574     const auto [ width, height ] = display_size();
00575     // Display the network interfaces on the top of the switch
00576     c.add(display_border
00577           ("rotate=90,anchor=north west", get_tikz_coordinate,
00578            [&, height = height] (auto i) {
00579              return std::tuple
00580                { core_size.x()
00581                    + ranges::distance(axi_ss_geo::s_me_range)
00582                    + ranges::distance(axi_ss_geo::s_dma_range)
00583                    + ranges::distance(axi_ss_geo::s_tile_ctrl_range)
00584                    + ranges::distance(axi_ss_geo::s_fifo_range) + i,
00585                  core_size.y() + height - 1 };
00586            },
00587            axi_ss_geo::m_north_range,
00588            axi_ss_geo::s_north_range));
00589     // Display the network interfaces on the bottom of the switch
00590     c.add(display_border("rotate=90,anchor=north east",
00591                          get_tikz_coordinate,
00592                          [&] (auto i) {
00593                            return std::tuple { core_size.x() + i,
00594                                                core_size.y() };
00595                          },
00596                          ranges::views::reverse(axi_ss_geo::s_me_range),
00597                          axi_ss_geo::s_dma_range,
00598                          axi_ss_geo::s_tile_ctrl_range,
00599                          axi_ss_geo::s_fifo_range,
00600                          axi_ss_geo::s_south_range,
00601                          axi_ss_geo::m_south_range));
00602     // Display the network interfaces on the left of the switch
00603     c.add(display_border("anchor=south east",
00604                          get_tikz_coordinate,
00605                          [&] (auto i) {
00606                            return std::tuple { core_size.x() - 1,
00607                                                core_size.y() + i + 1 };
00608                          },
00609                          ranges::views::reverse(axi_ss_geo::m_me_range),
00610                          axi_ss_geo::m_dma_range,
00611                          axi_ss_geo::m_tile_ctrl_range,
00612                          axi_ss_geo::m_fifo_range,
00613                          axi_ss_geo::m_west_range,
00614                          axi_ss_geo::s_west_range));
00615     // Display the network interfaces on the right of the switch
00616     c.add(display_border
00617           ("anchor=south west",
```

```
00618             get_tikz_coordinate,
00619             [&, width = width] (auto i) {
00620               return std::tuple
00621                 { core_size.x() + width - 1,
00622                    core_size.y() + i + 1
00623                    + ranges::distance(axi_ss_geo::m_me_range)
00624                    + ranges::distance(axi_ss_geo::m_dma_range)
00625                    + ranges::distance(axi_ss_geo::m_tile_ctrl_range)
00626                    + ranges::distance(axi_ss_geo::m_fifo_range) };
00627             },
00628             axi_ss_geo::s_east_range,
```

References c, and i.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display().

Here is the caller graph for this function:



9.1.2.1.3.4 display() [2/2]

```
template<typename AXIStreamGeography >
template<typename GTC >
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::display (
              latex::context & c,
              const vec< int, 2 > & core_size,
              const GTC & get_tikz_coordinate ) const  [inline]
```

Display the AXI stream switch.

Definition at line 567 of file axi_stream_switch.hpp.

```
00573                                                 {
00574     const auto [ width, height ] = display_size();
00575     // Display the network interfaces on the top of the switch
00576     c.add(display_border
00577           ("rotate=90,anchor=north west", get_tikz_coordinate,
00578            [&, height = height] (auto i) {
00579              return std::tuple
00580                { core_size.x()
00581                    + ranges::distance(axi_ss_geo::s_me_range)
00582                    + ranges::distance(axi_ss_geo::s_dma_range)
00583                    + ranges::distance(axi_ss_geo::s_tile_ctrl_range)
00584                    + ranges::distance(axi_ss_geo::s_fifo_range) + i,
00585                   core_size.y() + height - 1 };
00586            },
00587            axi_ss_geo::m_north_range,
00588            axi_ss_geo::s_north_range));
00589     // Display the network interfaces on the bottom of the switch
00590     c.add(display_border("rotate=90,anchor=north east",
00591                          get_tikz_coordinate,
00592                          [&] (auto i) {
00593                            return std::tuple { core_size.x() + i,
00594                                                core_size.y() };
00595                          },
00596                          ranges::views::reverse(axi_ss_geo::s_me_range),
00597                          axi_ss_geo::s_dma_range,
00598                          axi_ss_geo::s_tile_ctrl_range,
```

```
00599                         axi_ss_geo::s_fifo_range,
00600                         axi_ss_geo::s_south_range,
00601                         axi_ss_geo::m_south_range));
00602     // Display the network interfaces on the left of the switch
00603     c.add(display_border("anchor=south east",
00604                         get_tikz_coordinate,
00605                         [&] (auto i) {
00606                           return std::tuple { core_size.x() - 1,
00607                                               core_size.y() + i + 1 };
00608                         },
00609                         ranges::views::reverse(axi_ss_geo::m_me_range),
00610                         axi_ss_geo::m_dma_range,
00611                         axi_ss_geo::m_tile_ctrl_range,
00612                         axi_ss_geo::m_fifo_range,
00613                         axi_ss_geo::m_west_range,
00614                         axi_ss_geo::s_west_range));
00615     // Display the network interfaces on the right of the switch
00616     c.add(display_border
00617           ("anchor=south west",
00618            get_tikz_coordinate,
00619            [&, width = width] (auto i) {
00620              return std::tuple
00621                { core_size.x() + width - 1,
00622                   core_size.y() + i + 1
00623                   + ranges::distance(axi_ss_geo::m_me_range)
00624                   + ranges::distance(axi_ss_geo::m_dma_range)
00625                   + ranges::distance(axi_ss_geo::m_tile_ctrl_range)
00626                   + ranges::distance(axi_ss_geo::m_fifo_range) };
00627           },
00628           axi_ss_geo::s_east_range,
```

References c, and i.

9.1.2.1.3.5 display_border() [1/2] template<typename AXIStreamGeography > template<typename FG , typename FL , typename... EnumRanges> auto trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::display_border (

```
const std::string_view & node_attribute,
FG && global_coordinate_function,
FL && local_coordinate_function,
EnumRanges &&... enum_ranges ) const  [inline]
```

Display a border of the AXI stream switch.

Definition at line 520 of file axi_stream_switch.hpp.

```
00525                                                 {
00526     std::string out;
00527     // Index each connection displayed in order
00528     auto i = 0;
00529     boost::hana::for_each(boost::hana::make_tuple(enum_ranges...),
00530       [&] (auto&& r) {
00531         for (auto p :  r) {
00532           auto [x, y] = local_coordinate_function(i);
00533           out += (boost::format { R"(
00534 \coordinate(%3%) at %2%;
00535 \node[%1%] at %2% {%4%};)" }
00536 % node_attribute
00537 % global_coordinate_function(x, y)
00538 // Make a node name from enum name prefixed with master/slave statu7s
00539             % latex::context::clean_node(magic_enum::enum_name(p),
00540                                          is_axi_master(p) ?  "M" :  "S")
00541             // Use the enum name as the port name to display
00542             % magic_enum::enum_name(p)).str();
00543           ++i;
```

References i.

9.1.2.1.3.6 display_border() [2/2] template<typename AXIStreamGeography >

```
template<typename FG , typename FL , typename...  EnumRanges>
auto trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::display_↩
border (
              const std::string_view & node_attribute,
              FG && global_coordinate_function,
              FL && local_coordinate_function,
              EnumRanges &&...  enum_ranges ) const  [inline]
```

Display a border of the AXI stream switch.

Definition at line 520 of file axi_stream_switch.hpp.

```
00525                                                                    {
00526     std::string out;
00527     // Index each connection displayed in order
00528     auto i = 0;
00529     boost::hana::for_each(boost::hana::make_tuple(enum_ranges...),
00530       [&] (auto&& r) {
00531         for (auto p :  r) {
00532           auto [x, y] = local_coordinate_function(i);
00533           out += (boost::format { R"(
00534 \coordinate(%3%) at %2%;
00535 \node[%1%] at %2% {%4%};)" }
00536 % node_attribute
00537 % global_coordinate_function(x, y)
00538 // Make a node name from enum name prefixed with master/slave statu7s
00539             % latex::context::clean_node(magic_enum::enum_name(p),
00540                                          is_axi_master(p) ?  "M" :  "S")
00541             // Use the enum name as the port name to display
00542             % magic_enum::enum_name(p)).str();
00543           ++i;
```

References i.

9.1.2.1.3.7 display_routing_configuration() [1/2] template<typename AXIStreamGeography >

```
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::display_↩
routing_configuration (
              latex::context & c ) const  [inline]
```

Display the internal switch connections with routing arrows perpendicular to the borders.

< To the left

< Downwards

< To the right

< Upwards

< To the left

< Downwards

< To the right

< Upwards

Definition at line 477 of file axi_stream_switch.hpp.

```
00477                                                                    {
00478     for (const auto& [i, p] :    ranges::views::enumerate(input_ports)) {
00479       auto input_port = static_cast<spl>(i);
00480       auto rm = get_as_router_minion(p.get());
00481       for (auto output_port :  rm->get_master_port_dests()) {
00482         auto input_port_name = magic_enum::enum_name(input_port);
```

```
00483         // Compute the angle of the starting arrow from the input
00484         auto input_angle = tikz_edge_angle(input_port_name,
00485                                            { { "east", 180 }, ///< To the left
00486                                              { "north", -90 }, ///< Downwards
00487                                              { "west", 0 }, ///< To the right
00488                                              { "dma", 90 }, ///< Upwards
00489                                              { "fifo", 90 },
00490                                              { "me", 90 },
00491                                              { "tile_ctrl", 90 },
00492                                              { "south", 90 } });
00493         auto output_port_name = magic_enum::enum_name(output_port);
00494         // Compute the angle of the ending arrow to the output
00495         auto output_angle = tikz_edge_angle(output_port_name,
00496                                             { { "east", 180 }, ///< To the left
00497                                               { "north", -90 }, ///< Downwards
00498                                               { "dma", 0 }, ///< To the right
00499                                               { "fifo", 0 },
00500                                               { "me", 0 },
00501                                               { "tile_ctrl", 0 },
00502                                               { "west", 0 },
00503                                               { "south", 90 } }); ///< Upwards
00504         // Use specific color for core emission or reception
00505         auto color = input_port_name.starts_with("me") ?  "red"
00506           :  output_port_name.starts_with("me") ?  "teal"
00507           :  input_port_name.starts_with("dma") ?  "magenta"
00508           :  output_port_name.starts_with("dma") ?  "violet" :  "blue";
00509         c.add((boost::format { R"(
00510 \draw[%5%] (node cs:name=S%1%)
00511 to [out=%2%, in=%4%] (node cs:name=M%3%);)" }
00512 % c.clean_node(input_port_name) % input_angle
00513 % c.clean_node(output_port_name) % output_angle
00514 % color).str());
00515 }
```

References c, color, i, and p.

9.1.2.1.3.8 display_routing_configuration() [2/2]

```
template<typename AXIStreamGeography >
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::display_routing_configuration (
            latex::context & c ) const  [inline]
```

Display the internal switch connections with routing arrows perpendicular to the borders.

< To the left

< Downwards

< To the right

< Upwards

< To the left

< Downwards

< To the right

< Upwards

Definition at line 477 of file axi_stream_switch.hpp.

```
00477                                                                 {
00478     for (const auto& [i, p] :    ranges::views::enumerate(input_ports)) {
00479       auto input_port = static_cast<spl>(i);
00480       auto rm = get_as_router_minion(p.get());
00481       for (auto output_port :  rm->get_master_port_dests()) {
00482         auto input_port_name = magic_enum::enum_name(input_port);
00483         // Compute the angle of the starting arrow from the input
00484         auto input_angle = tikz_edge_angle(input_port_name,
00485                                            { { "east", 180 }, ///< To the left
00486                                              { "north", -90 }, ///< Downwards
```

```
00487                                           { "west", 0 }, ///< To the right
00488                                           { "dma", 90 }, ///< Upwards
00489                                           { "fifo", 90 },
00490                                           { "me", 90 },
00491                                           { "tile_ctrl", 90 },
00492                                           { "south", 90 } });
00493         auto output_port_name = magic_enum::enum_name(output_port);
00494         // Compute the angle of the ending arrow to the output
00495         auto output_angle = tikz_edge_angle(output_port_name,
00496                                           { { "east", 180 }, ///< To the left
00497                                             { "north", -90 }, ///< Downwards
00498                                             { "dma", 0 }, ///< To the right
00499                                             { "fifo", 0 },
00500                                             { "me", 0 },
00501                                             { "tile_ctrl", 0 },
00502                                             { "west", 0 },
00503                                             { "south", 90 } }); ///< Upwards
00504         // Use specific color for core emission or reception
00505         auto color = input_port_name.starts_with("me") ?  "red"
00506           :  output_port_name.starts_with("me") ?  "teal"
00507           :  input_port_name.starts_with("dma") ?  "magenta"
00508           :  output_port_name.starts_with("dma") ?  "violet" :  "blue";
00509         c.add((boost::format { R"(
00510 \draw[%5%] (node cs:name=S%1%)
00511 to [out=%2%, in=%4%] (node cs:name=M%3%);)" }
00512 % c.clean_node(input_port_name) % input_angle
00513 % c.clean_node(output_port_name) % output_angle
00514 % color).str());
00515 }
```

References c, color, i, and p.

9.1.2.1.3.9 display_size() [1/2] template<typename AIEStreamGeography >

static vec< int, 2 > trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AIEStreamGeography >::display_size ( ) [inline], [static]

Compute the size of the graphics representation of the switch.

Definition at line 547 of file axi_stream_switch.hpp.

```
00551                                   {
00552     // Combine the lengths of the most packed sides,
00553     // + 1/+ 2 to keep room for labels on each side
00554     return { 1 + ranges::distance(axi_ss_geo::s_me_range)
00555              + ranges::distance(axi_ss_geo::m_dma_range)
00556              + ranges::distance(axi_ss_geo::m_tile_ctrl_range)
00557              + ranges::distance(axi_ss_geo::m_fifo_range)
00558              + ranges::distance(axi_ss_geo::m_south_range)
00559              + ranges::distance(axi_ss_geo::s_south_range),
00560              2 + ranges::distance(axi_ss_geo::s_me_range)
00561              + ranges::distance(axi_ss_geo::s_dma_range)
00562              + ranges::distance(axi_ss_geo::s_tile_ctrl_range)
```

9.1.2.1.3.10 display_size() [2/2] template<typename AIEStreamGeography >

static vec< int, 2 > trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AIEStreamGeography >::display_size ( ) [inline], [static]

Compute the size of the graphics representation of the switch.

Definition at line 547 of file axi_stream_switch.hpp.

```
00551                                   {
00552     // Combine the lengths of the most packed sides,
00553     // + 1/+ 2 to keep room for labels on each side
00554     return { 1 + ranges::distance(axi_ss_geo::s_me_range)
00555              + ranges::distance(axi_ss_geo::m_dma_range)
00556              + ranges::distance(axi_ss_geo::m_tile_ctrl_range)
00557              + ranges::distance(axi_ss_geo::m_fifo_range)
00558              + ranges::distance(axi_ss_geo::m_south_range)
00559              + ranges::distance(axi_ss_geo::s_south_range),
00560              2 + ranges::distance(axi_ss_geo::s_me_range)
00561              + ranges::distance(axi_ss_geo::s_dma_range)
00562              + ranges::distance(axi_ss_geo::s_tile_ctrl_range)
```

9.1.2.1.3.11 get_as_router_minion() [1/2] template<typename AXIStreamGeography >

```
static router_minion * trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStream↩
Geography >::get_as_router_minion (
            communicator_port * rp )  [inline], [static]
```

Get the router_minion behind a communicator_port.

Definition at line 227 of file axi_stream_switch.hpp.

```
00227                                                {
00228 #ifdef NDEBUG
00229     // No assert mode
00230     auto rm = static_cast<router_minion *>(rp);
00231 #else
00232     // In mode with assertion, check this is really a router_minion
00233     auto rm = dynamic_cast<router_minion *>(rp);
00234     if (!rm) {
00235       throw ::trisycl::runtime_error {
00236         "This is not a routable switch port" };
00237     }
00238 #endif
00239     return rm;
00240   }
```

9.1.2.1.3.12 get_as_router_minion() [2/2] template<typename AXIStreamGeography >

```
static router_minion * trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStream↩
Geography >::get_as_router_minion (
            communicator_port * rp )  [inline], [static]
```

Get the router_minion behind a communicator_port.

Definition at line 227 of file axi_stream_switch.hpp.

```
00227                                                {
00228 #ifdef NDEBUG
00229     // No assert mode
00230     auto rm = static_cast<router_minion *>(rp);
00231 #else
00232     // In mode with assertion, check this is really a router_minion
00233     auto rm = dynamic_cast<router_minion *>(rp);
00234     if (!rm) {
00235       throw ::trisycl::runtime_error {
00236         "This is not a routable switch port" };
00237     }
00238 #endif
00239     return rm;
00240   }
```

9.1.2.1.3.13 in_connection() [1/2] template<typename AXIStreamGeography >

```
auto & trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::in_↩
connection (
            spl sp )  [inline]
```

Access the input connection behind an input port.

Parameters

in	*p*	is the input port number

Exceptions

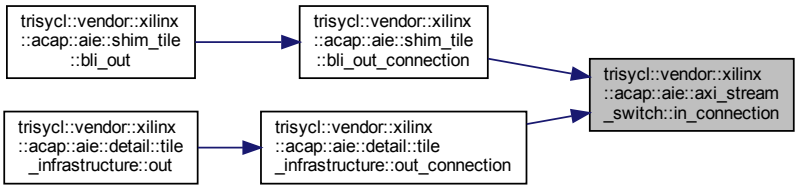
trisycl::runtime_error	if the port number is invalid

Definition at line 285 of file axi_stream_switch.hpp.

```
00285                         {
00286     using u_t = std::underlying_type_t<decltype(sp)>;
00287     auto port = static_cast<u_t>(sp);
00288     // \todo replace by std::ssize() in the future
00289     if (port >=  static_cast<std::ptrdiff_t>(std::size(input_ports)))
00290       throw ::trisycl::runtime_error {
00291         (boost::format {
00292           "in_connection:  %1% is not a valid port number between 0 and %2%" }
00293           % port % (std::size(input_ports) - 1)).str() };
00294     return input_ports[port];
00295   }
```

Referenced by trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_out_connection(), and trisycl::vendor::xilinx::acap::aie::de

Here is the caller graph for this function:



9.1.2.1.3.14 in_connection() [2/2]

```
template<typename AXIStreamGeography >
auto & trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::in_↩
connection (
             spl sp )  [inline]
```

Access the input connection behind an input port.

Parameters

in	*p*	is the input port number

Exceptions

trisycl::runtime_error	if the port number is invalid

Definition at line 285 of file axi_stream_switch.hpp.

```
00285                         {
00286     using u_t = std::underlying_type_t<decltype(sp)>;
00287     auto port = static_cast<u_t>(sp);
00288     // \todo replace by std::ssize() in the future
00289     if (port >=  static_cast<std::ptrdiff_t>(std::size(input_ports)))
00290       throw ::trisycl::runtime_error {
00291         (boost::format {
00292           "in_connection:  %1% is not a valid port number between 0 and %2%" }
00293           % port % (std::size(input_ports) - 1)).str() };
00294     return input_ports[port];
00295   }
```

9.1.2.1.3.15 input() [1/2] template<typename AXIStreamGeography >

```
auto & trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input (
            spl p )  [inline]
```

Get the input router port of the AXI stream switch.

Parameters

p	is the slave_port_layout for the stream

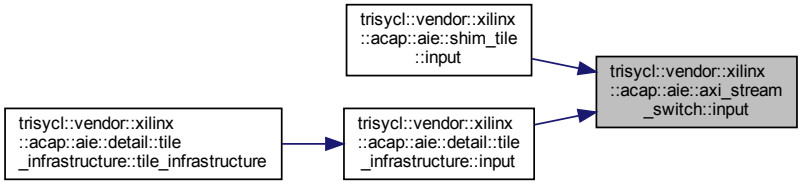
Definition at line 373 of file axi_stream_switch.hpp.

```
00373                     {
00374     // No index validation required because of type safety
00375     return input_ports[::trisycl::detail::underlying_value(p)];
00376   }
```

References p, and trisycl::detail::underlying_value.

Referenced by trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::input(), and trisycl::vendor::xilinx::acap::aie::detail::tile_infras

Here is the caller graph for this function:



9.1.2.1.3.16 input() [2/2] template<typename AXIStreamGeography >

```
auto & trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input (
            spl p )  [inline]
```

Get the input router port of the AXI stream switch.

Parameters

p	is the slave_port_layout for the stream

Definition at line 373 of file axi_stream_switch.hpp.

```
00373                     {
00374     // No index validation required because of type safety
00375     return input_ports[::trisycl::detail::underlying_value(p)];
00376   }
```

References p, and trisycl::detail::underlying_value.

9.1.2.1.3.17 out_connection() [1/2] template<typename AXIStreamGeography >

```
auto & trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::out_↩
connection (
            mpl mp )  [inline]
```

Access the output connection behind an output port.

Parameters

in	*p*	is the output port number

Exceptions

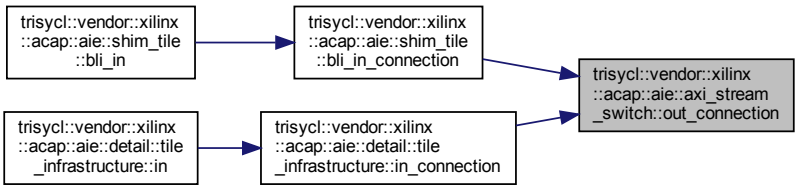
trisycl::runtime_error	if the port number is invalid

Definition at line 304 of file axi_stream_switch.hpp.

```
00304                                {
00305     using u_t = std::underlying_type_t<decltype(mp)>;
00306     auto port = static_cast<u_t>(mp);
00307     // \todo replace by std::ssize() in the future
00308     if (port >= static_cast<std::ptrdiff_t>(std::size(output_ports)))
00309       throw ::trisycl::runtime_error {
00310         (boost::format {
00311           "out_connection:  %1% is not a valid port number between 0 and %2%" }
00312           % port % (std::size(output_ports) - 1)).str() };
00313     return output_ports[port];
00314   }
```

Referenced by trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_in_connection(), and trisycl::vendor::xilinx::acap::aie::det

Here is the caller graph for this function:



9.1.2.1.3.18 out_connection() [2/2] template<typename AXIStreamGeography >

```
auto & trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::out_↩
connection (
            mpl mp )  [inline]
```

Access the output connection behind an output port.

Parameters

in	*p*	is the output port number

Exceptions

trisycl::runtime_error	if the port number is invalid

Definition at line 304 of file axi_stream_switch.hpp.

```
00304                                {
00305     using u_t = std::underlying_type_t<decltype(mp)>;
00306     auto port = static_cast<u_t>(mp);
00307     // \todo replace by std::ssize() in the future
00308     if (port >= static_cast<std::ptrdiff_t>(std::size(output_ports)))
00309       throw ::trisycl::runtime_error {
00310         (boost::format {
00311           "out_connection:  %1% is not a valid port number between 0 and %2%" }
00312            % port % (std::size(output_ports) - 1)).str() };
00313     return output_ports[port];
00314   }
```

9.1.2.1.3.19 output() [1/2] template<typename AXIStreamGeography >

```
auto & trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::output (
            mpl p )  [inline]
```

Get the input router port of the AXI stream switch.

Parameters

p	is the master_port_layout for the stream

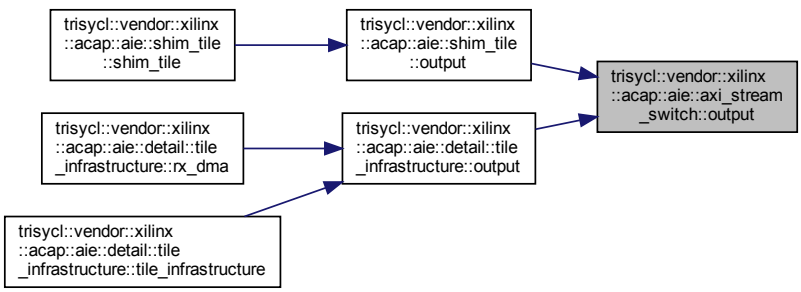
Definition at line 383 of file axi_stream_switch.hpp.

```
00383                          {
00384     // No index validation required because of type safety
00385     return output_ports[::trisycl::detail::underlying_value(p)];
00386   }
```

References p, and trisycl::detail::underlying_value.

Referenced by trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::output(), and trisycl::vendor::xilinx::acap::aie::detail::tile_infra

Here is the caller graph for this function:



9.1.2.1.3.20 output() [2/2] template<typename AXIStreamGeography >

```
auto & trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::output (
            mpl p )  [inline]
```

Get the input router port of the AXI stream switch.

Parameters

p	is the master_port_layout for the stream

Definition at line 383 of file axi_stream_switch.hpp.

```
00383                     {
00384     // No index validation required because of type safety
00385     return output_ports[::trisycl::detail::underlying_value(p)];
00386   }
```

References p, and trisycl::detail::underlying_value.

9.1.2.1.3.21 start() [1/2] template<typename AXIStreamGeography >

```
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::start (
            int x,
            int y,
            ::trisycl::detail::fiber_pool & fiber_executor )  [inline]
```

Start the tile infrastructure associated to the AIE device.

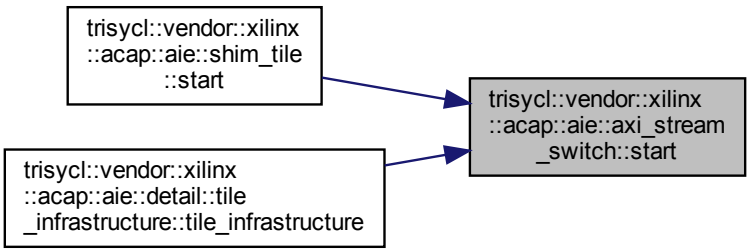
Definition at line 420 of file axi_stream_switch.hpp.

```
00420                                                                    {
00421     x_coordinate = x;
00422     y_coordinate = y;
00423     // Add a router worker on all switch input
00424     for (auto& p :  input_ports) {
00425       auto port_enum =
00426         magic_enum::enum_cast<spl>(std::distance(input_ports.begin(), &p));
00427 #ifdef TRISYCL_DEBUG
00428       auto port_name = magic_enum::enum_name(port_enum.value());
00429 #endif
00430       TRISYCL_DUMP_T("axi_stream_switch " « this
00431                      « " (" « x_coordinate « ',' « y_coordinate
00432                      « ") start initialize input:  " « port_name);
00433       // If the port comes from the FIFO, use a FIFO-featured
00434       // router_minion instead
00435       if (ranges::find(axi_ss_geo::s_fifo_range, port_enum)
00436           != std::end(axi_ss_geo::s_fifo_range))
00437         /* A special AXI switch routing interface which provides a
00438 FIFO.  Since a router_minion has already some internal FIFO
00439 behavior, some deeper FIFO behavior can actually be
00440 implemented with a specific router_minion with a deeper
00441 FIFO internally, to avoid having 2 active components, the
00442 FIFO + the router_minion */
00443         p = std::make_shared<router_minion>(*this,
00444                                             fiber_executor,
00445                                             axi_ss_geo::fifo_depth);
00446       else
00447         p = std::make_shared<router_minion>(*this, fiber_executor);
00448     }
00449     // Since the FIFO behavior is actually implemented with a
00450     // router_minion with a deeper FIFO internally, the "input" of the
00451     // "FIFO" can be just directly connected to this specific
00452     // router_minion
00453     for (auto [o, i] :  ranges::views::zip(axi_ss_geo::m_fifo_range,
00454                                            axi_ss_geo::s_fifo_range))
00455       output(o) = input(i);
00456   }
```

References i, p, TRISYCL_DUMP_T, x, and trisycl::y.

Referenced by trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::start(), and trisycl::vendor::xilinx::acap::aie::detail::tile_infras

Here is the caller graph for this function:



9.1.2.1.3.22 start() [2/2] template<typename AXIStreamGeography >

```
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::start (
            int x,
            int y,
            ::trisycl::detail::fiber_pool & fiber_executor )  [inline]
```

Start the tile infrastructure associated to the AIE device.

Definition at line 420 of file axi_stream_switch.hpp.

```
00420                                                                    {
00421     x_coordinate = x;
00422     y_coordinate = y;
00423     // Add a router worker on all switch input
00424     for (auto& p :  input_ports) {
00425       auto port_enum =
00426         magic_enum::enum_cast<spl>(std::distance(input_ports.begin(), &p));
00427 #ifdef TRISYCL_DEBUG
00428       auto port_name = magic_enum::enum_name(port_enum.value());
00429 #endif
00430       TRISYCL_DUMP_T("axi_stream_switch " « this
00431                      « " (" « x_coordinate « ',' « y_coordinate
00432                      « ") start initialize input:  " « port_name);
00433       // If the port comes from the FIFO, use a FIFO-featured
00434       // router_minion instead
00435       if (ranges::find(axi_ss_geo::s_fifo_range, port_enum)
00436           != std::end(axi_ss_geo::s_fifo_range))
00437         /* A special AXI switch routing interface which provides a
00438 FIFO.  Since a router_minion has already some internal FIFO
00439 behavior, some deeper FIFO behavior can actually be
00440 implemented with a specific router_minion with a deeper
00441 FIFO internally, to avoid having 2 active components, the
00442 FIFO + the router_minion */
00443         p = std::make_shared<router_minion>(*this,
00444                                             fiber_executor,
00445                                             axi_ss_geo::fifo_depth);
00446       else
00447         p = std::make_shared<router_minion>(*this, fiber_executor);
00448     }
00449     // Since the FIFO behavior is actually implemented with a
00450     // router_minion with a deeper FIFO internally, the "input" of the
00451     // "FIFO" can be just directly connected to this specific
00452     // router_minion
00453     for (auto [o, i] :  ranges::views::zip(axi_ss_geo::m_fifo_range,
00454                                            axi_ss_geo::s_fifo_range))
00455       output(o) = input(i);
00456   }
```

References i, p, TRISYCL_DUMP_T, x, and trisycl::y.

9.1.2.1.3.23 tikz_edge_angle() [1/2] template<typename AXIStreamGeography >

```
int trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::tikz_edge_↩
angle (
            const std::string_view & port_name,
            const std::initializer_list< std::tuple< std::string_view, int > > & desc )
const  [inline]
```

Compute the edge angle for a port name according a list of description associating a port name prefix and an associated angle in degree.

Definition at line 464 of file axi_stream_switch.hpp.

```
00466                                                                                {
00467     for (auto& [prefix, angle] :  desc)
00468       if (port_name.starts_with(prefix))
00469         return angle;
00470     assert(0 && "Unknown port name");
00471     return 0;
00472   }
```

9.1.2.1.3.24 tikz_edge_angle() [2/2] template<typename AXIStreamGeography >

```
int trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::tikz_edge_↩
angle (
            const std::string_view & port_name,
            const std::initializer_list< std::tuple< std::string_view, int > > & desc )
const  [inline]
```

Compute the edge angle for a port name according a list of description associating a port name prefix and an associated angle in degree.

Definition at line 464 of file axi_stream_switch.hpp.

```
00466                                                                                {
00467     for (auto& [prefix, angle] :  desc)
00468       if (port_name.starts_with(prefix))
00469         return angle;
00470     assert(0 && "Unknown port name");
00471     return 0;
00472   }
```

9.1.2.1.4 Member Data Documentation

9.1.2.1.4.1 input_ports template<typename AXIStreamGeography >

```
std::array< std::shared_ptr< communicator_port >, axi_ss_geo::nb_slave_port > trisycl::vendor::xilinx::acap:
AXIStreamGeography >::input_ports
```

The input ports used to send information to the switch.

Use a shared pointer to represent the interconnect between switches in a shortcut way.

Definition at line 352 of file axi_stream_switch.hpp.

9.1.2.1.4.2 output_ports template<typename AXIStreamGeography >

std::array< std::shared_ptr< communicator_port >, axi_ss_geo::nb_master_port > trisycl::vendor::xilinx::acap AXIStreamGeography >::output_ports

The input ports used to send information to the switch.

Use a shared pointer to represent the interconnect between switches in a shortcut way.

Definition at line 360 of file axi_stream_switch.hpp.

9.1.2.1.4.3 stream_latency template<typename AXIStreamGeography >

static constexpr auto trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::stream_latency = 4 [static], [constexpr]

Definition at line 335 of file axi_stream_switch.hpp.

9.1.2.1.4.4 translate_port template<typename AXIStreamGeography >

static auto trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >↩ ::translate_port [inline], [static]

Initial value:

```
= [] (int user_port,
                                        auto physical_port_min,
                                        auto physical_port_max,
                                        const auto& error_message) {

    auto port_min = static_cast<int>(physical_port_min);
    auto port_max = static_cast<int>(physical_port_max);
    auto last_user_port = port_max - port_min;
    if (user_port < 0 || user_port > last_user_port)
      throw ::trisycl::runtime_error {
        (boost::format {
          "%1%:  %2% is not a valid port number between 0 and %3%" }
          % error_message % user_port % last_user_port).str() };

    return static_cast<decltype(physical_port_min)>(port_min + user_port);
  }
```

Validate the user port number and translate it to the physical port number.

Parameters

in	*user_port*	is the logical port number, starting to 0
in	*physical_port_min*	is the physical lower port number to map to
in	*physical_port_max*	is the physical higher port number to map to

Exceptions

trisycl::runtime_error	if the port number is invalid

Returns

the physical port number in the switch corresponding to the logical port

Todo This makes sense only for core tile

Parameters

in	*user_port*	is the logical port number, starting to 0
in	*physical_port_min*	is the physical lower port number to map to
in	*physical_port_max*	is the physical higher port number to map to

Exceptions

trisycl::runtime_error	if the port number is invalid

Returns

the physical port number in the switch corresponding to the logical port

Todo This makes sense only for core tile

Definition at line 260 of file axi_stream_switch.hpp.

9.1.2.1.4.5 x_coordinate `template<typename AXIStreamGeography >`

`int trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::x_coordinate`

Keep the horizontal coordinate for debug purpose.

Definition at line 363 of file axi_stream_switch.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::connect_to(), trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::read(), trisycl::vendor::xilinx::acap::aie:
and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::write().

9.1.2.1.4.6 y_coordinate `template<typename AXIStreamGeography >`

`int trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::y_coordinate`

Keep the vertical coordinate for debug purpose.

Definition at line 366 of file axi_stream_switch.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::connect_to(), trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::read(), trisycl::vendor::xilinx::acap::aie:
and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::write().

9.1.2.2 struct trisycl::vendor::xilinx::acap::aie::cascade_stream

template<typename Geography>
struct trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >

Implementation of the cascade stream infrastructure between AI Engine tiles.

Based on Math Engine (ME) Architecture Specification, Revision v2.01

March 2019

3.5.4 Data Movement using Cascade Streams, p. 93

2.14 Device-level Floorplanning Guidelines, 2.14.2 Array Edges, p. 69

4.4 ME Core Interfaces, 4.4.4 Cascade Stream Interface, p. 120 explains there is a FIFO depth of 4

5.9.3.6 Cascade Streams: explains that there are only blocking accesses

Direct stream interface: One cascade stream in, one cascade stream out (384-bits)

Todo Remove this global infrastructure and distribute it on the tile_infrastructure itself

Based on Math Engine (ME) Architecture Specification, Revision v2.01

March 2019

3.5.4 Data Movement using Cascade Streams, p. 93

2.14 Device-level Floorplanning Guidelines, 2.14.2 Array Edges, p. 69

4.4 ME Core Interfaces, 4.4.4 Cascade Stream Interface, p. 120 explains there is a FIFO depth of 4

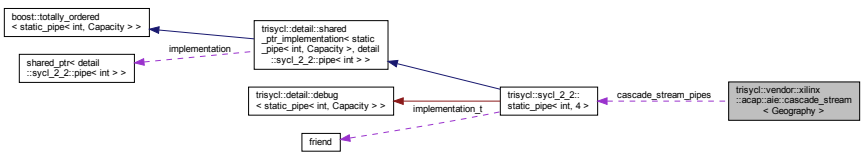
5.9.3.6 Cascade Streams: explains that there are only blocking accesses

Direct stream interface: One cascade stream in, one cascade stream out (384-bits)

Todo Remove this global infrastructure and distribute it on the tile_infrastructure itself

Definition at line 46 of file cascade_stream.hpp.

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >:



Public Types

- using geo = Geography
- using geo = Geography

Public Member Functions

- template<typename T >
  auto get_cascade_stream_in (int x, int y) const

  Get a blocking read accessor to the cascade stream input.
- template<typename T >
  auto get_cascade_stream_out (int x, int y) const

  Get a blocking write accessor to the cascade stream output.
- template<typename T >
  auto get_cascade_stream_in (int x, int y) const

  Get a blocking read accessor to the cascade stream input.
- template<typename T >
  auto get_cascade_stream_out (int x, int y) const

  Get a blocking write accessor to the cascade stream output.

Public Attributes

- ::trisycl::sycl_2_2::static_pipe< int, 4 > cascade_stream_pipes [geo::x_size *geo::y_size+1]

  The pipes for the cascade streams, with 1 spare pipe on each side of the PE string.

9.1.2.2.1 Member Typedef Documentation

9.1.2.2.1.1 geo [1/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::geo = Geography
```

Definition at line 47 of file cascade_stream.hpp.

9.1.2.2.1.2 geo [2/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::geo = Geography
```

Definition at line 47 of file cascade_stream.hpp.

9.1.2.2.2 Member Function Documentation

9.1.2.2.2.1 get_cascade_stream_in() [1/2] template<typename Geography >

```
template<typename T >
auto trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::get_cascade_stream_in (
            int x,
            int y ) const  [inline]
```

Get a blocking read accessor to the cascade stream input.

Parameters

	T	is the data type used to read from the cascade stream pipe
in	x	is the horizontal tile coordinate
in	y	is the vertical tile coordinate

Definition at line 76 of file cascade_stream.hpp.

```
00076                                                {
00077     return cascade_stream_pipes[geo::cascade_linear_id(x, y)]
00078       .template get_access<access::mode::read,
00079                            access::target::blocking_pipe>();
00080   }
```

References trisycl::access::blocking_pipe, trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::cascade_stream_pipes, trisycl::access::read, x, and trisycl::y.

9.1.2.2.2 get_cascade_stream_in() [2/2] template<typename Geography >

```
template<typename T >
auto trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::get_cascade_stream_in (
             int x,
             int y ) const  [inline]
```

Get a blocking read accessor to the cascade stream input.

Parameters

	T	is the data type used to read from the cascade stream pipe
in	x	is the horizontal tile coordinate
in	y	is the vertical tile coordinate

Definition at line 76 of file cascade_stream.hpp.

```
00076                                                {
00077     return cascade_stream_pipes[geo::cascade_linear_id(x, y)]
00078       .template get_access<access::mode::read,
00079                            access::target::blocking_pipe>();
00080   }
```

References trisycl::access::blocking_pipe, trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::cascade_stream_pipes, trisycl::access::read, x, and trisycl::y.

9.1.2.2.3 get_cascade_stream_out() [1/2] template<typename Geography >

```
template<typename T >
auto trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::get_cascade_stream_out (
             int x,
             int y ) const  [inline]
```

Get a blocking write accessor to the cascade stream output.

Parameters

	T	is the data type used to write to the cascade stream pipe
in	x	is the horizontal tile coordinate
in	y	is the vertical tile coordinate

Definition at line 93 of file cascade_stream.hpp.

```
00093                                                {
00094     // The output is connected to the down-stream neighbour of the cascade
00095     return cascade_stream_pipes[geo::cascade_linear_id(x, y) + 1]
00096       .template get_access<access::mode::write,
00097                            access::target::blocking_pipe>();
00098   }
```

References trisycl::access::blocking_pipe, trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::cascade_stream_pipes, trisycl::access::write, x, and trisycl::y.

9.1.2.2.4 get_cascade_stream_out() [2/2] template<typename Geography > template<typename T > auto trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::get_cascade_stream_out (int x, int y) const [inline]

Get a blocking write accessor to the cascade stream output.

Parameters

	T	is the data type used to write to the cascade stream pipe
in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Definition at line 93 of file cascade_stream.hpp.

```
00093                                                {
00094     // The output is connected to the down-stream neighbour of the cascade
00095     return cascade_stream_pipes[geo::cascade_linear_id(x, y) + 1]
00096       .template get_access<access::mode::write,
00097                            access::target::blocking_pipe>();
00098   }
```

References trisycl::access::blocking_pipe, trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::cascade_stream_pipes, trisycl::access::write, x, and trisycl::y.

9.1.2.2.3 Member Data Documentation

9.1.2.2.3.1 cascade_stream_pipes template<typename Geography > trisycl::sycl_2_2::static_pipe< int, 4 > trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::cascade_stream_pipes

The pipes for the cascade streams, with 1 spare pipe on each side of the PE string.

Todo Use a data type with 384 bits

Todo Use a data type with 384 bits

Definition at line 54 of file cascade_stream.hpp.

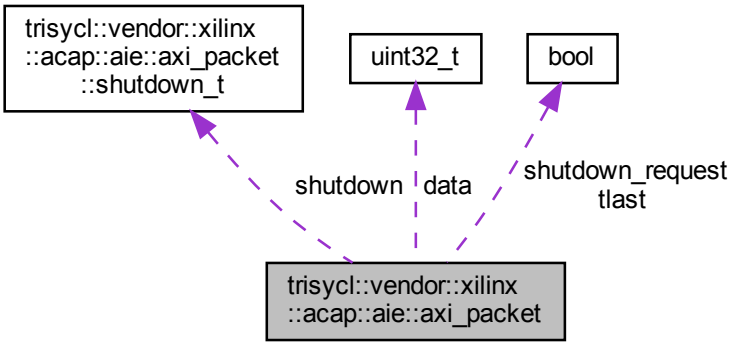
Referenced by trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::get_cascade_stream_in(), and trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::get_cascade_stream_out().

9.1.2.3 class trisycl::vendor::xilinx::acap::aie::axi_packet

Packet moving in AIE network-on-chip (NoC)

Definition at line 31 of file connection.hpp.

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::axi_packet:



Public Types

- using value_type = std::uint32_t

  Payload data type.
- using value_type = std::uint32_t

  Payload data type.

Public Member Functions

- axi_packet (const value_type &data)

  Implicit constructor from a data value.
- axi_packet (shutdown_t)

  Construct a shutdown request packet from axi_packet::shutdown.
- axi_packet ()=default
- axi_packet (const value_type &data)

  Implicit constructor from a data value.
- axi_packet (shutdown_t)

  Construct a shutdown request packet from axi_packet::shutdown.
- axi_packet ()=default

Public Attributes

- value_type data

  Network payload.
- bool tlast = false

  Signal end of packet stream.
- bool shutdown_request = false

  Signal a router shutdown.

Static Public Attributes

- static const shutdown_t shutdown

  Marker value used to construct a packet asking for shutdown.

9.1.2.3.1 Member Typedef Documentation

9.1.2.3.1.1 value_type [1/2] using trisycl::vendor::xilinx::acap::aie::axi_packet::value_type = std::uint32_t

Payload data type.

Definition at line 38 of file connection.hpp.

9.1.2.3.1.2 value_type [2/2] using trisycl::vendor::xilinx::acap::aie::axi_packet::value_type = std::uint32_t

Payload data type.

Definition at line 38 of file connection.hpp.

9.1.2.3.2 Constructor & Destructor Documentation

9.1.2.3.2.1 axi_packet() [1/6] trisycl::vendor::xilinx::acap::aie::axi_packet::axi_packet (const value_type & *data*) [inline]

Implicit constructor from a data value.

Definition at line 53 of file connection.hpp.

```
00053 :  data { data } {}
```

9.1.2.3.2.2 axi_packet() [2/6] trisycl::vendor::xilinx::acap::aie::axi_packet::axi_packet (shutdown_t) [inline]

Construct a shutdown request packet from axi_packet::shutdown.

Definition at line 56 of file connection.hpp.

```
00056 :  shutdown_request { true } {}
```

9.1.2.3.2.3 axi_packet() [3/6] trisycl::vendor::xilinx::acap::aie::axi_packet::axi_packet ( ) [default]

9.1.2.3.2.4 axi_packet() [4/6] trisycl::vendor::xilinx::acap::aie::axi_packet::axi_packet (const value_type & data) [inline]

Implicit constructor from a data value.

Definition at line 53 of file connection.hpp.

```
00053 : data { data } {}
```

9.1.2.3.2.5 axi_packet() [5/6] trisycl::vendor::xilinx::acap::aie::axi_packet::axi_packet (shutdown_t) [inline]

Construct a shutdown request packet from axi_packet::shutdown.

Definition at line 56 of file connection.hpp.

```
00056 : shutdown_request { true } {}
```

9.1.2.3.2.6 axi_packet() [6/6] trisycl::vendor::xilinx::acap::aie::axi_packet::axi_packet ( ) [default]

9.1.2.3.3 Member Data Documentation

9.1.2.3.3.1 data value_type trisycl::vendor::xilinx::acap::aie::axi_packet::data

Network payload.

Definition at line 44 of file connection.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::write(), and trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::write().

9.1.2.3.3.2 shutdown static const shutdown_t trisycl::vendor::xilinx::acap::aie::axi_packet::shutdown [static]

Marker value used to construct a packet asking for shutdown.

Definition at line 41 of file connection.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::~router_minion().

9.1.2.3.3.3 shutdown_request bool trisycl::vendor::xilinx::acap::aie::axi_packet::shutdown_request = false

Signal a router shutdown.

Definition at line 50 of file connection.hpp.

9.1.2.3.4 tlast `bool trisycl::vendor::xilinx::acap::aie::axi_packet::tlast = false`

Signal end of packet stream.

Definition at line 47 of file connection.hpp.

9.1.2.4 class trisycl::vendor::xilinx::acap::aie::dma_dsl

template<typename Tile, typename DMA, typename LockRef = bool>
class trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >

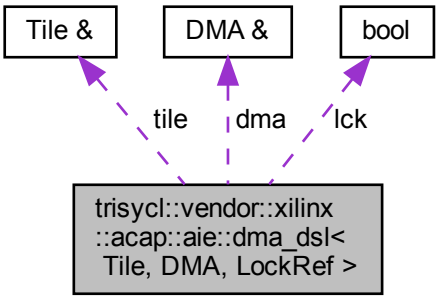
Implement a small DSL to chain DMA & locking operations like allowed by the hardware.

Parameters

`in`	*Tile*	is the type of tile to keep a reference on it
`in`	*DMA*	is the type of the DMA involved in the operation
`in`	*LockRef*	is the type of the reference to the lock once a lock() operation is involved. Otherwise just use a `bool` as a dummy type internally.

Definition at line 46 of file dma.hpp.

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >:



Public Member Functions

- dma_dsl (Tile &t, DMA &d)

  Keep track of operation with only DMA involved.
- dma_dsl (Tile &t, DMA &d, LockRef l)

  Keep track of operation with DMA and lock involved.
- auto lock (int l)

  Select a lock for the following operations.
- template<typename Cmd >
  auto & receive (Cmd &&cmd)

Enqueue a DMA receive command.

- template<typename Cmd >
  auto & send (Cmd &&cmd)

  Enqueue a DMA transfer command.
- auto & wait ()

  Wait for the transfers to complete.
- dma_dsl (Tile &t, DMA &d)

  Keep track of operation with only DMA involved.
- dma_dsl (Tile &t, DMA &d, LockRef l)

  Keep track of operation with DMA and lock involved.
- auto lock (int l)

  Select a lock for the following operations.
- template<typename Cmd >
  auto & receive (Cmd &&cmd)

  Enqueue a DMA receive command.
- template<typename Cmd >
  auto & send (Cmd &&cmd)

  Enqueue a DMA transfer command.
- auto & wait ()

  Wait for the transfers to complete.

Private Attributes

- Tile & tile

  Keep track of the tile involved in the operation chain.
- DMA & dma

  Keep track of the DMA involved in the operation chain.
- LockRef lck

  Keep track of the current lock involved in the operation chain if any.

9.1.2.4.1 Constructor & Destructor Documentation

9.1.2.4.1.1 dma_dsl() [1/4] template<typename Tile , typename DMA , typename LockRef = bool>

```
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::dma_dsl (
            Tile & t,
            DMA & d )  [inline]
```

Keep track of operation with only DMA involved.

Definition at line 59 of file dma.hpp.

```
00060       :  tile { t }
00061       , dma { d } {}
```

9.1.2.4.1.2 dma_dsl() [2/4] template<typename Tile , typename DMA , typename LockRef = bool>

```
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::dma_dsl (
            Tile & t,
            DMA & d,
            LockRef l )  [inline]
```

Keep track of operation with DMA and lock involved.

Definition at line 64 of file dma.hpp.

```
00065       :  tile { t }
00066       , dma { d }
00067       , lck { l } {}
```

9.1.2.4.1.3 dma_dsl() [3/4] template<typename Tile , typename DMA , typename LockRef = bool>

```
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::dma_dsl (
            Tile & t,
            DMA & d )  [inline]
```

Keep track of operation with only DMA involved.

Definition at line 59 of file dma.hpp.

```
00060       :  tile { t }
00061       , dma { d } {}
```

9.1.2.4.1.4 dma_dsl() [4/4] template<typename Tile , typename DMA , typename LockRef = bool>

```
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::dma_dsl (
            Tile & t,
            DMA & d,
            LockRef l )  [inline]
```

Keep track of operation with DMA and lock involved.

Definition at line 64 of file dma.hpp.

```
00065       :  tile { t }
00066       , dma { d }
00067       , lck { l } {}
```

9.1.2.4.2 Member Function Documentation

9.1.2.4.2.1 lock() [1/2] template<typename Tile , typename DMA , typename LockRef = bool>

```
auto trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::lock (
            int l )  [inline]
```

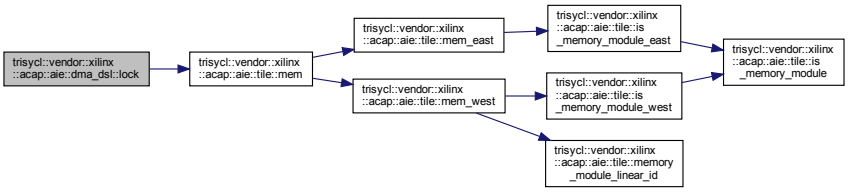
Select a lock for the following operations.

Definition at line 70 of file dma.hpp.

```
00070                 {
00071     auto& lock_ref = tile.mem().lock(l);
00072     return dma_dsl<Tile, DMA, decltype(lock_ref)> { tile, dma, lock_ref };
00073   }
```

References trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::dma, trisycl::vendor::xilinx::acap::aie::tile< AIE_Progra
and trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::tile.

Here is the call graph for this function:



9.1.2.4.2.2 lock() [2/2] template<typename Tile , typename DMA , typename LockRef = bool>

```
auto trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::lock (
            int l )  [inline]
```

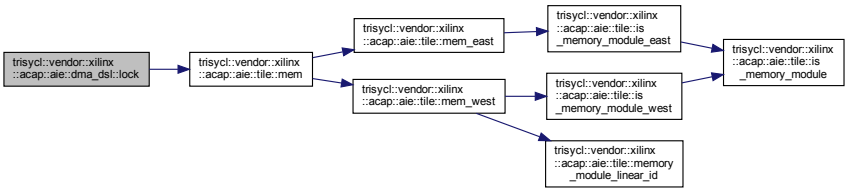
Select a lock for the following operations.

Definition at line 70 of file dma.hpp.

```
00070                     {
00071     auto& lock_ref = tile.mem().lock(l);
00072     return dma_dsl<Tile, DMA, decltype(lock_ref)> { tile, dma, lock_ref };
00073   }
```

References trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::dma, trisycl::vendor::xilinx::acap::aie::tile< AIE_Progra
and trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::tile.

Here is the call graph for this function:



9.1.2.4.2.3 receive() [1/2] template<typename Tile , typename DMA , typename LockRef = bool>

```
template<typename Cmd >
auto & trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::receive (
            Cmd && cmd )  [inline]
```

Enqueue a DMA receive command.

Definition at line 76 of file dma.hpp.

```
00076                                           {
00077     dma.receive(std::forward<Cmd>(cmd));
00078     // To be able to chain another DMA operation on it
00079     return *this;
00080   }
```

9.1.2.4.2.4 receive() [2/2]

```
template<typename Tile , typename DMA , typename LockRef = bool>
template<typename Cmd >
auto & trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::receive (
            Cmd && cmd )  [inline]
```

Enqueue a DMA receive command.

Definition at line 76 of file dma.hpp.

```
00076                                                {
00077     dma.receive(std::forward<Cmd>(cmd));
00078     // To be able to chain another DMA operation on it
00079     return *this;
00080   }
```

9.1.2.4.2.5 send() [1/2]

```
template<typename Tile , typename DMA , typename LockRef = bool>
template<typename Cmd >
auto & trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::send (
            Cmd && cmd )  [inline]
```

Enqueue a DMA transfer command.

Definition at line 83 of file dma.hpp.

```
00083                                                {
00084     dma.send(std::forward<Cmd>(cmd));
00085     // To be able to chain another DMA operation on it
00086     return *this;
00087   }
```

9.1.2.4.2.6 send() [2/2]

```
template<typename Tile , typename DMA , typename LockRef = bool>
template<typename Cmd >
auto & trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::send (
            Cmd && cmd )  [inline]
```

Enqueue a DMA transfer command.

Definition at line 83 of file dma.hpp.

```
00083                                                {
00084     dma.send(std::forward<Cmd>(cmd));
00085     // To be able to chain another DMA operation on it
00086     return *this;
00087   }
```

9.1.2.4.2.7 wait() [1/2]

```
template<typename Tile , typename DMA , typename LockRef = bool>
auto & trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::wait ( )  [inline]
```

Wait for the transfers to complete.

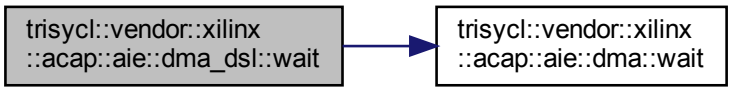
The API assumes that the program is written in such a way that at some point the queue will be empty so there is no continuous DMA command enqueuing for example.

Definition at line 94 of file dma.hpp.

```
00094                {
00095     dma.wait();
00096     // To be able to chain another DMA operation on it
00097     return *this;
00098   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::wait().

Here is the call graph for this function:



9.1.2.4.2.8 wait() [2/2] template<typename Tile , typename DMA , typename LockRef = bool>

auto & trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::wait ( ) [inline]

Wait for the transfers to complete.

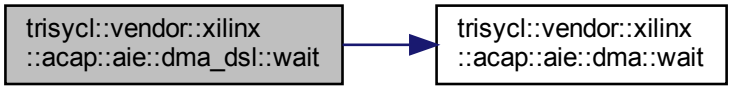
The API assumes that the program is written in such a way that at some point the queue will be empty so there is no continuous DMA command enqueuing for example.

Definition at line 94 of file dma.hpp.

```
00094               {
00095     dma.wait();
00096     // To be able to chain another DMA operation on it
00097     return *this;
00098   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::wait().

Here is the call graph for this function:



9.1.2.4.3 Member Data Documentation

9.1.2.4.3.1 dma template<typename Tile , typename DMA , typename LockRef = bool>

DMA & trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::dma [private]

Keep track of the DMA involved in the operation chain.

Definition at line 51 of file dma.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::lock().

9.1.2.4.3.2 lck `template<typename Tile , typename DMA , typename LockRef = bool>`

`LockRef trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::lck [private]`

Keep track of the current lock involved in the operation chain if any.

Definition at line 55 of file dma.hpp.

9.1.2.4.3.3 tile `template<typename Tile , typename DMA , typename LockRef = bool>`

`Tile & trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::tile [private]`

Keep track of the tile involved in the operation chain.

Definition at line 48 of file dma.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::lock().

9.1.2.5 struct trisycl::vendor::xilinx::acap::aie::geography

template<typename Layout>
struct trisycl::vendor::xilinx::acap::aie::geography< Layout >

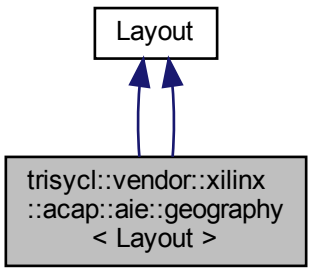
Some geographic information about the AI Engine array.

Parameters

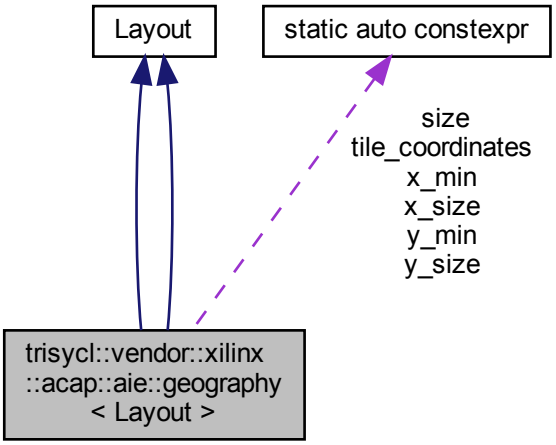
Layout	is a class describing the basic layout of the CGRA, setting for example the x & y size

Definition at line 33 of file geography.hpp.

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::geography< Layout >:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::geography< Layout >:



Public Types

- using layout = Layout

  Keep the layout used for this CGRA.
- using layout = Layout

  Keep the layout used for this CGRA.

Static Public Member Functions

- template<template< int X, int Y > typename Tileable>
  static auto generate_tiles ()

  Generate a tuple by tiling the Tileable object onto the CGRA geometry.
- static auto constexpr linear_id (int x, int y)

  Compute the linearized coordinate of a tile.
- static auto constexpr linear_x (int linear_id)

  Compute the horizontal coordinate of a tile from linear id.
- static auto constexpr linear_y (int linear_id)

  Compute the vertical coordinate of a tile from a linear id.
- static bool constexpr is_x_valid (int x)

  Test if an horizontal coordinate is valid.
- static void validate_x (int x)

  Validate an horizontal coordinate.
- static bool constexpr is_y_valid (int y)

  Test if a vertical coordinate is valid.
- static void validate_y (int y)

  Validate a vertical coordinate.
- static bool constexpr is_x_y_valid (int x, int y)

  Test if horizontal and vertical coordinates are valid.

- static void validate_x_y (int x, int y)

  Validate the horizontal and vertical coordinates.
- static bool constexpr is_west_column (int x)

  Test if a tile is in the Western column.
- static bool constexpr is_east_column (int x)

  Test if a tile is in the Eastern column.
- static bool constexpr is_south_row (int y)

  Test if a tile is in the Southern row.
- static bool constexpr is_north_row (int y)

  Test if a tile is in the Northern row.
- static bool constexpr is_cascade_start (int x, int y)

  Test if a tile owns the start of the cascade_stream.
- static bool constexpr is_cascade_end (int x, int y)

  Test if a tile owns the end of the cascade_stream.
- static auto constexpr cascade_linear_id (int x, int y)

  Compute the linearized coordinate of a tile along the cascade stream.
- static auto constexpr cascade_linear_x (int linear_id)

  Compute the horizontal coordinate of a tile from a cascade linear id.
- static auto constexpr cascade_linear_y (int linear_id)

  Compute the vertical coordinate of a tile from a cascade linear id.
- static bool constexpr is_valid_memory_module_offset (int dx, int dy)

  Test if a memory module coordinate offset is plausible.
- static bool constexpr is_memory_module (int x, int y, int dx, int dy)

  Test if a memory module exists and is connected to a tile.
- static auto constexpr memory_module_dx_coordinate (int y, int dx)

  Compute the memory module x coordinate delta from its tile.
- static auto constexpr memory_module_linear_id (int x, int y, int dx, int dy)

  Compute the linear id of a memory module attached to a tile.
- template<template< int X, int Y > typename Tileable>
  static auto generate_tiles ()

  Generate a tuple by tiling the Tileable object onto the CGRA geometry.
- static auto constexpr linear_id (int x, int y)

  Compute the linearized coordinate of a tile.
- static auto constexpr linear_x (int linear_id)

  Compute the horizontal coordinate of a tile from linear id.
- static auto constexpr linear_y (int linear_id)

  Compute the vertical coordinate of a tile from a linear id.
- static bool constexpr is_x_valid (int x)

  Test if an horizontal coordinate is valid.
- static void validate_x (int x)

  Validate an horizontal coordinate.
- static bool constexpr is_y_valid (int y)

  Test if a vertical coordinate is valid.
- static void validate_y (int y)

  Validate a vertical coordinate.
- static bool constexpr is_x_y_valid (int x, int y)

  Test if horizontal and vertical coordinates are valid.
- static void validate_x_y (int x, int y)

  Validate the horizontal and vertical coordinates.
- static bool constexpr is_west_column (int x)

  Test if a tile is in the Western column.

- static bool constexpr is_east_column (int x)

  Test if a tile is in the Eastern column.
- static bool constexpr is_south_row (int y)

  Test if a tile is in the Southern row.
- static bool constexpr is_north_row (int y)

  Test if a tile is in the Northern row.
- static bool constexpr is_cascade_start (int x, int y)

  Test if a tile owns the start of the cascade_stream.
- static bool constexpr is_cascade_end (int x, int y)

  Test if a tile owns the end of the cascade_stream.
- static auto constexpr cascade_linear_id (int x, int y)

  Compute the linearized coordinate of a tile along the cascade stream.
- static auto constexpr cascade_linear_x (int linear_id)

  Compute the horizontal coordinate of a tile from a cascade linear id.
- static auto constexpr cascade_linear_y (int linear_id)

  Compute the vertical coordinate of a tile from a cascade linear id.
- static bool constexpr is_valid_memory_module_offset (int dx, int dy)

  Test if a memory module coordinate offset is plausible.
- static bool constexpr is_memory_module (int x, int y, int dx, int dy)

  Test if a memory module exists and is connected to a tile.
- static auto constexpr memory_module_dx_coordinate (int y, int dx)

  Compute the memory module x coordinate delta from its tile.
- static auto constexpr memory_module_linear_id (int x, int y, int dx, int dy)

  Compute the linear id of a memory module attached to a tile.

Static Public Attributes

- static auto constexpr x_min = 0

  Minimum horizontal coordinate.
- static auto constexpr y_min = 0

  Minimum vertical coordinate.
- static auto constexpr x_size = layout::x_max + 1

  Number of tile horizontally in the CGRA.
- static auto constexpr y_size = layout::y_max + 1

  Number of tile vertically in the CGRA.
- static auto constexpr size = x_size∗y_size

  Total number of tiles in the CGRA.
- static auto constexpr tile_coordinates

  A tuple with the coordinate tuples of all the tiles inside the CGRA geometry.

9.1.2.5.1 Member Typedef Documentation

9.1.2.5.1.1 layout [1/2] template<typename Layout >

```
using trisycl::vendor::xilinx::acap::aie::geography< Layout >::layout = Layout
```

Keep the layout used for this CGRA.

Definition at line 35 of file geography.hpp.

9.1.2.5.1.2 layout [2/2] template<typename Layout >

```
using trisycl::vendor::xilinx::acap::aie::geography< Layout >::layout = Layout
```

Keep the layout used for this CGRA.

Definition at line 35 of file geography.hpp.

9.1.2.5.2 Member Function Documentation

9.1.2.5.2.1 cascade_linear_id() [1/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_↩
linear_id (
            int x,
            int y )  [inline], [static], [constexpr]
```

Compute the linearized coordinate of a tile along the cascade stream.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Definition at line 256 of file geography.hpp.

```
00256                                                    {
00257     return x_size*y + ((y & 1) ?  layout::x_max - x :  x);
00258   }
```

References x, trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_size, and trisycl::y.

9.1.2.5.2.2 cascade_linear_id() [2/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_↩
linear_id (
            int x,
            int y )  [inline], [static], [constexpr]
```

Compute the linearized coordinate of a tile along the cascade stream.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Definition at line 256 of file geography.hpp.

```
00256                                                    {
00257     return x_size*y + ((y & 1) ?  layout::x_max - x :  x);
00258   }
```

References x, trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_size, and trisycl::y.

9.1.2.5.2.3 cascade_linear_x() [1/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_↩
linear_x (
            int linear_id )  [inline], [static], [constexpr]
```

Compute the horizontal coordinate of a tile from a cascade linear id.

Parameters

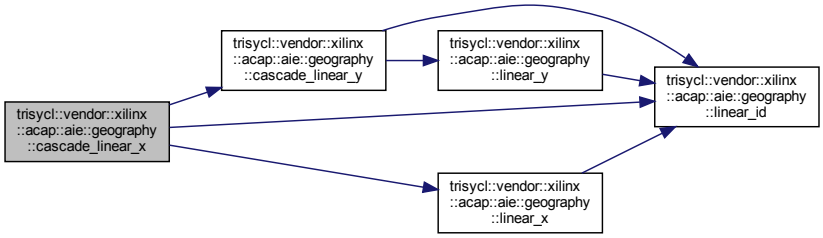
in	*linear↩ _id*	is a linear id along the cascade stream

Definition at line 266 of file geography.hpp.

```
00266                                                    {
00267     auto x = linear_x(linear_id);
00268     // According to the row parity, symmetrize the x coordinate
00269     return (cascade_linear_y(linear_id) & 1) ?  layout::x_max - x :  x;
00270   }
```

References trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_linear_y(), trisycl::vendor::xilinx::acap::aie::geography< trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_x(), and x.

Here is the call graph for this function:



9.1.2.5.2.4 cascade_linear_x() [2/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_↩
linear_x (
            int linear_id )  [inline], [static], [constexpr]
```

Compute the horizontal coordinate of a tile from a cascade linear id.

Parameters

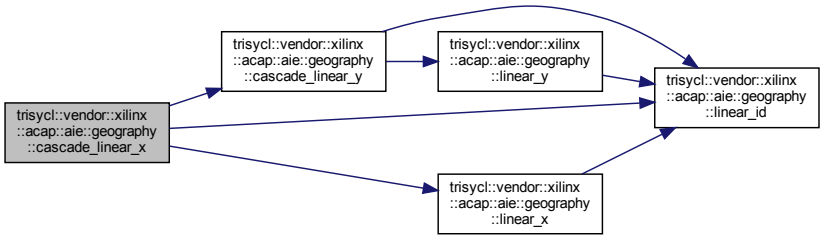
in	*linear↩ _id*	is a linear id along the cascade stream

Definition at line 266 of file geography.hpp.

```
00266                                                    {
00267     auto x = linear_x(linear_id);
00268     // According to the row parity, symmetrize the x coordinate
00269     return (cascade_linear_y(linear_id) & 1) ?  layout::x_max - x :  x;
00270   }
```

References trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_linear_y(), trisycl::vendor::xilinx::acap::aie::geography< trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_x(), and x.

Here is the call graph for this function:



9.1.2.5.2.5 cascade_linear_y() [1/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_↩
linear_y (
            int linear_id )  [inline], [static], [constexpr]
```

Compute the vertical coordinate of a tile from a cascade linear id.

Parameters

in	*linear↩ _id*	is a linear id along the cascade stream

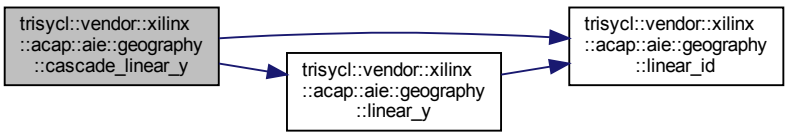
Definition at line 278 of file geography.hpp.

```
00278                                                    {
00279     // Same as the linear id, actually
00280     return linear_y(linear_id);
00281   }
```

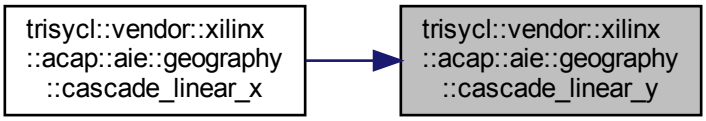
References trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_id(), and trisycl::vendor::xilinx::acap::aie::geography< Layo

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_linear_x().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.5.2.6 cascade_linear_y() [2/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_↩
linear_y (
            int linear_id )  [inline], [static], [constexpr]
```

Compute the vertical coordinate of a tile from a cascade linear id.

Parameters

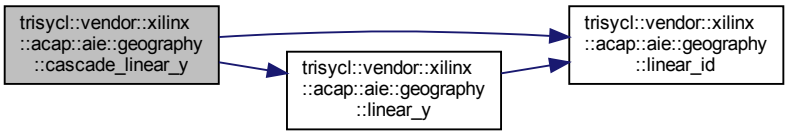
in	*linear↩ _id*	is a linear id along the cascade stream

Definition at line 278 of file geography.hpp.

```
00278                                                    {
00279     // Same as the linear id, actually
00280     return linear_y(linear_id);
00281   }
```

References trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_id(), and trisycl::vendor::xilinx::acap::aie::geography< Layo

Here is the call graph for this function:



9.1.2.5.2.7 generate_tiles() [1/2] template<typename Layout >

```
template<template< int X, int Y > typename Tileable>
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::generate_tiles ( )  [inline],
[static]
```

Generate a tuple by tiling the Tileable object onto the CGRA geometry.

Parameters

Tileable	is the type to tile according to coordinate X and Y

Definition at line 69 of file geography.hpp.

```
00069                                 {
00070     return boost::hana::transform(
00071         tile_coordinates
00072       , [] (auto coord) {
00073           return Tileable<boost::hana::at_c<1>(coord),
00074                           boost::hana::at_c<0>(coord)> {};
00075         }
00076                                   );
00077   }
```

References coord, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::tile_coordinates.

9.1.2.5.2.8 generate_tiles() [2/2] template<typename Layout >

`template<template< int X, int Y > typename Tileable>`
`static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::generate_tiles ( ) [inline], [static]`

Generate a tuple by tiling the Tileable object onto the CGRA geometry.

Parameters

Tileable	is the type to tile according to coordinate X and Y

Definition at line 69 of file geography.hpp.

```
00069                                 {
00070     return boost::hana::transform(
00071         tile_coordinates
00072       , [] (auto coord) {
00073           return Tileable<boost::hana::at_c<1>(coord),
00074                           boost::hana::at_c<0>(coord)> {};
00075         }
00076                                   );
00077   }
```

References coord, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::tile_coordinates.

9.1.2.5.2.9 is_cascade_end() [1/2] template<typename Layout >

`static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_cascade_end (`
`int x,`
`int y ) [inline], [static], [constexpr]`

Test if a tile owns the end of the cascade_stream.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Definition at line 243 of file geography.hpp.

```
00243                                                    {
00244     // The x position depends on the parity of the last line
00245     return x == ((y & 1) ?  x_min :  layout::x_max) && y == layout::y_max;
00246   }
```

References x, trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_min, and trisycl::y.

9.1.2.5.2.10 is_cascade_end() [2/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_cascade_end
(
              int x,
              int y )  [inline], [static], [constexpr]
```

Test if a tile owns the end of the cascade_stream.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Definition at line 243 of file geography.hpp.

```
00243                                                    {
00244     // The x position depends on the parity of the last line
00245     return x == ((y & 1) ?  x_min :  layout::x_max) && y == layout::y_max;
00246   }
```

References x, trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_min, and trisycl::y.

9.1.2.5.2.11 is_cascade_start() [1/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_cascade_↩
start (
              int x,
              int y )  [inline], [static], [constexpr]
```

Test if a tile owns the start of the cascade_stream.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Definition at line 232 of file geography.hpp.

```
00232                                                    {
00233     return x == x_min && y == y_min;
00234   }
```

References x, trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_min, trisycl::y, and trisycl::vendor::xilinx::acap::aie::geography

9.1.2.5.2.12 is_cascade_start() [2/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_cascade_↩
start (
            int x,
            int y )  [inline], [static], [constexpr]
```

Test if a tile owns the start of the cascade_stream.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Definition at line 232 of file geography.hpp.

```
00232                                                                    {
00233     return x == x_min && y == y_min;
00234   }
```

References x, trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_min, trisycl::y, and trisycl::vendor::xilinx::acap::aie::geography

9.1.2.5.2.13 is_east_column() [1/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_east_column
(
            int x )  [inline], [static], [constexpr]
```

Test if a tile is in the Eastern column.

Parameters

in	*x*	is the horizontal tile coordinate

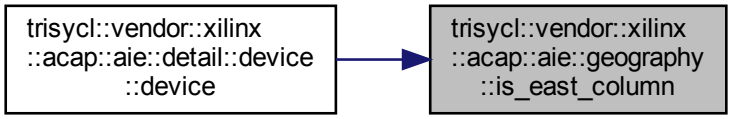
Definition at line 203 of file geography.hpp.

```
00203                                                {
00204     return x == layout::x_max;
00205   }
```

References x.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::device().

Here is the caller graph for this function:



9.1.2.5.2.14 is_east_column() [2/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_east_column
(
            int x )  [inline], [static], [constexpr]
```

Test if a tile is in the Eastern column.

Parameters

in	*x*	is the horizontal tile coordinate

Definition at line 203 of file geography.hpp.

```
00203                                                {
00204     return x == layout::x_max;
00205   }
```

References x.

9.1.2.5.2.15 is_memory_module() [1/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_memory_↩
module (
            int x,
            int y,
            int dx,
            int dy )  [inline], [static], [constexpr]
```

Test if a memory module exists and is connected to a tile.

Parameters

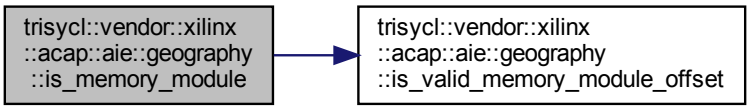
in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate
in	*dx*	is the horizontal displacement (-1,0,+1)
in	*dy*	is the vertical displacement (-1,0,+1)

Definition at line 312 of file geography.hpp.

```
00312                                                                    {
00313     if (!is_valid_memory_module_offset(dx, dy))
00314       return false;
00315     if (y == y_min && dy == -1)
00316       // No memory module on the South of the first (Southern) line
00317       return false;
00318     else if (y == layout::y_max && dy == 1)
00319       // No memory module on the North of the last (Northern) line
00320       return false;
00321     else if (x == x_min && !(y & 1) && (dx == -1))
00322       // No memory module on the West of the first (Western) column on
00323       // even lines
00324       return false;
00325     else if (x == layout::x_max && (y & 1) && (dx == 1))
00326       // No memory module on the East of the last (Eastern) column on
00327       // odd lines
00328       return false;
00329     else return true;
00330   }
```

References trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_valid_memory_module_offset(), x, trisycl::vendor::xilinx::acap::
trisycl::y, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::y_min.

Here is the call graph for this function:



9.1.2.5.2.16 is_memory_module() [2/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_memory_↩
module (
            int x,
            int y,
            int dx,
            int dy )  [inline], [static], [constexpr]
```

Test if a memory module exists and is connected to a tile.

Parameters

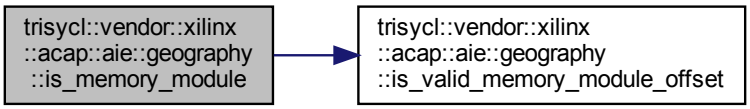
in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate
in	*dx*	is the horizontal displacement (-1,0,+1)
in	*dy*	is the vertical displacement (-1,0,+1)

Definition at line 312 of file geography.hpp.

```
00312                                                                          {
00313     if (!is_valid_memory_module_offset(dx, dy))
00314       return false;
00315     if (y == y_min && dy == -1)
00316       // No memory module on the South of the first (Southern) line
00317       return false;
00318     else if (y == layout::y_max && dy == 1)
00319       // No memory module on the North of the last (Northern) line
00320       return false;
00321     else if (x == x_min && !(y & 1) && (dx == -1))
00322       // No memory module on the West of the first (Western) column on
00323       // even lines
00324       return false;
00325     else if (x == layout::x_max && (y & 1) && (dx == 1))
00326       // No memory module on the East of the last (Eastern) column on
00327       // odd lines
00328       return false;
00329     else return true;
00330   }
```

References trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_valid_memory_module_offset(), x, trisycl::vendor::xilinx::acap::
trisycl::y, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::y_min.

Here is the call graph for this function:



9.1.2.5.2.17 is_north_row() [1/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_north_row (
            int y )  [inline], [static], [constexpr]
```

Test if a tile is in the Northern row.

Parameters

in	*y*	is the vertical tile coordinate

Definition at line 221 of file geography.hpp.

```
00221                                                {
00222     return y == layout::y_max;
00223   }
```

References trisycl::y.

9.1.2.5.2.18 is_north_row() [2/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_north_row (
            int y )  [inline], [static], [constexpr]
```

Test if a tile is in the Northern row.

Parameters

in	*y*	is the vertical tile coordinate

Definition at line 221 of file geography.hpp.

```
00221                                                {
00222     return y == layout::y_max;
00223   }
```

References trisycl::y.

9.1.2.5.2.19 is_south_row() [1/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_south_row (
            int y )  [inline], [static], [constexpr]
```

Test if a tile is in the Southern row.

Parameters

in	*y*	is the vertical tile coordinate

Definition at line 212 of file geography.hpp.

```
00212                                          {
00213     return y == y_min;
00214   }
```

References trisycl::y, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::y_min.

9.1.2.5.2.20 is_south_row() [2/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_south_row (
            int y )  [inline], [static], [constexpr]
```

Test if a tile is in the Southern row.

Parameters

in	*y*	is the vertical tile coordinate

Definition at line 212 of file geography.hpp.

```
00212                                          {
00213     return y == y_min;
00214   }
```

References trisycl::y, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::y_min.

9.1.2.5.2.21 is_valid_memory_module_offset() [1/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_valid_↩
memory_module_offset (
            int dx,
            int dy )  [inline], [static], [constexpr]
```

Test if a memory module coordinate offset is plausible.

1 of the displacement is 0 and the other one is -1 or 1

Parameters

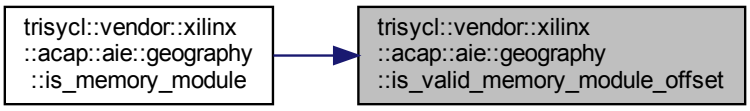
in	*dx*	is the horizontal displacement (-1,0,+1)
in	*dy*	is the vertical displacement (-1,0,+1)

Definition at line 292 of file geography.hpp.

```
00292                                                                    {
00293     if (dx == 0)
00294       // Cannot use std::abs() because it is not constexpr in C++17
00295       return dy == -1 || dy == 1;
00296     else if (dy == 0)
00297       return dx == -1 || dx == 1;
00298     else return false;
00299   }
```

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_memory_module().

Here is the caller graph for this function:



9.1.2.5.2.22 is_valid_memory_module_offset() [2/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_valid_↩
memory_module_offset (
              int dx,
              int dy )  [inline], [static], [constexpr]
```

Test if a memory module coordinate offset is plausible.

1 of the displacement is 0 and the other one is -1 or 1

Parameters

in	*dx*	is the horizontal displacement (-1,0,+1)
in	*dy*	is the vertical displacement (-1,0,+1)

Definition at line 292 of file geography.hpp.

```
00292                                                                    {
00293     if (dx == 0)
00294       // Cannot use std::abs() because it is not constexpr in C++17
00295       return dy == -1 || dy == 1;
00296     else if (dy == 0)
00297       return dx == -1 || dx == 1;
00298     else return false;
00299   }
```

9.1.2.5.2.23 is_west_column() [1/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_west_column
(
              int x )  [inline], [static], [constexpr]
```

Test if a tile is in the Western column.

Parameters

in	*x*	is the horizontal tile coordinate

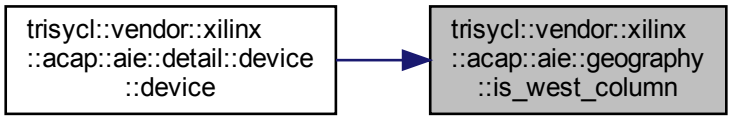
Definition at line 194 of file geography.hpp.

```
00194                                                {
00195     return x == x_min;
00196   }
```

References x, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_min.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::device().

Here is the caller graph for this function:



9.1.2.5.2.24 is_west_column() [2/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_west_column
(
              int x )  [inline], [static], [constexpr]
```

Test if a tile is in the Western column.

Parameters

in	*x*	is the horizontal tile coordinate

Definition at line 194 of file geography.hpp.

```
00194                                                {
00195     return x == x_min;
00196   }
```

References x, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_min.

9.1.2.5.2.25 is_x_valid() [1/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_valid (
              int x )  [inline], [static], [constexpr]
```

Test if an horizontal coordinate is valid.

Parameters

in	*x*	is the horizontal tile coordinate

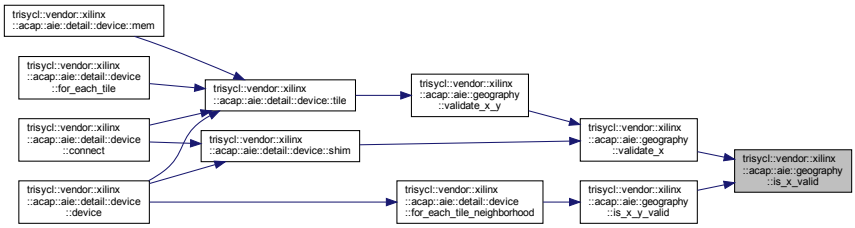
Definition at line 113 of file geography.hpp.

```
00113                                                {
00114     return x_min <= x && x <= layout::x_max;
00115   }
```

References x, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_min.

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_y_valid(), and trisycl::vendor::xilinx::acap::aie::geography

Here is the caller graph for this function:



9.1.2.5.2.26 is_x_valid() [2/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_valid (
            int x )  [inline], [static], [constexpr]
```

Test if an horizontal coordinate is valid.

Parameters

in	*x*	is the horizontal tile coordinate

Definition at line 113 of file geography.hpp.

```
00113                                                {
00114     return x_min <= x && x <= layout::x_max;
00115   }
```

References x, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_min.

9.1.2.5.2.27 is_x_y_valid() [1/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_y_valid (
            int x,
            int y )  [inline], [static], [constexpr]
```

Test if horizontal and vertical coordinates are valid.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

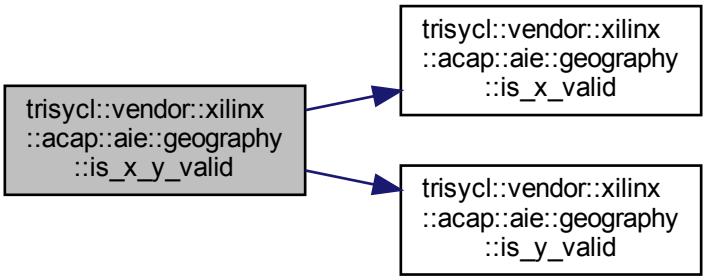
Definition at line 169 of file geography.hpp.

```
00169                                                    {
00170     return is_x_valid(x) && is_y_valid(y);
00171   }
```

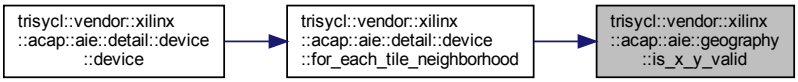
References trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_valid(), trisycl::vendor::xilinx::acap::aie::geography< Layout x, and trisycl::y.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_neighborhood().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.5.2.28 is_x_y_valid() [2/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_y_valid (
            int x,
            int y )  [inline], [static], [constexpr]
```

Test if horizontal and vertical coordinates are valid.

Parameters

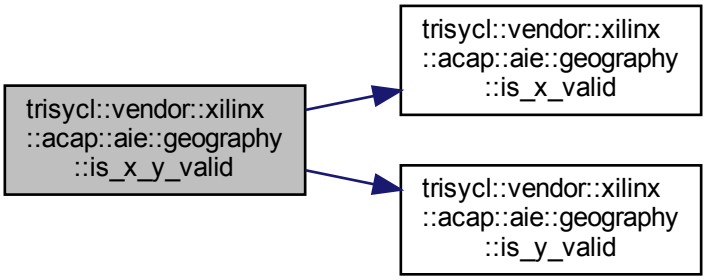
in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Definition at line 169 of file geography.hpp.

```
00169                                                   {
00170     return is_x_valid(x) && is_y_valid(y);
00171   }
```

References trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_valid(), trisycl::vendor::xilinx::acap::aie::geography< Layout x, and trisycl::y.

Here is the call graph for this function:



9.1.2.5.2.29 is_y_valid() [1/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_y_valid (
            int y )  [inline], [static], [constexpr]
```

Test if a vertical coordinate is valid.

Parameters

in	*y*	is the vertical tile coordinate

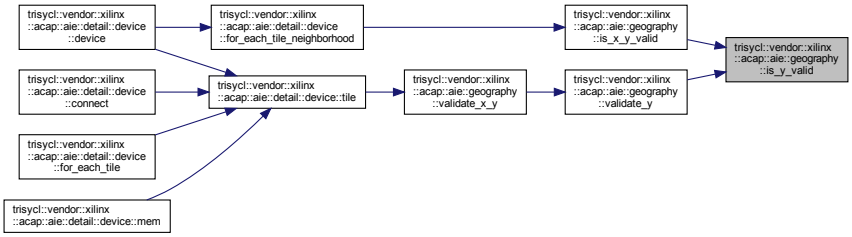
Definition at line 140 of file geography.hpp.

```
00140                                           {
00141     return y_min <= y && y <= layout::y_max;
00142   }
```

References trisycl::y, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::y_min.

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_y_valid(), and trisycl::vendor::xilinx::acap::aie::geography

Here is the caller graph for this function:



9.1.2.5.2.30 is_y_valid() [2/2] template<typename Layout >

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_y_valid (
            int y )  [inline], [static], [constexpr]
```

Test if a vertical coordinate is valid.

Parameters

in	*y*	is the vertical tile coordinate

Definition at line 140 of file geography.hpp.

```
00140                                          {
00141     return y_min <= y && y <= layout::y_max;
00142   }
```

References trisycl::y, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::y_min.

9.1.2.5.2.31 linear_id() [1/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_id (
            int x,
            int y )  [inline], [static], [constexpr]
```

Compute the linearized coordinate of a tile.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

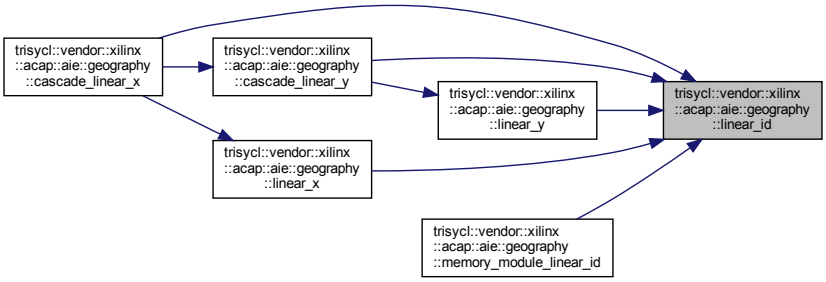
Definition at line 86 of file geography.hpp.

```
00086                                                {
00087     return x + x_size*y;
00088   }
```

References x, trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_size, and trisycl::y.

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_linear_x(), trisycl::vendor::xilinx::acap::aie::geography
trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_x(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_y(),
and trisycl::vendor::xilinx::acap::aie::geography< Layout >::memory_module_linear_id().

Here is the caller graph for this function:



9.1.2.5.2.32 linear_id() [2/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_id (
            int x,
            int y )  [inline], [static], [constexpr]
```

Compute the linearized coordinate of a tile.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Definition at line 86 of file geography.hpp.

```
00086                                                {
00087     return x + x_size*y;
00088   }
```

References x, trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_size, and trisycl::y.

9.1.2.5.2.33 linear_x() [1/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_x (
            int linear_id )  [inline], [static], [constexpr]
```

Compute the horizontal coordinate of a tile from linear id.

Parameters

in	*linear↩ _id*	is a linearized coordinate

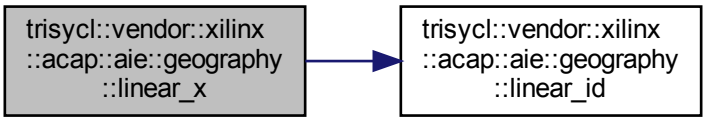
Definition at line 95 of file geography.hpp.

```
00095                                                {
00096     return linear_id%x_size;
00097   }
```

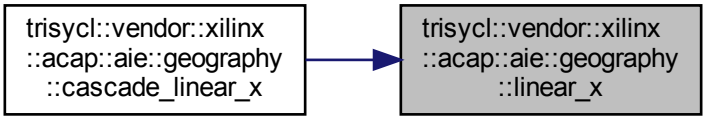
References trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_id(), and trisycl::vendor::xilinx::acap::aie::geography< Layo

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_linear_x().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.5.2.34 linear_x() [2/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_x (
              int linear_id )  [inline], [static], [constexpr]
```

Compute the horizontal coordinate of a tile from linear id.

Parameters

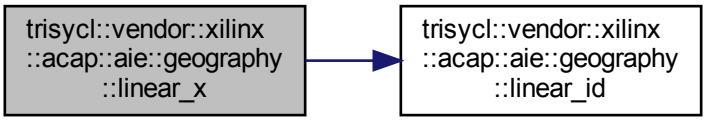
in	*linear↩ _id*	is a linearized coordinate

Definition at line 95 of file geography.hpp.

```
00095                                                {
00096     return linear_id%x_size;
00097   }
```

References trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_id(), and trisycl::vendor::xilinx::acap::aie::geography< Layo

Here is the call graph for this function:



9.1.2.5.2.35 linear_y() [1/2]

```
template<typename Layout >
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_y (
            int linear_id ) [inline], [static], [constexpr]
```

Compute the vertical coordinate of a tile from a linear id.

Parameters

in	*linear↩ _id*	is a linearized coordinate

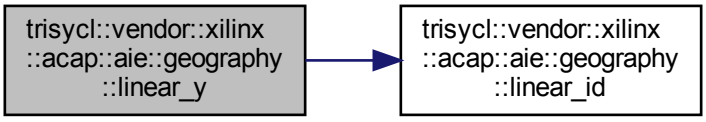
Definition at line 104 of file geography.hpp.

```
00104                                                    {
00105     return linear_id/x_size;
00106   }
```

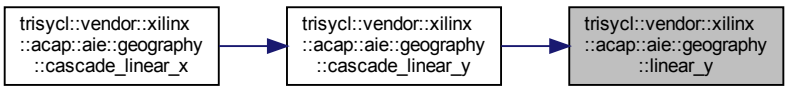
References trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_id(), and trisycl::vendor::xilinx::acap::aie::geography< Layo

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_linear_y().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.5.2.36 linear_y() [2/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_y (
            int linear_id ) [inline], [static], [constexpr]
```

Compute the vertical coordinate of a tile from a linear id.

Parameters

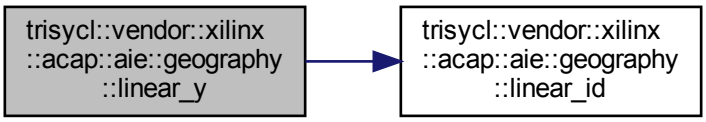
in	*linear_id*	is a linearized coordinate

Definition at line 104 of file geography.hpp.

```
00104                                                    {
00105     return linear_id/x_size;
00106   }
```

References trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_id(), and trisycl::vendor::xilinx::acap::aie::geography< Layo

Here is the call graph for this function:



9.1.2.5.2.37 memory_module_dx_coordinate() [1/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::memory_module_dx_coordinate (
            int y,
            int dx ) [inline], [static], [constexpr]
```

Compute the memory module x coordinate delta from its tile.

Parameters

in	*y*	is the vertical tile coordinate
in	*dx*	is the horizontal displacement (-1,0,+1)

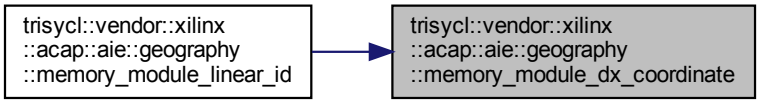
Definition at line 339 of file geography.hpp.

```
00339                                                                    {
00340     // Mimic Figure 2-20 on p.  65
00341     return (dx - 1)/2 + (y & 1);
00342   }
```

References trisycl::y.

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::memory_module_linear_id().

Here is the caller graph for this function:



9.1.2.5.2.38 memory_module_dx_coordinate() [2/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::memory_module↩
_dx_coordinate (
             int y,
             int dx )  [inline], [static], [constexpr]
```

Compute the memory module x coordinate delta from its tile.

Parameters

in	*y*	is the vertical tile coordinate
in	*dx*	is the horizontal displacement (-1,0,+1)

Definition at line 339 of file geography.hpp.

```
00339                                                                    {
00340     // Mimic Figure 2-20 on p.  65
00341     return (dx - 1)/2 + (y & 1);
00342   }
```

References trisycl::y.

9.1.2.5.2.39 memory_module_linear_id() [1/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::memory_module↩
_linear_id (
             int x,
             int y,
             int dx,
             int dy )  [inline], [static], [constexpr]
```

Compute the linear id of a memory module attached to a tile.

Parameters

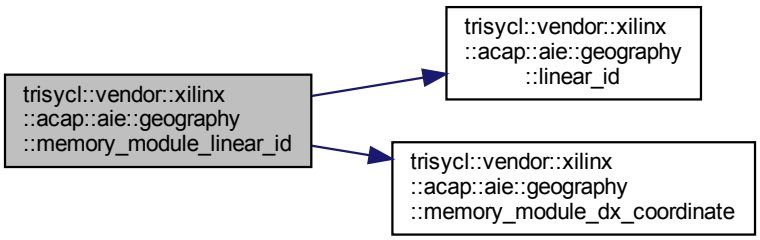
in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate
in	*dx*	is the horizontal displacement (-1,0,+1)
in	*dy*	is the vertical displacement (-1,0,+1)

Definition at line 355 of file geography.hpp.

```
00355                                                                                {
00356     if (dx == 0)
00357       return linear_id(x, y + dy);
00358     else
00359       // Take into account the horizontal chessboard-like skewing
00360       return linear_id(x + memory_module_dx_coordinate(y, dx), y);
00361   }
```

References trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_id(), trisycl::vendor::xilinx::acap::aie::geography< Layout >
x, and trisycl::y.

Here is the call graph for this function:



9.1.2.5.2.40 memory_module_linear_id() [2/2] template<typename Layout >

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::memory_module↩
_linear_id (
             int x,
             int y,
             int dx,
             int dy )  [inline], [static], [constexpr]
```

Compute the linear id of a memory module attached to a tile.

Parameters

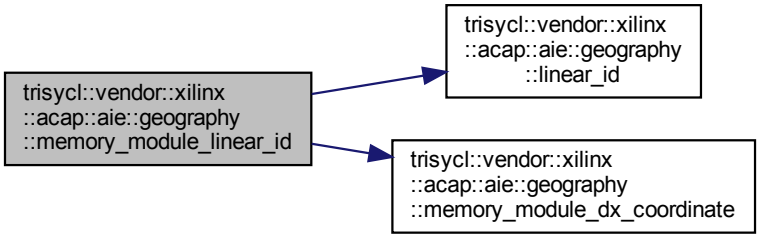
in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate
in	*dx*	is the horizontal displacement (-1,0,+1)
in	*dy*	is the vertical displacement (-1,0,+1)

Definition at line 355 of file geography.hpp.

```
00355                                                                                {
00356     if (dx == 0)
00357       return linear_id(x, y + dy);
00358     else
00359       // Take into account the horizontal chessboard-like skewing
00360       return linear_id(x + memory_module_dx_coordinate(y, dx), y);
00361   }
```

References trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_id(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::memory_module_dx_coordinate(), trisycl::x, and trisycl::y.

Here is the call graph for this function:



9.1.2.5.2.41 validate_x() [1/2] template<typename Layout >

```
static void trisycl::vendor::xilinx::acap::aie::geography< Layout >::validate_x (
              int x )  [inline], [static]
```

Validate an horizontal coordinate.

Do nothing but throwing if the coordinate is not a valid tile coordinate.

Parameters

in	*x*	is the horizontal tile coordinate

Exceptions

trisycl::runtime_error	if the coordinate is invalid

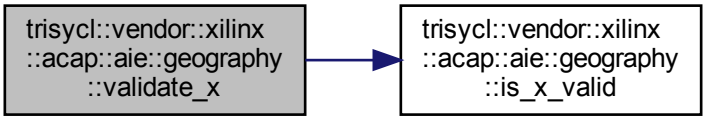
Definition at line 127 of file geography.hpp.

```
00127                                  {
00128     if (!is_x_valid(x)) {
00129       throw ::trisycl::runtime_error {
00130         (boost::format { "The x coordinate %1% is not between %2% and %3%" }
00131           % x % x_min % layout::x_max).str() };
00132     }
00133   }
```

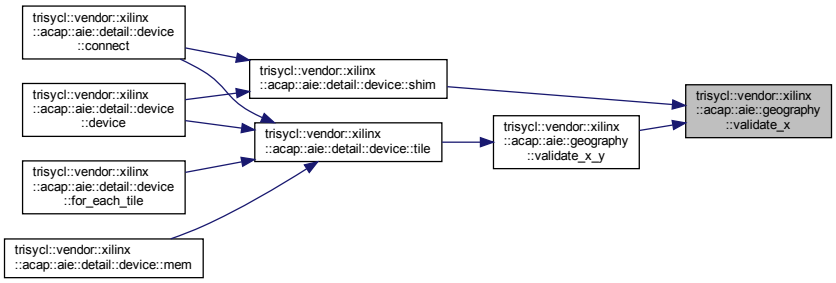
References trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_valid(), x, and trisycl::vendor::xilinx::acap::aie::geography< L

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::shim(), and trisycl::vendor::xilinx::acap::aie::geography< La

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.5.2.42 validate_x() [2/2] template<typename Layout >

```
static void trisycl::vendor::xilinx::acap::aie::geography< Layout >::validate_x (
              int x )  [inline], [static]
```

Validate an horizontal coordinate.

Do nothing but throwing if the coordinate is not a valid tile coordinate.

Parameters

in	*x*	is the horizontal tile coordinate

Exceptions

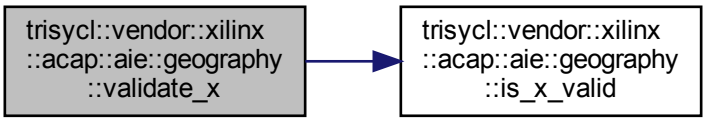
trisycl::runtime_error	if the coordinate is invalid

Definition at line 127 of file geography.hpp.

```
00127                                  {
00128     if (!is_x_valid(x)) {
00129       throw ::trisycl::runtime_error {
00130         (boost::format { "The x coordinate %1% is not between %2% and %3%" }
00131           % x % x_min % layout::x_max).str() };
00132     }
00133   }
```

References trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_valid(), x, and trisycl::vendor::xilinx::acap::aie::geography< L

Here is the call graph for this function:



9.1.2.5.2.43 validate_x_y() [1/2] template<typename Layout >

```
static void trisycl::vendor::xilinx::acap::aie::geography< Layout >::validate_x_y (
              int x,
              int y )  [inline], [static]
```

Validate the horizontal and vertical coordinates.

Do nothing but throwing if the coordinates are not valid tile coordinates.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Exceptions

trisycl::runtime_error	if the coordinate is invalid

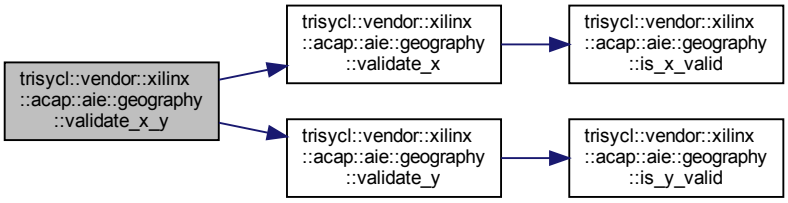
Definition at line 184 of file geography.hpp.

```
00184                                                  {
00185     validate_x(x);
00186     validate_y(y);
00187   }
```

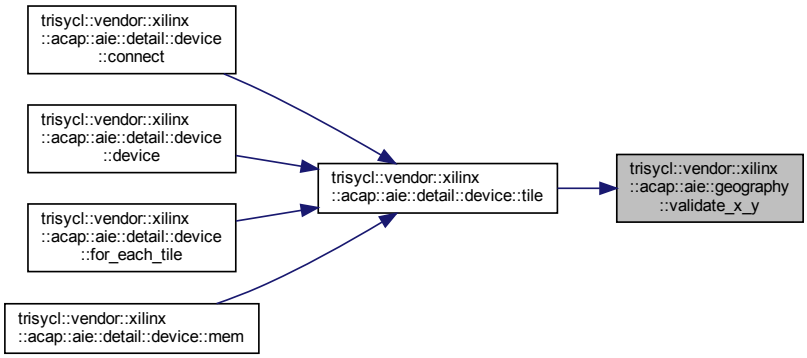
References trisycl::vendor::xilinx::acap::aie::geography< Layout >::validate_x(), trisycl::vendor::xilinx::acap::aie::geography< Layout x, and trisycl::y.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::tile().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.5.2.44 validate_x_y() [2/2] template<typename Layout >

```
static void trisycl::vendor::xilinx::acap::aie::geography< Layout >::validate_x_y (
            int x,
            int y )  [inline], [static]
```

Validate the horizontal and vertical coordinates.

Do nothing but throwing if the coordinates are not valid tile coordinates.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Exceptions

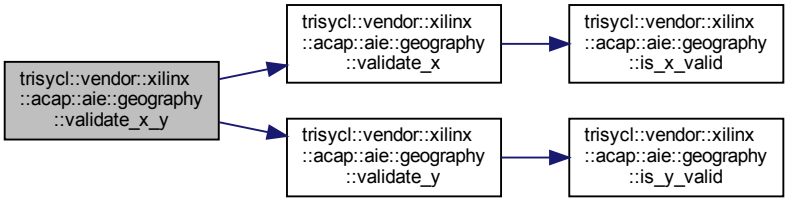
trisycl::runtime_error	if the coordinate is invalid

Definition at line 184 of file geography.hpp.

```
00184                                                {
00185     validate_x(x);
00186     validate_y(y);
00187   }
```

References trisycl::vendor::xilinx::acap::aie::geography< Layout >::validate_x(), trisycl::vendor::xilinx::acap::aie::geography< Layout x, and trisycl::y.

Here is the call graph for this function:



9.1.2.5.2.45 validate_y() [1/2] template<typename Layout >

```
static void trisycl::vendor::xilinx::acap::aie::geography< Layout >::validate_y (
            int y )  [inline], [static]
```

Validate a vertical coordinate.

Do nothing but throwing if the coordinate is not a valid tile coordinate.

Parameters

in	*y*	is the vertical tile coordinate

Exceptions

trisycl::runtime_error	if the coordinate is invalid

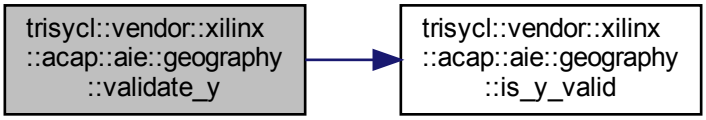
Definition at line 154 of file geography.hpp.

```
00154                                                {
00155     if (!is_y_valid(y)) {
00156       throw ::trisycl::runtime_error {
00157         (boost::format { "The y coordinate %1% is not between %2% and %3%" }
00158           % y % y_min % layout::y_max).str() };
00159     }
00160   }
```

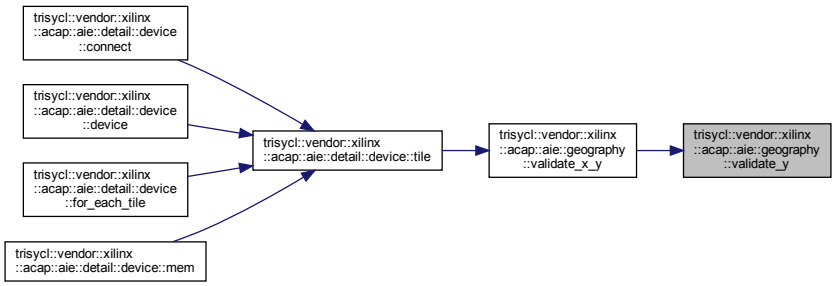
References trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_y_valid(), trisycl::y, and trisycl::vendor::xilinx::acap::aie::geogra

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::validate_x_y().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.5.2.46 validate_y() [2/2] template<typename Layout >

```
static void trisycl::vendor::xilinx::acap::aie::geography< Layout >::validate_y (
            int y )  [inline], [static]
```

Validate a vertical coordinate.

Do nothing but throwing if the coordinate is not a valid tile coordinate.

Parameters

in	*y*	is the vertical tile coordinate

Exceptions

trisycl::runtime_error	if the coordinate is invalid

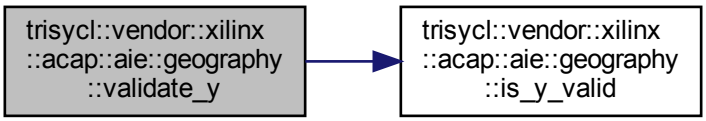
Definition at line 154 of file geography.hpp.

```
00154                                   {
```

```
00155     if (!is_y_valid(y)) {
00156       throw ::trisycl::runtime_error {
00157         (boost::format { "The y coordinate %1% is not between %2% and %3%" }
00158           % y % y_min % layout::y_max).str() };
00159     }
00160   }
```

References trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_y_valid(), trisycl::y, and trisycl::vendor::xilinx::acap::aie::geogra

Here is the call graph for this function:



9.1.2.5.3 Member Data Documentation

9.1.2.5.3.1 size `template<typename Layout >`

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::size = x_size*y_size
[static], [constexpr]
```

Total number of tiles in the CGRA.

Definition at line 50 of file geography.hpp.

9.1.2.5.3.2 tile_coordinates `template<typename Layout >`

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::tile_coordinates
[static], [constexpr]
```

Initial value:

```
= boost::hana::cartesian_product(
    boost::hana::make_tuple(
        boost::hana::range_c<int, y_min, y_size>
      , boost::hana::range_c<int, x_min, x_size>
                            )
                                                                                )
```

A tuple with the coordinate tuples of all the tiles inside the CGRA geometry.

Definition at line 55 of file geography.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::generate_tiles().

9.1.2.5.3.3 x_min `template<typename Layout >`

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_min = 0
[static], [constexpr]
```

Minimum horizontal coordinate.

Definition at line 38 of file geography.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_cascade_end(), trisycl::vendor::xilinx::acap::aie::geography
trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_memory_module(), trisycl::vendor::xilinx::acap::aie::geography< Layout >
trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_valid(), and trisycl::vendor::xilinx::acap::aie::geography< Layout >::valid

9.1.2.5.3.4 x_size `template<typename Layout >`

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_size = layout↩
::x_max + 1  [static], [constexpr]
```

Number of tile horizontally in the CGRA.

Definition at line 44 of file geography.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_linear_id(), trisycl::vendor::xilinx::acap::aie::detail::de
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_x_index(), trisycl::vendor::xilinx::acap::aie::geography< Layo
trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_x(), and trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_

9.1.2.5.3.5 y_min `template<typename Layout >`

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::y_min = 0
[static], [constexpr]
```

Minimum vertical coordinate.

Definition at line 41 of file geography.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_cascade_start(), trisycl::vendor::xilinx::acap::aie::geography
trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_south_row(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_y_
and trisycl::vendor::xilinx::acap::aie::geography< Layout >::validate_y().

9.1.2.5.3.6 y_size `template<typename Layout >`

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::y_size = layout↩
::y_max + 1  [static], [constexpr]
```

Number of tile vertically in the CGRA.

Definition at line 47 of file geography.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_index(), and trisycl::vendor::xilinx::acap::aie::d

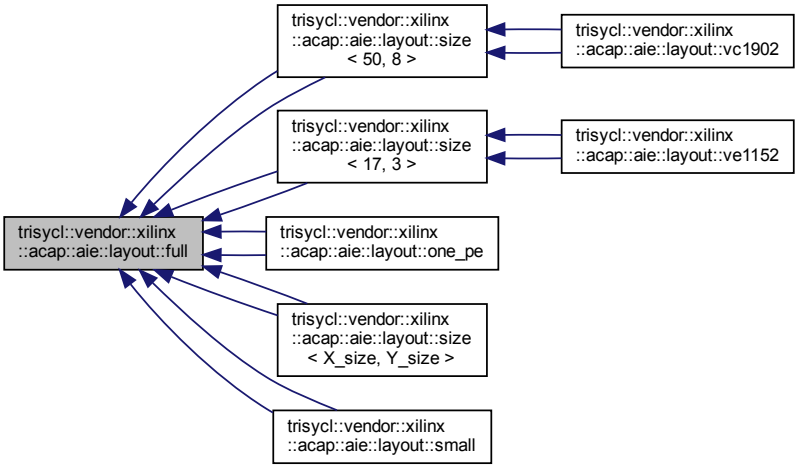
9.1.2.6 struct trisycl::vendor::xilinx::acap::aie::layout::full

Describe the layout of a "big" AI Engine array.

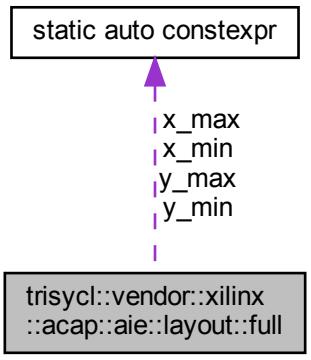
Use some constrains from Figure 2-8 "ME Array Address Map Example", p. 44

Definition at line 26 of file full.hpp.

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::layout::full:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::layout::full:



Static Public Member Functions

- static bool constexpr is_noc_tile (int x, int y)
- static bool constexpr is_pl_tile (int x, int y)
- static bool constexpr is_noc_tile (int x, int y)
- static bool constexpr is_pl_tile (int x, int y)

Static Public Attributes

- static auto constexpr x_min = 0
- static auto constexpr x_max = 5
- static auto constexpr y_min = 0
- static auto constexpr y_max = 4

9.1.2.6.1 Member Function Documentation

9.1.2.6.1.1 is_noc_tile() [1/2]

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::layout↩
::full::is_noc_tile (
            int x,
            int y )  [inline], [static], [constexpr]
```

Definition at line 32 of file full.hpp.

```
00032                                                {
00033     return y == 0 && 2 <= x && x <= 3;
00034   }
```

References x, and trisycl::y.

9.1.2.6.1.2 is_noc_tile() [2/2]

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::layout↩
::full::is_noc_tile (
            int x,
            int y )  [inline], [static], [constexpr]
```

Definition at line 32 of file full.hpp.

```
00032                                                {
00033     return y == 0 && 2 <= x && x <= 3;
00034   }
```

References x, and trisycl::y.

9.1.2.6.1.3 is_pl_tile() [1/2]

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::layout↩
::full::is_pl_tile (
            int x,
            int y )  [inline], [static], [constexpr]
```

Definition at line 36 of file full.hpp.

```
00036                                                 {
00037     return y == 0 && ((0 <= x && x <= 1) || (4 <= x && x <= 5));
00038   }
```

References x, and trisycl::y.

9.1.2.6.1.4 is_pl_tile() [2/2] static bool constexpr trisycl::vendor::xilinx::acap::aie::layout↩ ::full::is_pl_tile (int x, int y) [inline], [static], [constexpr]

Definition at line 36 of file full.hpp.

```
00036                                                {
00037     return y == 0 && ((0 <= x && x <= 1) || (4 <= x && x <= 5));
00038   }
```

References x, and trisycl::y.

9.1.2.6.2 Member Data Documentation

9.1.2.6.2.1 x_max static auto constexpr trisycl::vendor::xilinx::acap::aie::layout::full::x_↩ max = 5 [static], [constexpr]

Definition at line 28 of file full.hpp.

9.1.2.6.2.2 x_min static auto constexpr trisycl::vendor::xilinx::acap::aie::layout::full::x_min = 0 [static], [constexpr]

Definition at line 27 of file full.hpp.

9.1.2.6.2.3 y_max static auto constexpr trisycl::vendor::xilinx::acap::aie::layout::full::y_↩ max = 4 [static], [constexpr]

Definition at line 30 of file full.hpp.

9.1.2.6.2.4 y_min static auto constexpr trisycl::vendor::xilinx::acap::aie::layout::full::y_min = 0 [static], [constexpr]

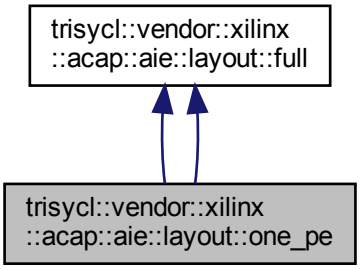
Definition at line 29 of file full.hpp.

9.1.2.7 struct trisycl::vendor::xilinx::acap::aie::layout::one_pe

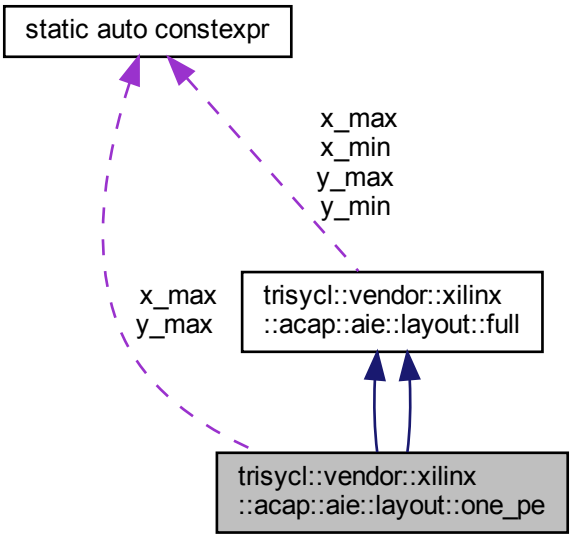
Describe the layout of a tiny AI Engine with only 1 tile.

Definition at line 23 of file one_pe.hpp.

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::layout::one_pe:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::layout::one_pe:



Static Public Attributes

- static auto constexpr x_max = 0
- static auto constexpr y_max = 0

Additional Inherited Members

9.1.2.7.1 Member Data Documentation

9.1.2.7.1.1 x_max `static auto constexpr trisycl::vendor::xilinx::acap::aie::layout::one_pe↩ ::x_max = 0 [static], [constexpr]`

Definition at line 24 of file one_pe.hpp.

9.1.2.7.1.2 y_max `static auto constexpr trisycl::vendor::xilinx::acap::aie::layout::one_pe↩ ::y_max = 0 [static], [constexpr]`

Definition at line 25 of file one_pe.hpp.

9.1.2.8 struct trisycl::vendor::xilinx::acap::aie::layout::size

template<int X_size, int Y_size>
struct trisycl::vendor::xilinx::acap::aie::layout::size< X_size, Y_size >

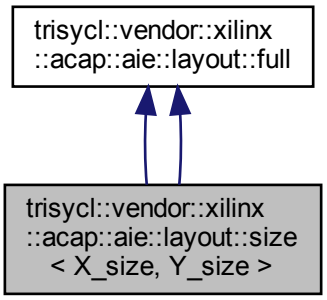
Fully specify the size of an AI Engine layout.

Parameters

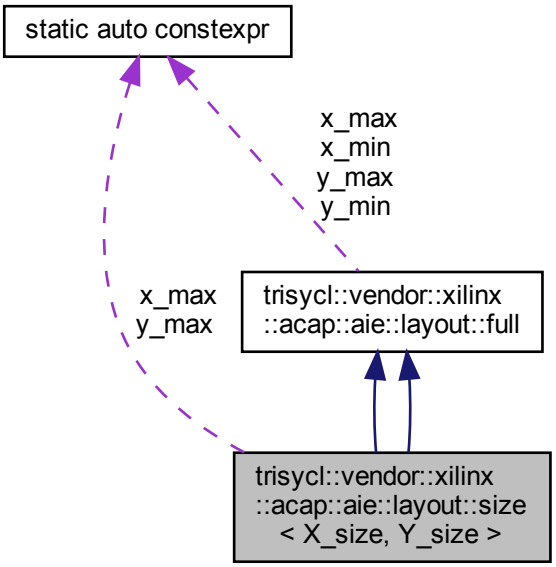
X_size	is the number of tile horizontally
Y_size	is the number of tile vertically

Definition at line 30 of file size.hpp.

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::layout::size< X_size, Y_size >:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::layout::size< X_size, Y_size >:



Static Public Attributes

- static auto constexpr x_max = X_size - 1
  Maximum horizontal coordinate.
- static auto constexpr y_max = Y_size - 1
  Maximum vertical coordinate.

Additional Inherited Members

9.1.2.8.1 Member Data Documentation

9.1.2.8.1.1 x_max `template<int X_size, int Y_size>`

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::layout::size< X_size, Y_size >::x_↩
max = X_size - 1  [static], [constexpr]
```

Maximum horizontal coordinate.

Definition at line 34 of file size.hpp.

9.1.2.8.1.2 y_max `template<int X_size, int Y_size>`

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::layout::size< X_size, Y_size >::y_↩
max = Y_size - 1  [static], [constexpr]
```

Maximum vertical coordinate.

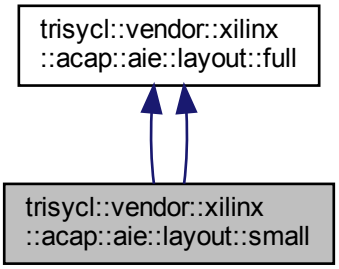
Definition at line 37 of file size.hpp.

9.1.2.9 struct trisycl::vendor::xilinx::acap::aie::layout::small

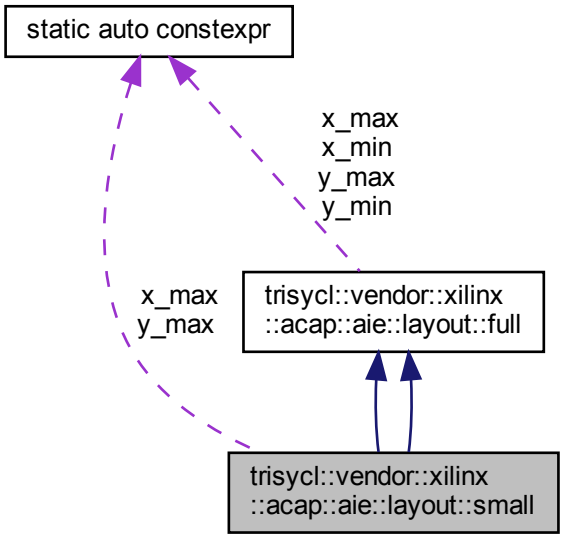
Describe the layout of a small AI Engine array with only a few PE.

Definition at line 23 of file small.hpp.

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::layout::small:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::layout::small:



Static Public Attributes

- static auto constexpr x_max = 1
- static auto constexpr y_max = 2

Additional Inherited Members

9.1.2.9.1 Member Data Documentation

9.1.2.9.1.1 x_max `static auto constexpr trisycl::vendor::xilinx::acap::aie::layout::small::x_max = 1 [static], [constexpr]`

Definition at line 24 of file small.hpp.

9.1.2.9.1.2 y_max `static auto constexpr trisycl::vendor::xilinx::acap::aie::layout::small::y_max = 2 [static], [constexpr]`

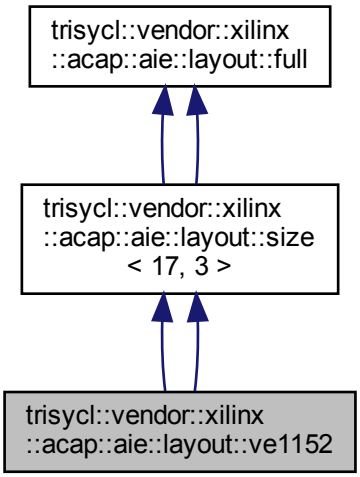
Definition at line 25 of file small.hpp.

9.1.2.10 struct trisycl::vendor::xilinx::acap::aie::layout::ve1152

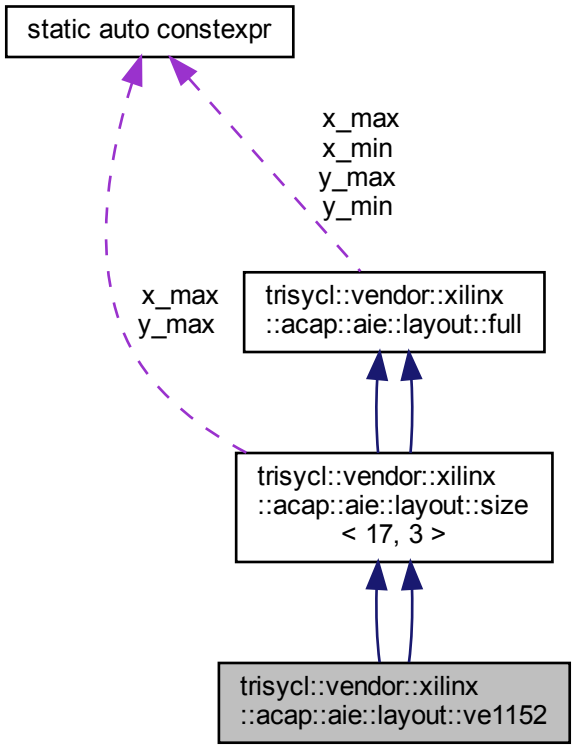
Describe the layout of a Versal VE1152 device

Definition at line 24 of file versal.hpp.

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::layout::ve1152:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::layout::ve1152:



Additional Inherited Members

9.1.2.11 struct trisycl::vendor::xilinx::acap::aie::lock_unit

The lock infrastructure used by AI Engine memory modules and shim tiles.

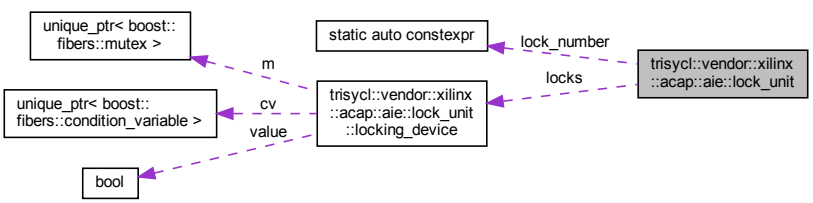
Based on Math Engine (ME) Architecture Specification, Revision v1.5 June 2018

4.4.6 Lock Interface, p. 115

4.7 Lock Unit, p. 129

Definition at line 37 of file lock.hpp.

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::lock_unit:



Public Types

- using lock_id_t = decltype(lock_number)
- using lock_id_t = decltype(lock_number)

Public Member Functions

- auto & lock (int i)

  Get the requested lock.
- auto & lock (int i)

  Get the requested lock.

Public Attributes

- locking_device locks [lock_number]

  The locking units of the locking device.

Static Public Attributes

- static auto constexpr lock_number = 16

9.1.2.11.1 Member Typedef Documentation

9.1.2.11.1.1 lock_id_t [1/2] `using trisycl::vendor::xilinx::acap::aie::lock_unit::lock_id_t = decltype(lock_number)`

Definition at line 41 of file lock.hpp.

9.1.2.11.1.2 lock_id_t [2/2] `using trisycl::vendor::xilinx::acap::aie::lock_unit::lock_id_t = decltype(lock_number)`

Definition at line 41 of file lock.hpp.

9.1.2.11.2 Member Function Documentation

9.1.2.11.2.1 lock() [1/2] auto & trisycl::vendor::xilinx::acap::aie::lock_unit::lock (int i) [inline]

Get the requested lock.

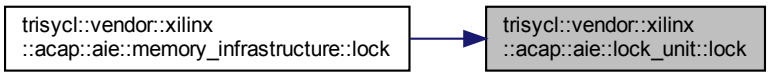
Definition at line 99 of file lock.hpp.

```
00099                     {
00100     assert(0 <= i && i < lock_number);
00101     return locks[i];
00102   }
```

References i, lock_number, and locks.

Referenced by trisycl::vendor::xilinx::acap::aie::memory_infrastructure::lock().

Here is the caller graph for this function:



9.1.2.11.2.2 lock() [2/2] auto & trisycl::vendor::xilinx::acap::aie::lock_unit::lock (int i) [inline]

Get the requested lock.

Definition at line 99 of file lock.hpp.

```
00099                     {
00100     assert(0 <= i && i < lock_number);
00101     return locks[i];
00102   }
```

References i, lock_number, and locks.

9.1.2.11.3 Member Data Documentation

9.1.2.11.3.1 lock_number static auto constexpr trisycl::vendor::xilinx::acap::aie::lock_unit↩ ::lock_number = 16 [static], [constexpr]

Definition at line 39 of file lock.hpp.

Referenced by lock().

9.1.2.11.3.2 locks `locking_device trisycl::vendor::xilinx::acap::aie::lock_unit::locks`

The locking units of the locking device.

Definition at line 96 of file lock.hpp.

Referenced by lock().

9.1.2.12 struct trisycl::vendor::xilinx::acap::aie::memory

template<typename AIE, int X, int Y>
struct trisycl::vendor::xilinx::acap::aie::memory< AIE, X, Y >

The AI Engine Memory Module infrastructure of a tile.

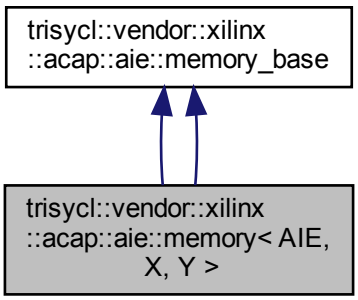
This is the type you need to inherit from to define the content of a CGRA tile memory module.

Parameters

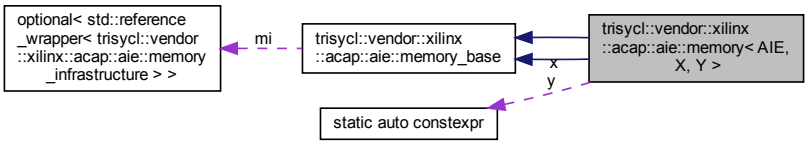
AIE	is the type representing the full CGRA with the programs and memory contents
X	is the horizontal coordinate of the memory module
Y	is the vertical coordinate of the memory module

Definition at line 43 of file memory.hpp.

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::memory< AIE, X, Y >:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::memory< AIE, X, Y >:



Static Public Attributes

- static auto constexpr x = X

  The horizontal tile coordinates in the CGRA grid (starting at 0 and increasing towards East)
- static auto constexpr y = Y

  The vertical tile coordinates in the CGRA grid (starting at increasing towards North)

Additional Inherited Members

9.1.2.12.1 Member Data Documentation

9.1.2.12.1.1 x `template<typename AIE , int X, int Y>`

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::memory< AIE, X, Y >::x = X  [static],
[constexpr]
```

The horizontal tile coordinates in the CGRA grid (starting at 0 and increasing towards East)

Definition at line 46 of file memory.hpp.

9.1.2.12.1.2 y `template<typename AIE , int X, int Y>`

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::memory< AIE, X, Y >::y = Y  [static],
[constexpr]
```

The vertical tile coordinates in the CGRA grid (starting at increasing towards North)

Definition at line 49 of file memory.hpp.

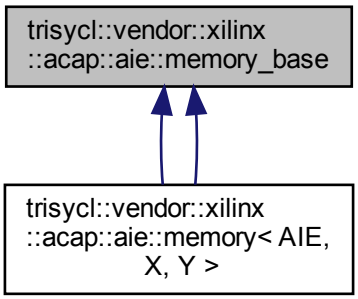
9.1.2.13 struct trisycl::vendor::xilinx::acap::aie::memory_base

The AI Engine Memory Module infrastructure of a tile common to all the tiles.

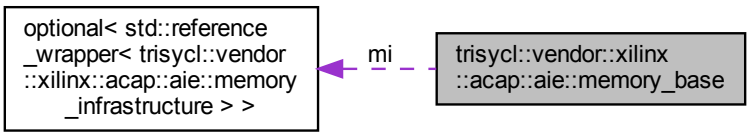
This allows some type erasure while accessing the common memory module infrastructure.

Definition at line 37 of file memory_base.hpp.

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::memory_base:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::memory_base:



Public Member Functions

- auto & lock (int i)

  Get an access to the a specific lock.
- void set_memory_infrastructure (memory_infrastructure &m)

  Store a way to access to hardware infrastructure of the tile.
- auto & lock (int i)

  Get an access to the a specific lock.
- void set_memory_infrastructure (memory_infrastructure &m)

  Store a way to access to hardware infrastructure of the tile.

Public Attributes

- std::optional< std::reference_wrapper< memory_infrastructure > > mi

  Keep a reference to the memory_infrastructure hardware features.

9.1.2.13.1 Member Function Documentation

9.1.2.13.1.1 lock() [1/2] auto & trisycl::vendor::xilinx::acap::aie::memory_base::lock (int i) [inline]

Get an access to the a specific lock.

Definition at line 42 of file memory_base.hpp.

```
00042                     {
00043     // value() will throw if there is some missed initialization
00044     return mi.value().get().lock(i);
00045   }
```

References i, and mi.

9.1.2.13.1.2 lock() [2/2] auto & trisycl::vendor::xilinx::acap::aie::memory_base::lock (int i) [inline]

Get an access to the a specific lock.

Definition at line 42 of file memory_base.hpp.

```
00042                     {
00043     // value() will throw if there is some missed initialization
00044     return mi.value().get().lock(i);
00045   }
```

References i, and mi.

9.1.2.13.1.3 set_memory_infrastructure() [1/2] void trisycl::vendor::xilinx::acap::aie::memory_base::set_memory_infrastructure (memory_infrastructure & m) [inline]

Store a way to access to hardware infrastructure of the tile.

Definition at line 48 of file memory_base.hpp.

```
00048 { mi = m; }
```

References mi.

9.1.2.13.1.4 set_memory_infrastructure() [2/2] void trisycl::vendor::xilinx::acap::aie::memory_base::set_memory_infrastructure (memory_infrastructure & m) [inline]

Store a way to access to hardware infrastructure of the tile.

Definition at line 48 of file memory_base.hpp.

```
00048 { mi = m; }
```

References mi.

9.1.2.13.2 Member Data Documentation

9.1.2.13.2.1 mi `std::optional< std::reference_wrapper< memory_infrastructure > > trisycl::vendor::xilinx::acap::aie::memory_base::mi`

Keep a reference to the memory_infrastructure hardware features.

Definition at line 39 of file memory_base.hpp.

Referenced by lock(), and set_memory_infrastructure().

9.1.2.14 class trisycl::vendor::xilinx::acap::aie::memory_infrastructure

The AI Engine memory infrastructure common to all the tiles.

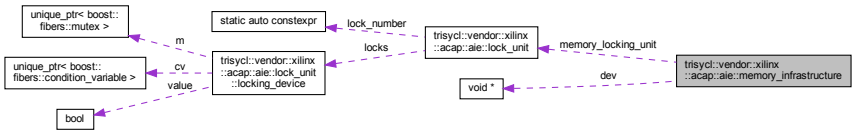
This allows some type erasure while accessing the common memory infrastructure.

Parameters

Geography	is the `geography` type representing the full layout of the architecture

Definition at line 33 of file memory_infrastructure.hpp.

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::memory_infrastructure:



Public Member Functions

- memory_infrastructure (auto &dev)

  Start the memory infrastructure associated to the AIE device tile.
- auto & lock (int i)

  Get access to a specific lock in this memory module.
- memory_infrastructure (auto &dev)

  Start the memory infrastructure associated to the AIE device tile.
- auto & lock (int i)

  Get access to a specific lock in this memory module.

Private Attributes

- lock_unit memory_locking_unit

  The lock unit of the memory tile.
- void * dev

  Keep track of the aie::detail::device for hardware resource control in device mode or for debugging purpose for better messages.

9.1.2.14.1 Constructor & Destructor Documentation

9.1.2.14.1.1 memory_infrastructure() [1/2] trisycl::vendor::xilinx::acap::aie::memory_infrastructure↩ ::memory_infrastructure (auto & dev) [inline]

Start the memory infrastructure associated to the AIE device tile.

Parameters

in	*dev*	is the aie::detail::device used to control hardware when using real hardware and provide some debug information from inside the tile_infrastructure.

Use auto concept here to avoid explicit type causing circular dependency

Definition at line 55 of file memory_infrastructure.hpp.

```
00056       : dev { &dev } {}
```

9.1.2.14.1.2 memory_infrastructure() [2/2] trisycl::vendor::xilinx::acap::aie::memory_infrastructure↩ ::memory_infrastructure (auto & dev) [inline]

Start the memory infrastructure associated to the AIE device tile.

Parameters

in	*dev*	is the aie::detail::device used to control hardware when using real hardware and provide some debug information from inside the tile_infrastructure.

Use auto concept here to avoid explicit type causing circular dependency

Definition at line 55 of file memory_infrastructure.hpp.

```
00056       : dev { &dev } {}
```

9.1.2.14.2 Member Function Documentation

9.1.2.14.2.1 lock() [1/2] auto & trisycl::vendor::xilinx::acap::aie::memory_infrastructure::lock (int i) [inline]

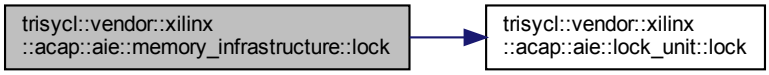
Get access to a specific lock in this memory module.

Definition at line 59 of file memory_infrastructure.hpp.

```
00059 { return memory_locking_unit.lock(i); }
```

References i, trisycl::vendor::xilinx::acap::aie::lock_unit::lock(), and memory_locking_unit.

Here is the call graph for this function:



9.1.2.14.2.2 lock() [2/2] auto & trisycl::vendor::xilinx::acap::aie::memory_infrastructure::lock (int *i*) [inline]

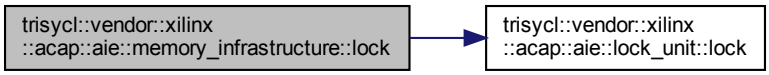
Get access to a specific lock in this memory module.

Definition at line 59 of file memory_infrastructure.hpp.

```
00059 { return memory_locking_unit.lock(i); }
```

References i, trisycl::vendor::xilinx::acap::aie::lock_unit::lock(), and memory_locking_unit.

Here is the call graph for this function:



9.1.2.14.3 Member Data Documentation

9.1.2.14.3.1 dev void * trisycl::vendor::xilinx::acap::aie::memory_infrastructure::dev [private]

Keep track of the aie::detail::device for hardware resource control in device mode or for debugging purpose for better messages.

Use void* for now to avoid cyclic header dependencies for now instead of the aie::detail::device

Definition at line 43 of file memory_infrastructure.hpp.

9.1.2.14.3.2 memory_locking_unit lock_unit trisycl::vendor::xilinx::acap::aie::memory_infrastructure::memory_locking_unit [private]

The lock unit of the memory tile.

Definition at line 35 of file memory_infrastructure.hpp.

Referenced by lock().

9.1.2.15 struct trisycl::vendor::xilinx::acap::aie::program

template<typename AIEDevice, template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>
struct trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >

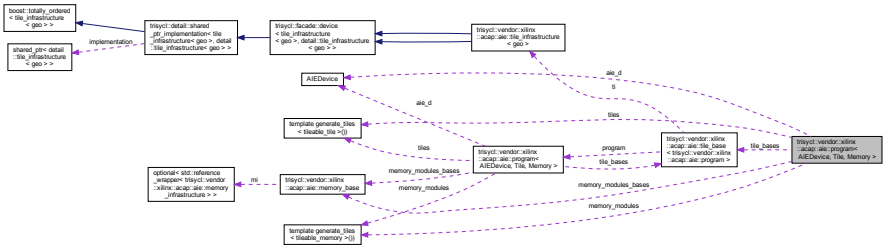
Define an AI Engine CGRA program with its code and memory per core.

Parameters

AIEDevice	is the device description of the machine to instantiate with the physical size
Tile	is the description of the program tiles to instantiate. By default each tile will run an empty program.
Memory	is the description of the machine memory modules. By default the machine has empty memory modules.

Definition at line 54 of file program.hpp.

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >:



Public Types

- using geo = typename AIEDevice::geo

  The geography of the CGRA.
- using device = AIEDevice
- template<int X, int Y>
  using tileable_memory = Memory< program, X, Y >

  Type describing all the memory modules of the CGRA.
- template<int X, int Y>
  using tileable_tile = Tile< program, X, Y >

  Type describing the programs of all the cores in the CGRA.
- using geo = typename AIEDevice::geo

  The geography of the CGRA.
- using device = AIEDevice
- template<int X, int Y>
  using tileable_memory = Memory< program, X, Y >

  Type describing all the memory modules of the CGRA.
- template<int X, int Y>
  using tileable_tile = Tile< program, X, Y >

  Type describing the programs of all the cores in the CGRA.

Public Member Functions

- memory_base & memory_module (int x, int y)

  Access to the common infrastructure part of a memory module.
- template<int LinearId>
  auto & memory_module ()

  Access to a heterogeneous memory module by its linear id.
- template<int X, int Y>
  auto & memory_module ()

  Access to a heterogeneous memory module by its coordinates.
- template<typename F >
  void for_each_memory_base (F &&f)

  Iterate on all the memory module bases of the AIE in an homogeneous way.
- template<int LinearId>
  auto & tile ()

  Access to a heterogeneous tile by linear id.
- template<int X, int Y>
  auto & tile ()

  Access to a heterogeneous tile by its coordinates.
- template<typename F >
  void for_each_tile_base (F &&f)

  Iterate on all the tile bases of the AIE in an homogeneous way.
- program (AIEDevice &aie_d)

  Create the AIE program with the tiles and memory modules.
- template<typename KernelName , typename KernelType >
  void kernel_outliner (KernelType k)

  Instantiate a kernel in a form that can be outlined by the SYCL device compiler.
- void wait ()

  Wait for the end of the execution of each tile.
- void run ()

  Launch the programs of all the tiles of the CGRA in their own executor (CPU thread, fiber...) and wait for their completion.
- template<typename Invocable >
  void run (Invocable &&f)

  Run synchronously an heterogeneous invocable collectively on the device.
- auto & cascade ()

  Access the cascade connections.
- memory_base & memory_module (int x, int y)

  Access to the common infrastructure part of a memory module.
- template<int LinearId>
  auto & memory_module ()

  Access to a heterogeneous memory module by its linear id.
- template<int X, int Y>
  auto & memory_module ()

  Access to a heterogeneous memory module by its coordinates.
- template<typename F >
  void for_each_memory_base (F &&f)

  Iterate on all the memory module bases of the AIE in an homogeneous way.
- template<int LinearId>
  auto & tile ()

  Access to a heterogeneous tile by linear id.
- template<int X, int Y>
  auto & tile ()

Access to a heterogeneous tile by its coordinates.

- template<typename F >
  void for_each_tile_base (F &&f)

  Iterate on all the tile bases of the AIE in an homogeneous way.
- program (AIEDevice &aie_d)

  Create the AIE program with the tiles and memory modules.
- template<typename KernelName , typename KernelType >
  void kernel_outliner (KernelType k)

  Instantiate a kernel in a form that can be outlined by the SYCL device compiler.
- void wait ()

  Wait for the end of the execution of each tile.
- void run ()

  Launch the programs of all the tiles of the CGRA in their own executor (CPU thread, fiber...) and wait for their completion.
- template<typename Invocable >
  void run (Invocable &&f)

  Run synchronously an heterogeneous invocable collectively on the device.
- auto & cascade ()

  Access the cascade connections.

Public Attributes

- AIEDevice aie_d

  The device running this program.
- decltype(geo::template generate_tiles< tileable_memory >()) memory_modules = geo::template generate_tiles<tileable_memory>()

  The tiled memory modules of the CGRA.
- memory_base * memory_modules_bases [geo::y_size][geo::x_size]

  Keep track of all the tiled memory modules as a type-erased memory_modules_base type to have a simpler access to the basic position-independent memory module features.
- decltype(geo::template generate_tiles< tileable_tile >()) tiles

  The tiled programs of the CGRA.
- tile_base< program > * tile_bases [geo::y_size][geo::x_size]

  Keep track of all the tiles as a type-erased tile_base type to have a simpler access to the basic position-independent tile features.

9.1.2.15.1 Member Typedef Documentation

9.1.2.15.1.1 device [1/2]

```
template<typename AIEDevice , template< typename AIE, int X, int Y
> typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
using trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::device = AIEDevice
```

Definition at line 58 of file program.hpp.

9.1.2.15.1.2 device [2/2]

```
template<typename AIEDevice , template< typename AIE, int X, int Y
> typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
using trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::device = AIEDevice
```

Definition at line 58 of file program.hpp.

9.1.2.15.1.3 geo [1/2]

```
template<typename AIEDevice , template< typename AIE, int X, int Y >
typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
using trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::geo = typename
AIEDevice::geo
```

The geography of the CGRA.

Definition at line 57 of file program.hpp.

9.1.2.15.1.4 geo [2/2]

```
template<typename AIEDevice , template< typename AIE, int X, int Y >
typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
using trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::geo = typename
AIEDevice::geo
```

The geography of the CGRA.

Definition at line 57 of file program.hpp.

9.1.2.15.1.5 tileable_memory [1/2]

```
template<typename AIEDevice , template< typename AIE, int
X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename
Memory = acap::aie::memory>
template<int X, int Y>
using trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tileable_memory
= Memory<program, X, Y>
```

Type describing all the memory modules of the CGRA.

Definition at line 65 of file program.hpp.

9.1.2.15.1.6 tileable_memory [2/2]

```
template<typename AIEDevice , template< typename AIE, int
X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename
Memory = acap::aie::memory>
template<int X, int Y>
using trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tileable_memory
= Memory<program, X, Y>
```

Type describing all the memory modules of the CGRA.

Definition at line 65 of file program.hpp.

9.1.2.15.1.7 tileable_tile [1/2]

```
template<typename AIEDevice , template< typename AIE, int X, int
Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
template<int X, int Y>
using trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tileable_tile =
Tile<program, X, Y>
```

Type describing the programs of all the cores in the CGRA.

Definition at line 81 of file program.hpp.

9.1.2.15.1.8 tileable_tile [2/2]

```
template<typename AIEDevice , template< typename AIE, int X, int
Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
template<int X, int Y>
using trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tileable_tile =
Tile<program, X, Y>
```

Type describing the programs of all the cores in the CGRA.

Definition at line 81 of file program.hpp.

9.1.2.15.2 Constructor & Destructor Documentation

9.1.2.15.2.1 program() [1/2]

```
template<typename AIEDevice , template< typename AIE, int X, int
Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::program (
            AIEDevice & aie_d )  [inline]
```

Create the AIE program with the tiles and memory modules.

Definition at line 177 of file program.hpp.

```
00177                           :  aie_d { aie_d } {
00178     boost::hana::for_each(tiles, [&] (auto& t) {
00179         // Inform each tile about its program
00180         t.set_program(*this);
00181         // Inform each tile about their tile infrastructure
00182         t.set_tile_infrastructure(aie_d.tile(t.x, t.y));
00183         // Keep track of each base tile
00184         tile_bases[t.y][t.x] = &t;
00185       });
00186     // Connect each memory to its infrastructure
00187     boost::hana::for_each(memory_modules, [&] (auto& m) {
00188         // Inform each tile about their tile infrastructure
00189         m.set_memory_infrastructure(aie_d.mem(m.x, m.y));
00190         // Keep track of each base tile
00191         memory_modules_bases[m.y][m.x] = &m;
00192       });
00193   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::aie_d, trisycl::vendor::xilinx::acap::aie::program<
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_modules_bases, trisycl::vendor::xilinx::acap::aie::pr
and trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tiles.

9.1.2.15.2.2 program() [2/2] template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>

trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::program (AIEDevice & *aie_d*) [inline]

Create the AIE program with the tiles and memory modules.

Definition at line 177 of file program.hpp.

```
00177                         :  aie_d { aie_d } {
00178     boost::hana::for_each(tiles, [&] (auto& t) {
00179         // Inform each tile about its program
00180         t.set_program(*this);
00181         // Inform each tile about their tile infrastructure
00182         t.set_tile_infrastructure(aie_d.tile(t.x, t.y));
00183         // Keep track of each base tile
00184         tile_bases[t.y][t.x] = &t;
00185       });
00186     // Connect each memory to its infrastructure
00187     boost::hana::for_each(memory_modules, [&] (auto& m) {
00188         // Inform each tile about their tile infrastructure
00189         m.set_memory_infrastructure(aie_d.mem(m.x, m.y));
00190         // Keep track of each base tile
00191         memory_modules_bases[m.y][m.x] = &m;
00192       });
00193   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::aie_d, trisycl::vendor::xilinx::acap::aie::program< trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_modules_bases, trisycl::vendor::xilinx::acap::aie::pro and trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tiles.

9.1.2.15.3 Member Function Documentation

9.1.2.15.3.1 cascade() [1/2] template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>

auto & trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::cascade ( ) [inline]

Access the cascade connections.

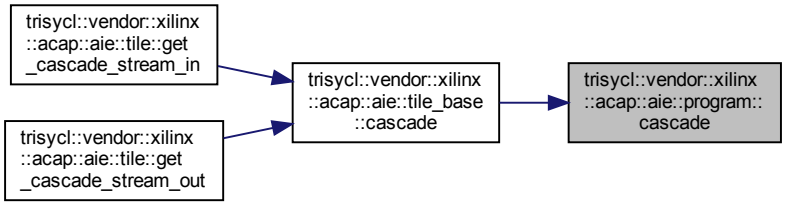
Definition at line 276 of file program.hpp.

```
00276                   {
00277     return aie_d.cascade();
00278   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::aie_d.

Referenced by trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::cascade().

Here is the caller graph for this function:



9.1.2.15.3.2 cascade() [2/2]

```
template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>
auto & trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::cascade ( ) [inline]
```

Access the cascade connections.

Definition at line 276 of file program.hpp.

```
00276                 {
00277     return aie_d.cascade();
00278   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::aie_d.

9.1.2.15.3.3 for_each_memory_base() [1/2]

```
template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>
template<typename F >
void trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::for_each_memory_base (
            F && f )  [inline]
```

Iterate on all the memory module bases of the AIE in an homogeneous way.

Parameters

in	*F*	is the function to apply on each memory module base

Definition at line 135 of file program.hpp.

```
00135                                 {
00136     for (auto y = 0; y != geo::y_size; ++y)
00137       for (auto x = 0; x != geo::x_size; ++x)
00138         f(*memory_modules_bases[y][x]);
00139   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_modules_bases, x, and trisycl::y.

9.1.2.15.3.4 for_each_memory_base() [2/2]

```
template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>
template<typename F >
void trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::for_each_memory_base (
            F && f )  [inline]
```

Iterate on all the memory module bases of the AIE in an homogeneous way.

Parameters

in	*F*	is the function to apply on each memory module base

Definition at line 135 of file program.hpp.

```
00135                                       {
00136     for (auto y = 0; y != geo::y_size; ++y)
00137       for (auto x = 0; x != geo::x_size; ++x)
00138         f(*memory_modules_bases[y][x]);
00139   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_modules_bases, x, and trisycl::y.

9.1.2.15.3.5 for_each_tile_base() [1/2]

```
template<typename AIEDevice , template< typename AIE,
int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename
Memory = acap::aie::memory>
template<typename F >
void trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::for_each_tile_↩
base (
             F && f )  [inline]
```

Iterate on all the tile bases of the AIE in an homogeneous way.

Parameters

in	*F*	is the function to apply on each tile base

Definition at line 169 of file program.hpp.

```
00169                                       {
00170     for (auto y = 0; y != geo::y_size; ++y)
00171       for (auto x = 0; x != geo::x_size; ++x)
00172         f(*tile_bases[y][x]);
00173   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tile_bases, x, and trisycl::y.

9.1.2.15.3.6 for_each_tile_base() [2/2]

```
template<typename AIEDevice , template< typename AIE,
int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename
Memory = acap::aie::memory>
template<typename F >
void trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::for_each_tile_↩
base (
             F && f )  [inline]
```

Iterate on all the tile bases of the AIE in an homogeneous way.

Parameters

in	*F*	is the function to apply on each tile base

Definition at line 169 of file program.hpp.

```
00169                                       {
00170     for (auto y = 0; y != geo::y_size; ++y)
00171       for (auto x = 0; x != geo::x_size; ++x)
00172         f(*tile_bases[y][x]);
00173   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tile_bases, x, and trisycl::y.

9.1.2.15.3.7 kernel_outliner() [1/2]

```
template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>
template<typename KernelName , typename KernelType >
void trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::kernel_outliner (
            KernelType k )  [inline]
```

Instantiate a kernel in a form that can be outlined by the SYCL device compiler.

Parameters

in	*KernelName*	k is the kernel name type required by SYCL
in	*k*	is the kernel functor

Definition at line 207 of file program.hpp.

```
00207                                       {
00208     k();
00209   }
```

9.1.2.15.3.8 kernel_outliner() [2/2]

```
template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>
template<typename KernelName , typename KernelType >
void trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::kernel_outliner (
            KernelType k )  [inline]
```

Instantiate a kernel in a form that can be outlined by the SYCL device compiler.

Parameters

in	*KernelName*	k is the kernel name type required by SYCL
in	*k*	is the kernel functor

Definition at line 207 of file program.hpp.

```
00207                                       {
00208     k();
00209   }
```

9.1.2.15.3.9 memory_module() [1/6]

```
template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>
template<int LinearId>
auto & trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_module (
)  [inline]
```

Access to a heterogeneous memory module by its linear id.

Parameters

in	*Linear↩ Id*	is the linear id

Definition at line 112 of file program.hpp.

```
00112                           {
00113     return boost::hana::at_c<LinearId>(memory_modules);
00114   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_modules.

9.1.2.15.3.10 memory_module() [2/6]

```
template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>
template<int X, int Y>
auto & trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_module ( ) [inline]
```

Access to a heterogeneous memory module by its coordinates.

Parameters

in	*X*	is the horizontal memory module coordinate
in	*Y*	is the vertical memory module coordinate

Definition at line 124 of file program.hpp.

```
00124                           {
00125     return memory_module<geo::linear_id(X, Y)>();
00126   }
```

9.1.2.15.3.11 memory_module() [3/6]

```
template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>
template<int LinearId>
auto & trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_module ( ) [inline]
```

Access to a heterogeneous memory module by its linear id.

Parameters

in	*Linear↩ Id*	is the linear id

Definition at line 112 of file program.hpp.

```
00112                           {
00113     return boost::hana::at_c<LinearId>(memory_modules);
00114   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_modules.

9.1.2.15.3.12 memory_module() [4/6]

```
template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>
template<int X, int Y>
auto & trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_module ( ) [inline]
```

Access to a heterogeneous memory module by its coordinates.

Parameters

in	*X*	is the horizontal memory module coordinate
in	*Y*	is the vertical memory module coordinate

Definition at line 124 of file program.hpp.

```
00124                                {
00125     return memory_module<geo::linear_id(X, Y)>();
00126   }
```

9.1.2.15.3.13 memory_module() [5/6]

```
template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>
memory_base & trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory↩
_module (
              int x,
              int y ) [inline]
```

Access to the common infrastructure part of a memory module.

Parameters

in	*x*	is the horizontal memory module coordinate
in	*y*	is the vertical memory module coordinate

Definition at line 101 of file program.hpp.

```
00101                                                {
00102     geo::validate_x_y(x, y);
00103     return *memory_modules_bases[y][x];
00104   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_modules_bases, x, and trisycl::y.

9.1.2.15.3.14 memory_module() [6/6]

```
template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>
memory_base & trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory↩
_module (
```

```
            int x,
            int y )  [inline]
```

Access to the common infrastructure part of a memory module.

Parameters

in	*x*	is the horizontal memory module coordinate
in	*y*	is the vertical memory module coordinate

Definition at line 101 of file program.hpp.

```
00101                                                {
00102     geo::validate_x_y(x, y);
00103     return *memory_modules_bases[y][x];
00104   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_modules_bases, x, and trisycl::y.

9.1.2.15.3.15 run() [1/4]

```
template<typename AIEDevice , template< typename AIE, int X, int Y
> typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
void trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::run ( )  [inline]
```

Launch the programs of all the tiles of the CGRA in their own executor (CPU thread, fiber...) and wait for their completion.

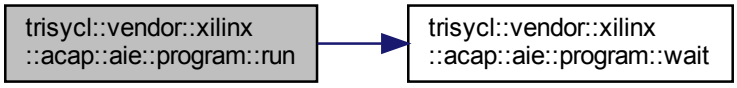
This is the main member function to use to launch the execution.

Definition at line 225 of file program.hpp.

```
00225             {
00226     // Start each tile program in its own executor
00227     boost::hana::for_each(tiles, [&] (auto& t) {
00228         t.single_task([&] {
00229             TRISYCL_DUMP_T("Starting AIE tile (" « t.x « ',' « t.y
00230                            « ") linear id = " « t.linear_id());
00231             /* Just use a capture by reference in the following
00232 because there is direct execution here */
00233             auto kernel = [&] {
00234                             // If the tile has an operator(), use it
00235                             if constexpr (requires { t(); })
00236                               return [&] { t(); };
00237                             /* Else the kernel should have a run
00238 member function and use it.  */
00239                             else
00240                               return [&] { t.run(); };
00241             }();
00242             using kernel_type = decltype(kernel);
00243             // Use the kernel type as its SYCL name too
00244             kernel_outliner<kernel_type, kernel_type>(kernel);
00245             TRISYCL_DUMP_T("Stopping AIE tile (" « t.x « ',' « t.y « ')');
00246           });
00247       });
00248     wait();
00249   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tiles, TRISYCL_DUMP_T, and trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::wait().

Here is the call graph for this function:



9.1.2.15.3.16 run() [2/4]

```
template<typename AIEDevice , template< typename AIE, int X, int Y
> typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
void trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::run ( )  [inline]
```

Launch the programs of all the tiles of the CGRA in their own executor (CPU thread, fiber...) and wait for their completion.

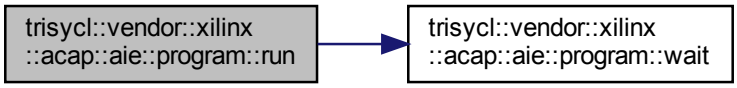
This is the main member function to use to launch the execution.

Definition at line 225 of file program.hpp.

```
00225             {
00226     // Start each tile program in its own executor
00227     boost::hana::for_each(tiles, [&] (auto& t) {
00228         t.single_task([&] {
00229             TRISYCL_DUMP_T("Starting AIE tile (" « t.x « ',' « t.y
00230                            « ") linear id = " « t.linear_id());
00231             /* Just use a capture by reference in the following
00232 because there is direct execution here */
00233             auto kernel = [&] {
00234                               // If the tile has an operator(), use it
00235                               if constexpr (requires { t(); })
00236                                 return [&] { t(); };
00237                               /* Else the kernel should have a run
00238 member function and use it.  */
00239                               else
00240                                 return [&] { t.run(); };
00241             }();
00242             using kernel_type = decltype(kernel);
00243             // Use the kernel type as its SYCL name too
00244             kernel_outliner<kernel_type, kernel_type>(kernel);
00245             TRISYCL_DUMP_T("Stopping AIE tile (" « t.x « ',' « t.y « ')');
00246           });
00247       });
00248     wait();
00249   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tiles, TRISYCL_DUMP_T, and trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::wait().

Here is the call graph for this function:



9.1.2.15.3.17 run() [3/4]

```
template<typename AIEDevice , template< typename AIE, int X, int Y
> typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::run (
             Invocable && f )  [inline]
```

Run synchronously an heterogeneous invocable collectively on the device.

Parameters

f	is an invocable taking an heterogeneous tile handler

Todo Factorize out the 2 run functions

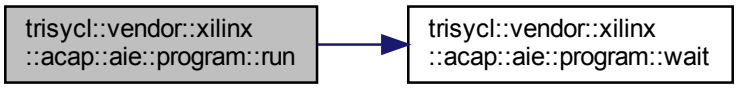
Each tile gets its own copy of work

Definition at line 258 of file program.hpp.

```
00258                          {
00259     // Start each tile program in its own executor
00260     boost::hana::for_each(tiles, [&] (auto& t) {
00261         t.single_task([&] {
00262             TRISYCL_DUMP_T("Starting AIE tile (" « t.x « ',' « t.y
00263                            « ") linear id = " « t.linear_id());
00264 /// Each tile gets its own copy of work
00265             auto kernel = [&, work = f] { work(t); };
00266             using kernel_type = decltype(kernel);
00267             // Use the kernel type as its SYCL name too
00268             kernel_outliner<kernel_type, kernel_type>(kernel);
00269             TRISYCL_DUMP_T("Stopping AIE tile (" « t.x « ',' « t.y « ')');
00270           });
00271       });
00272     wait();
00273   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tiles, TRISYCL_DUMP_T, and trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::wait().

Here is the call graph for this function:



9.1.2.15.3.18 run() [4/4]

```
template<typename AIEDevice , template< typename AIE, int X, int Y
> typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::run (
             Invocable && f )  [inline]
```

Run synchronously an heterogeneous invocable collectively on the device.

Parameters

f	is an invocable taking an heterogeneous tile handler

Todo Factorize out the 2 run functions

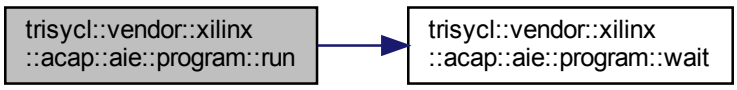
Each tile gets its own copy of work

Definition at line 258 of file program.hpp.

```
00258                           {
00259     // Start each tile program in its own executor
00260     boost::hana::for_each(tiles, [&] (auto& t) {
00261         t.single_task([&] {
00262             TRISYCL_DUMP_T("Starting AIE tile (" « t.x « ',' « t.y
00263                            « ") linear id = " « t.linear_id());
00264 /// Each tile gets its own copy of work
00265             auto kernel = [&, work = f] { work(t); };
00266             using kernel_type = decltype(kernel);
00267             // Use the kernel type as its SYCL name too
00268             kernel_outliner<kernel_type, kernel_type>(kernel);
00269             TRISYCL_DUMP_T("Stopping AIE tile (" « t.x « ',' « t.y « ')');
00270           });
00271       });
00272     wait();
00273   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tiles, TRISYCL_DUMP_T, and trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::wait().

Here is the call graph for this function:



9.1.2.15.3.19 tile() [1/4]

```
template<typename AIEDevice , template< typename AIE, int X, int Y
> typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
template<int LinearId>
auto & trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tile ( )  [inline]
```

Access to a heterogeneous tile by linear id.

Parameters

in	*Linear↩ Id*	is the linear id

Definition at line 147 of file program.hpp.

```
00147                 {
00148     return boost::hana::at_c<LinearId>(tiles);
00149   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tiles.

9.1.2.15.3.20 tile() [2/4] template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory> template<int X, int Y> auto & trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tile ( ) [inline]

Access to a heterogeneous tile by its coordinates.

Parameters

in	*X*	is the horizontal tile coordinate
in	*Y*	is the vertical tile coordinate

Definition at line 159 of file program.hpp.

```
00159                 {
00160     return tile<geo::linear_id(X, Y)>();
00161   }
```

9.1.2.15.3.21 tile() [3/4] template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory> template<int LinearId> auto & trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tile ( ) [inline]

Access to a heterogeneous tile by linear id.

Parameters

in	*Linear↩ Id*	is the linear id

Definition at line 147 of file program.hpp.

```
00147                 {
00148     return boost::hana::at_c<LinearId>(tiles);
00149   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tiles.

9.1.2.15.3.22 tile() [4/4] template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>

```
template<int X, int Y>
auto & trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tile ( )  [inline]
```

Access to a heterogeneous tile by its coordinates.

Parameters

in	*X*	is the horizontal tile coordinate
in	*Y*	is the vertical tile coordinate

Definition at line 159 of file program.hpp.

```
00159                 {
00160     return tile<geo::linear_id(X, Y)>();
00161   }
```

9.1.2.15.3.23 wait() [1/2]

`template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>`
`void trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::wait ( ) [inline]`

Wait for the end of the execution of each tile.

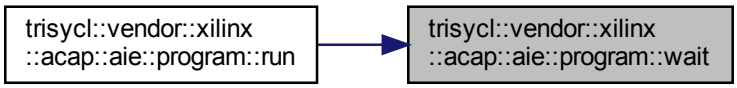
Definition at line 212 of file program.hpp.

```
00212               {
00213     boost::hana::for_each(tiles, [&] (auto& t) {
00214         TRISYCL_DUMP_T("Joining AIE tile (" « t.x « ',' « t.y « ")...");
00215         t.wait();
00216         TRISYCL_DUMP_T("Joined AIE tile (" « t.x « ',' « t.y « ')');
00217       });
00218   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tiles, and TRISYCL_DUMP_T.

Referenced by trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::run().

Here is the caller graph for this function:



9.1.2.15.3.24 wait() [2/2]

`template<typename AIEDevice , template< typename AIE, int X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie::memory>`
`void trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::wait ( ) [inline]`

Wait for the end of the execution of each tile.

Definition at line 212 of file program.hpp.

```
00212               {
00213     boost::hana::for_each(tiles, [&] (auto& t) {
00214         TRISYCL_DUMP_T("Joining AIE tile (" « t.x « ',' « t.y « ")...");
00215         t.wait();
00216         TRISYCL_DUMP_T("Joined AIE tile (" « t.x « ',' « t.y « ')');
00217       });
00218   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tiles, and TRISYCL_DUMP_T.

9.1.2.15.4 Member Data Documentation

9.1.2.15.4.1 aie_d

```
template<typename AIEDevice , template< typename AIE, int X, int Y > typename
Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie↩
::memory>
AIEDevice trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::aie_d
```

The device running this program.

Definition at line 61 of file program.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::cascade(), and trisycl::vendor::xilinx::acap::aie

9.1.2.15.4.2 memory_modules

```
template<typename AIEDevice , template< typename AIE, int X, int
Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
decltype(geo::template generate_tiles< tileable_memory >()) trisycl::vendor::xilinx::acap::aie::program<
AIEDevice, Tile, Memory >::memory_modules = geo::template generate_tiles<tileable_memory>()
```

The tiled memory modules of the CGRA.

Unfortunately it is not possible to use `auto` here... Otherwise it could be just:

```
static inline auto
```

Definition at line 72 of file program.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_module(), and trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::program().

9.1.2.15.4.3 memory_modules_bases

```
template<typename AIEDevice , template< typename AIE, int
X, int Y > typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename
Memory = acap::aie::memory>
memory_base * trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory↩
_modules_bases
```

Keep track of all the tiled memory modules as a type-erased memory_modules_base type to have a simpler access to the basic position-independent memory module features.

Definition at line 77 of file program.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::for_each_memory_base(), trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::memory_module(), and trisycl::vendor::xilinx::acap::aie::progr

9.1.2.15.4.4 tile_bases

```
template<typename AIEDevice , template< typename AIE, int X, int Y >
typename Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory =
acap::aie::memory>
tile_base< program > * trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory
>::tile_bases
```

Keep track of all the tiles as a type-erased tile_base type to have a simpler access to the basic position-independent tile features.

Definition at line 93 of file program.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::for_each_tile_base(), and trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::program().

9.1.2.15.4.5 tiles

```
template<typename AIEDevice , template< typename AIE, int X, int Y > typename
Tile = acap::aie::tile, template< typename AIE, int X, int Y > typename Memory = acap::aie↩
::memory>
decltype(geo::template generate_tiles< tileable_tile >()) trisycl::vendor::xilinx::acap::aie::program<
AIEDevice, Tile, Memory >::tiles
```

Initial value:

```
=
    geo::template generate_tiles<tileable_tile>()
```

The tiled programs of the CGRA.

Unfortunately it is not possible to use `auto` here... Otherwise it could be just:

```
static inline auto
```

Definition at line 87 of file program.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::program(), trisycl::vendor::xilinx::acap::aie::pro
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::tile(), and trisycl::vendor::xilinx::acap::aie::program< AIEDevi

9.1.2.16 struct trisycl::vendor::xilinx::acap::aie::queue

template<typename AIEDevice>
struct trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >

The AI Engine minimal queue.

Todo Make a real queue implementation instead of this mock-up

Parameters

AIEDevice	is the AIE device targeted by the queue

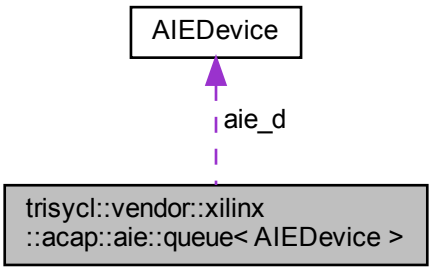
Todo Make a real queue implementation instead of this mock-up

Parameters

AIEDevice	is the AIE device targeted by the queue

Definition at line 30 of file queue.hpp.

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >:



Public Types

- using device = AIEDevice
- using geo = typename device::geo
- using layout = typename geo::layout
- using device = AIEDevice
- using geo = typename device::geo
- using layout = typename geo::layout

Public Member Functions

- queue (device &d)
- template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template< typename Device, int X, int Y > typename Memory = acap::aie::memory>
  void run () const

  Run synchronously a program execution on this queue.
- template<typename Invocable >
  void run (Invocable &&f) const

  Run synchronously an heterogeneous invocable on this queue.
- void wait () const

  Wait for all the device tiles on this queue to finish.
- template<typename Invocable >
  void uniform_run (Invocable &&f) const

  Run synchronously a uniform invocable on this queue.
- template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template< typename Device, int X, int Y > typename Memory = acap::aie::memory>
  auto submit () const

  Submit a program execution on this queue.

- queue (device &d)
- template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template< typename Device, int X, int Y > typename Memory = acap::aie::memory>
  void run () const

  Run synchronously a program execution on this queue.
- template<typename Invocable >
  void run (Invocable &&f) const

  Run synchronously an heterogeneous invocable on this queue.
- void wait () const

  Wait for all the device tiles on this queue to finish.
- template<typename Invocable >
  void uniform_run (Invocable &&f) const

  Run synchronously a uniform invocable on this queue.
- template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template< typename Device, int X, int Y > typename Memory = acap::aie::memory>
  auto submit () const

  Submit a program execution on this queue.

Public Attributes

- device & aie_d

9.1.2.16.1 Member Typedef Documentation

9.1.2.16.1.1 device [1/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::device = AIEDevice
```

Definition at line 31 of file queue.hpp.

9.1.2.16.1.2 device [2/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::device = AIEDevice
```

Definition at line 31 of file queue.hpp.

9.1.2.16.1.3 geo [1/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::geo = typename device::geo
```

Definition at line 32 of file queue.hpp.

9.1.2.16.1.4 geo [2/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::geo = typename device::geo
```

Definition at line 32 of file queue.hpp.

9.1.2.16.1.5 layout [1/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::layout = typename geo::layout
```

Definition at line 33 of file queue.hpp.

9.1.2.16.1.6 layout [2/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::layout = typename geo::layout
```

Definition at line 33 of file queue.hpp.

9.1.2.16.2 Constructor & Destructor Documentation

9.1.2.16.2.1 queue() [1/2] template<typename AIEDevice >

```
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue (
            device & d )  [inline]
```

Definition at line 38 of file queue.hpp.

```
00039       :  aie_d { d } {}
```

9.1.2.16.2.2 queue() [2/2] template<typename AIEDevice >

```
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue (
            device & d )  [inline]
```

Definition at line 38 of file queue.hpp.

```
00039       :  aie_d { d } {}
```

9.1.2.16.3 Member Function Documentation

9.1.2.16.3.1 run() [1/4] template<typename AIEDevice >

```
template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template<
typename Device, int X, int Y > typename Memory = acap::aie::memory>
void trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::run ( ) const  [inline]
```

Run synchronously a program execution on this queue.

Parameters

Tile	is the description of the program tiles to instantiate. By default each tile will run an empty program.
Memory	is the description of the machine memory modules. By default the machine has empty memory modules.

Definition at line 63 of file queue.hpp.

```
00063                   {
00064     program<device, Tile, Memory> { aie_d }.run();
00065   }
```

References trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::aie_d.

9.1.2.16.3.2 run() [2/4] template<typename AIEDevice >

```
template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template<
typename Device, int X, int Y > typename Memory = acap::aie::memory>
void trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::run ( ) const  [inline]
```

Run synchronously a program execution on this queue.

Parameters

Tile	is the description of the program tiles to instantiate. By default each tile will run an empty program.
Memory	is the description of the machine memory modules. By default the machine has empty memory modules.

Definition at line 63 of file queue.hpp.

```
00063                     {
00064     program<device, Tile, Memory> { aie_d }.run();
00065   }
```

References trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::aie_d.

9.1.2.16.3.3 run() [3/4] template<typename AIEDevice >

```
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::run (
             Invocable && f ) const  [inline]
```

Run synchronously an heterogeneous invocable on this queue.

Parameters

f	is an invocable taking an heterogeneous tile handler

Definition at line 71 of file queue.hpp.

```
00071                                                          {
00072     program<device> { aie_d }.run(std::forward<Invocable>(f));
00073   }
```

References trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::aie_d.

9.1.2.16.3.4 run() [4/4] template<typename AIEDevice >

```
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::run (
             Invocable && f ) const  [inline]
```

Run synchronously an heterogeneous invocable on this queue.

Parameters

f	is an invocable taking an heterogeneous tile handler

Definition at line 71 of file queue.hpp.

```
00071                                                                 {
00072     program<device> { aie_d }.run(std::forward<Invocable>(f));
00073   }
```

References trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::aie_d.

9.1.2.16.3.5 submit() [1/2] template<typename AIEDevice >

```
template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template<
typename Device, int X, int Y > typename Memory = acap::aie::memory>
auto trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::submit ( ) const  [inline]
```

Submit a program execution on this queue.

Todo SUBMIT

Parameters

Tile	is the description of the program tiles to instantiate. By default each tile will run an empty program.
Memory	is the description of the machine memory modules. By default the machine has empty memory modules.

\fixme implement a real queue with lifetime support because here it assume the device will last...

Definition at line 106 of file queue.hpp.

```
00106                     {
00107     return queue_waiter { std::async(std::launch::async,
00108 /** \fixme implement a real queue with
00109 lifetime support because here it assume
00110 the device will last...  */
00111                                      [*this] { this->run<Tile, Memory>(); }) };
00112   }
```

9.1.2.16.3.6 submit() [2/2] template<typename AIEDevice >

```
template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template<
typename Device, int X, int Y > typename Memory = acap::aie::memory>
auto trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::submit ( ) const  [inline]
```

Submit a program execution on this queue.

Todo SUBMIT

Parameters

Tile	is the description of the program tiles to instantiate. By default each tile will run an empty program.
Memory	is the description of the machine memory modules. By default the machine has empty memory modules.

\fixme implement a real queue with lifetime support because here it assume the device will last...

Definition at line 106 of file queue.hpp.

```
00106                       {
00107     return queue_waiter { std::async(std::launch::async,
00108 /** \fixme implement a real queue with
00109 lifetime support because here it assume
00110 the device will last...  */
00111                                      [*this] { this->run<Tile, Memory>(); }) };
00112   }
```

9.1.2.16.3.7 uniform_run() [1/2] template<typename AIEDevice >

```
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::uniform_run (
             Invocable && f ) const  [inline]
```

Run synchronously a uniform invocable on this queue.

Parameters

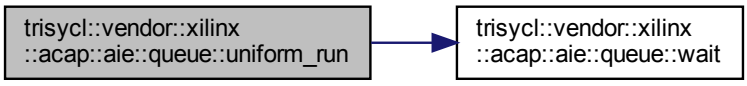
f	is an invocable taking a uniform tile handler

Definition at line 84 of file queue.hpp.

```
00084                                                                    {
00085     aie_d.for_each_tile(
00086         [work = std::forward<Invocable>(f)](auto& t) {
00087           t.single_task(work);
00088         });
00089     wait();
00090   }
```

References trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::aie_d, and trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >:

Here is the call graph for this function:



9.1.2.16.3.8 uniform_run() [2/2] template<typename AIEDevice >

```
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::uniform_run (
             Invocable && f ) const  [inline]
```

Run synchronously a uniform invocable on this queue.

Parameters

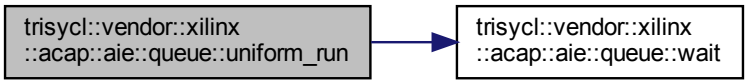
f	is an invocable taking a uniform tile handler

Definition at line 84 of file queue.hpp.

```
00084                                                                    {
00085     aie_d.for_each_tile(
00086         [work = std::forward<Invocable>(f)](auto& t) {
00087           t.single_task(work);
00088         });
00089     wait();
00090   }
```

References trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::aie_d, and trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::

Here is the call graph for this function:



9.1.2.16.3.9 wait() [1/2] template<typename AIEDevice >

```
void trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::wait ( ) const  [inline]
```

Wait for all the device tiles on this queue to finish.

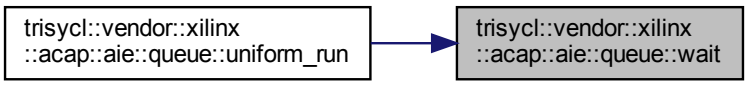
Definition at line 76 of file queue.hpp.

```
00076                     {
00077     aie_d.for_each_tile([](auto& t) { t.wait(); });
00078   }
```

References trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::aie_d.

Referenced by trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::uniform_run().

Here is the caller graph for this function:



9.1.2.16.3.10 wait() [2/2] template<typename AIEDevice >

void trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::wait ( ) const [inline]

Wait for all the device tiles on this queue to finish.

Definition at line 76 of file queue.hpp.

```
00076                   {
00077     aie_d.for_each_tile([](auto& t) { t.wait(); });
00078   }
```

References trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::aie_d.

9.1.2.16.4 Member Data Documentation

9.1.2.16.4.1 aie_d template<typename AIEDevice >

device& trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::aie_d

Todo Why using a reference since now a device is a shared_ptr?

Todo Why using a reference since now a device is a shared_ptr?

Definition at line 36 of file queue.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::run(), trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::u and trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::wait().

9.1.2.17 class trisycl::vendor::xilinx::acap::aie::shim_tile

template<typename AIEDevice>
class trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >

The AI Engine shim tile infrastructure.

Parameters

AIE	is the type representing the full CGRA with the programs and memory contents

Todo Assume only AIE Shim PL tiles for now

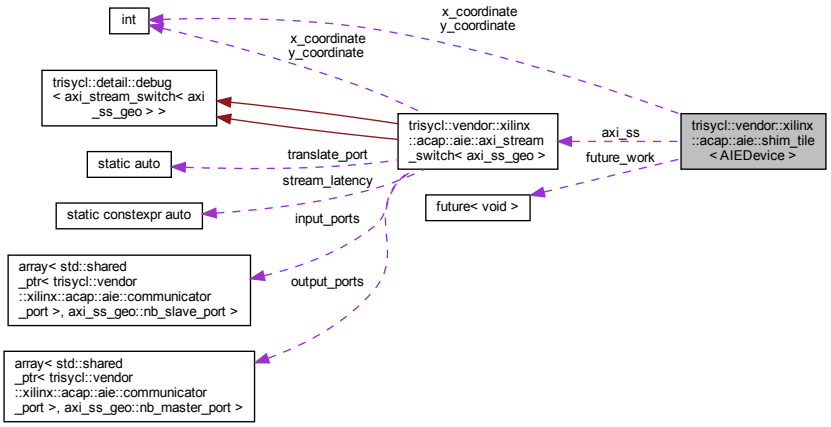
Parameters

AIE	is the type representing the full CGRA with the programs and memory contents

Todo Assume only AIE Shim PL tiles for now

Definition at line 32 of file shim_tile.hpp.

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >:



Public Member Functions

- shim_tile ()

  Construct the shim tile infrastructure.
- void start (int x, ::trisycl::detail::fiber_pool &fiber_executor)

  Start the shim tile infrastructure associated to the AIE device.
- auto & input (spl p)

  Get the input router port of the AXI stream switch.
- auto & output (mpl p)

  Get the output router port of the AXI stream switch.
- auto & bli_in_connection (int port)

  Get the BLI input connection from the shim tile.
- auto & bli_out_connection (int port)

  Get the BLI output connection to the shim tile.
- auto & bli_in (int port)

  Get the BLI (Boundary Logic Interface) input port from the PL (Programmable Logic in the FPGA) to read data from.
- auto & bli_out (int port)

  Get the BLI (Boundary Logic Interface) output port to the PL (Programmable Logic in the FPGA) to write data to.
- void connect (typename geo::shim_axi_stream_switch::slave_port_layout sp, typename geo::shim_axi_↩ stream_switch::master_port_layout mp)

  Configure a connection of the shim tile AXI stream switch.
- shim_tile ()

  Construct the shim tile infrastructure.
- void start (int x, ::trisycl::detail::fiber_pool &fiber_executor)

  Start the shim tile infrastructure associated to the AIE device.
- auto & input (spl p)

  Get the input router port of the AXI stream switch.
- auto & output (mpl p)

  Get the output router port of the AXI stream switch.
- auto & bli_in_connection (int port)

  Get the BLI input connection from the shim tile.
- auto & bli_out_connection (int port)

Get the BLI output connection to the shim tile.

- auto & bli_in (int port)

  Get the BLI (Boundary Logic Interface) input port from the PL (Programmable Logic in the FPGA) to read data from.
- auto & bli_out (int port)

  Get the BLI (Boundary Logic Interface) output port to the PL (Programmable Logic in the FPGA) to write data to.
- void connect (typename geo::shim_axi_stream_switch::slave_port_layout sp, typename geo::shim_axi_↩ stream_switch::master_port_layout mp)

  Configure a connection of the shim tile AXI stream switch.

Static Public Member Functions

- static auto translate_input_port (int port)

  Map the input BLI id/port to the shim AXI stream switch port.
- static auto translate_output_port (int port)

  Map the input BLI id/port to the shim AXI stream switch port.
- static auto translate_input_port (int port)

  Map the input BLI id/port to the shim AXI stream switch port.
- static auto translate_output_port (int port)

  Map the input BLI id/port to the shim AXI stream switch port.

Private Types

- using geo = typename AIEDevice::geo
- using axi_ss_geo = typename geo::shim_axi_stream_switch
- using mpl = typename axi_ss_geo::master_port_layout
- using spl = typename axi_ss_geo::slave_port_layout
- using axi_ss_t = axi_stream_switch< axi_ss_geo >
- using geo = typename AIEDevice::geo
- using axi_ss_geo = typename geo::shim_axi_stream_switch
- using mpl = typename axi_ss_geo::master_port_layout
- using spl = typename axi_ss_geo::slave_port_layout
- using axi_ss_t = axi_stream_switch< axi_ss_geo >

Private Attributes

- int x_coordinate

  Keep the horizontal coordinate for debug purpose.
- int y_coordinate = -1
- axi_ss_t axi_ss

  The AXI stream switch of the core tile.
- std::future< void > future_work

  Keep track of the std::thread execution in this tile.

9.1.2.17.1 Member Typedef Documentation

9.1.2.17.1.1 axi_ss_geo [1/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss_geo = typename geo↩
::shim_axi_stream_switch  [private]
```

Definition at line 34 of file shim_tile.hpp.

9.1.2.17.1.2 axi_ss_geo [2/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss_geo = typename geo↩
::shim_axi_stream_switch  [private]
```

Definition at line 34 of file shim_tile.hpp.

9.1.2.17.1.3 axi_ss_t [1/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss_t = axi_stream_switch<axi_ss_geo>
[private]
```

Definition at line 37 of file shim_tile.hpp.

9.1.2.17.1.4 axi_ss_t [2/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss_t = axi_stream_switch<axi_ss_geo>
[private]
```

Definition at line 37 of file shim_tile.hpp.

9.1.2.17.1.5 geo [1/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::geo = typename AIEDevice↩
::geo  [private]
```

Definition at line 33 of file shim_tile.hpp.

9.1.2.17.1.6 geo [2/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::geo = typename AIEDevice↩
::geo  [private]
```

Definition at line 33 of file shim_tile.hpp.

9.1.2.17.1.7 mpl [1/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::mpl = typename axi_ss_geo↩
::master_port_layout  [private]
```

Definition at line 35 of file shim_tile.hpp.

9.1.2.17.1.8 mpl [2/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::mpl = typename axi_ss_geo↩
::master_port_layout  [private]
```

Definition at line 35 of file shim_tile.hpp.

9.1.2.17.1.9 spl [1/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::spl = typename axi_ss_geo↩
::slave_port_layout  [private]
```

Definition at line 36 of file shim_tile.hpp.

9.1.2.17.1.10 spl [2/2] template<typename AIEDevice >

```
using trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::spl = typename axi_ss_geo↩
::slave_port_layout  [private]
```

Definition at line 36 of file shim_tile.hpp.

9.1.2.17.2 Constructor & Destructor Documentation

9.1.2.17.2.1 shim_tile() [1/2] template<typename AIEDevice >

```
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::shim_tile ( )  [inline]
```

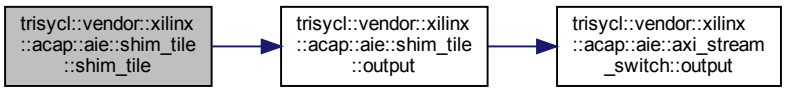
Construct the shim tile infrastructure.

Definition at line 63 of file shim_tile.hpp.

```
00063             {
00064     // Connect the core receivers to its AXI stream switch
00065     for (auto p :  axi_ss_geo::m_south_range)
00066       output(p) = std::make_shared<port_receiver<axi_ss_t»
00067         (axi_ss, "BLI_shim_receiver");
00068   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice > and p.

Here is the call graph for this function:



9.1.2.17.2.2 shim_tile() [2/2] template<typename AIEDevice >

trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::shim_tile ( ) [inline]

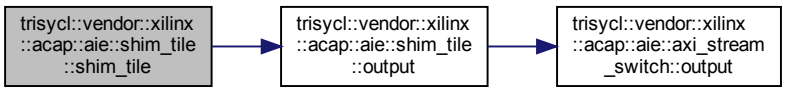
Construct the shim tile infrastructure.

Definition at line 63 of file shim_tile.hpp.

```
00063             {
00064     // Connect the core receivers to its AXI stream switch
00065     for (auto p :  axi_ss_geo::m_south_range)
00066       output(p) = std::make_shared<port_receiver<axi_ss_t»
00067         (axi_ss, "BLI_shim_receiver");
00068   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice > and p.

Here is the call graph for this function:



9.1.2.17.3 Member Function Documentation

9.1.2.17.3.1 bli_in() [1/2] template<typename AIEDevice >

auto & trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_in (
int *port*) [inline]

Get the BLI (Boundary Logic Interface) input port from the PL (Programmable Logic in the FPGA) to read data from.

Parameters

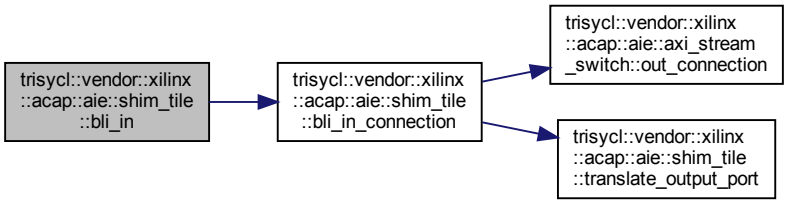
in	*port*	is the BLI id/port to use

Definition at line 155 of file shim_tile.hpp.

```
00155                                {
00156     return *bli_in_connection(port);
00157   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_in_connection().

Here is the call graph for this function:



9.1.2.17.3.2 bli_in() [2/2] template<typename AIEDevice >

```
auto & trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_in (
            int port )  [inline]
```

Get the BLI (Boundary Logic Interface) input port from the PL (Programmable Logic in the FPGA) to read data from.

Parameters

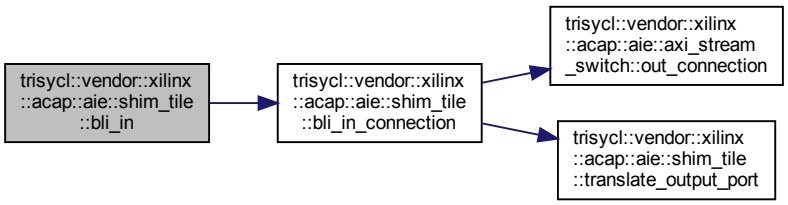
in	*port*	is the BLI id/port to use

Definition at line 155 of file shim_tile.hpp.

```
00155                                {
00156     return *bli_in_connection(port);
00157   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_in_connection().

Here is the call graph for this function:



9.1.2.17.3.3 bli_in_connection() [1/2] template<typename AIEDevice >

```
auto & trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_in_connection (
            int port )  [inline]
```

Get the BLI input connection from the shim tile.

Parameters

in	*port*	is the BLI id/port to use

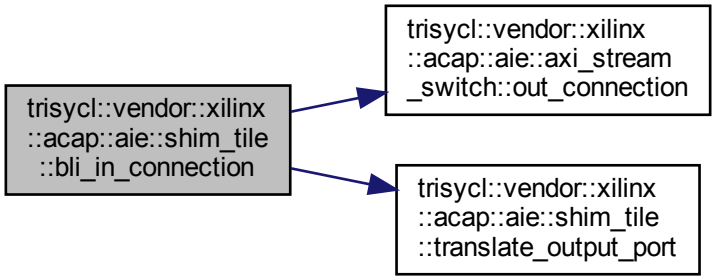
Definition at line 133 of file shim_tile.hpp.

```
00133                                       {
00134     // The input is actually the output of the switch
00135     return axi_ss.out_connection(translate_output_port(port));
00136   }
```

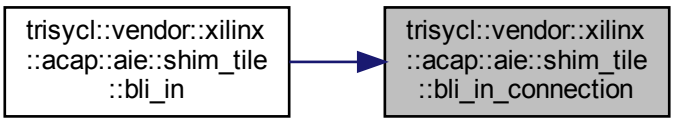
References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AX and trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::translate_output_port().

Referenced by trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_in().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.17.3.4 bli_in_connection() [2/2] template<typename AIEDevice >

```
auto & trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_in_connection (
            int port )  [inline]
```

Get the BLI input connection from the shim tile.

Parameters

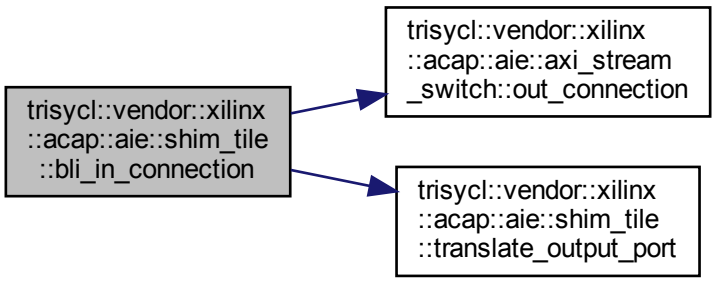
in	*port*	is the BLI id/port to use

Definition at line 133 of file shim_tile.hpp.

```
00133                                       {
00134     // The input is actually the output of the switch
00135     return axi_ss.out_connection(translate_output_port(port));
00136   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AX and trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::translate_output_port().

Here is the call graph for this function:



9.1.2.17.3.5 bli_out() [1/2] template<typename AIEDevice >

```
auto & trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_out (
            int port )  [inline]
```

Get the BLI (Boundary Logic Interface) output port to the PL (Programmable Logic in the FPGA) to write data to.

Parameters

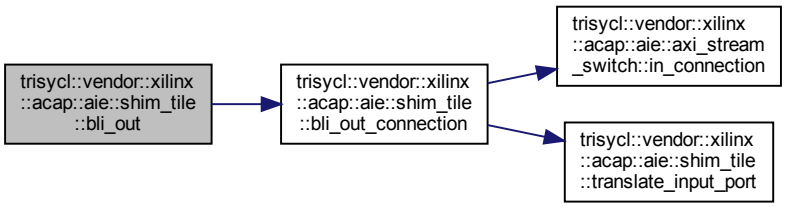
in	*port*	is the BLI id/port to use

Definition at line 165 of file shim_tile.hpp.

```
00165                                       {
00166     return *bli_out_connection(port);
00167   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_out_connection().

Here is the call graph for this function:



9.1.2.17.3.6 bli_out() [2/2] template<typename AIEDevice >

```
auto & trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_out (
            int port )  [inline]
```

Get the BLI (Boundary Logic Interface) output port to the PL (Programmable Logic in the FPGA) to write data to.

Parameters

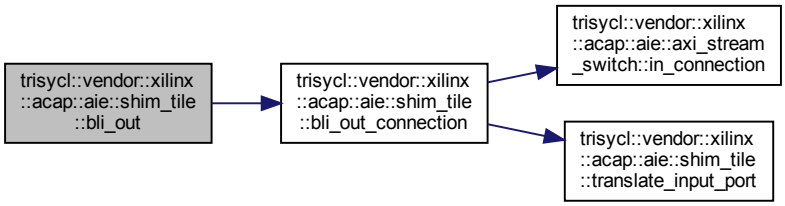
in	*port*	is the BLI id/port to use

Definition at line 165 of file shim_tile.hpp.

```
00165                                {
00166     return *bli_out_connection(port);
00167   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_out_connection().

Here is the call graph for this function:



9.1.2.17.3.7 bli_out_connection() [1/2] template<typename AIEDevice >

```
auto & trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_out_connection (
            int port )  [inline]
```

Get the BLI output connection to the shim tile.

Parameters

in	*port*	is the BLI id/port to use

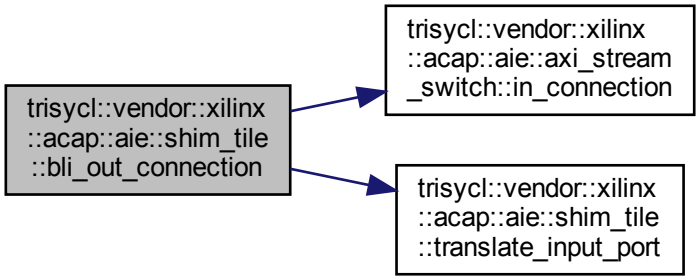
Definition at line 143 of file shim_tile.hpp.

```
00143                                       {
00144     // The output is actually the input of the switch
00145     return axi_ss.in_connection(translate_input_port(port));
00146   }
```

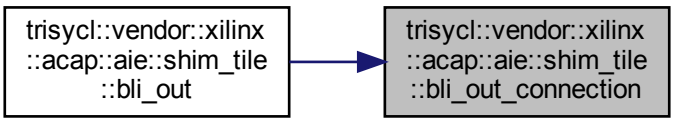
References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AX and trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::translate_input_port().

Referenced by trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_out().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.17.3.8 bli_out_connection() [2/2] template<typename AIEDevice >

```
auto & trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_out_connection (
            int port )  [inline]
```

Get the BLI output connection to the shim tile.

Parameters

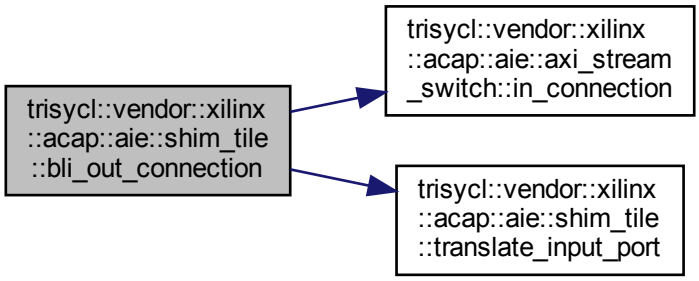
in	*port*	is the BLI id/port to use

Definition at line 143 of file shim_tile.hpp.

```
00143                                       {
00144     // The output is actually the input of the switch
00145     return axi_ss.in_connection(translate_input_port(port));
00146   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AX
and trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::translate_input_port().

Here is the call graph for this function:



9.1.2.17.3.9 connect() [1/2] template<typename AIEDevice >

```
void trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::connect (
            typename geo::shim_axi_stream_switch::slave_port_layout sp,
            typename geo::shim_axi_stream_switch::master_port_layout mp )  [inline]
```

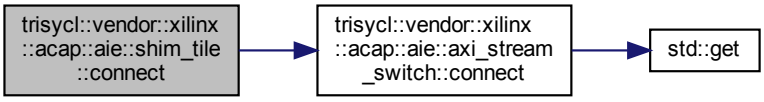
Configure a connection of the shim tile AXI stream switch.

Definition at line 171 of file shim_tile.hpp.

```
00172                                                                   {
00173     axi_ss.connect(sp, mp);
00174   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, and trisycl::vendor::xilinx::acap::aie::axi_stream_switch

Here is the call graph for this function:



9.1.2.17.3.10 connect() [2/2] template<typename AIEDevice >

```
void trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::connect (
            typename geo::shim_axi_stream_switch::slave_port_layout sp,
            typename geo::shim_axi_stream_switch::master_port_layout mp ) [inline]
```

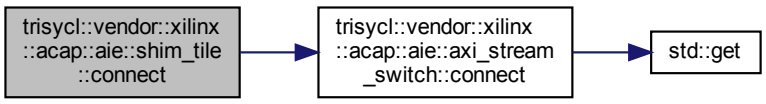
Configure a connection of the shim tile AXI stream switch.

Definition at line 171 of file shim_tile.hpp.

```
00172                                                                    {
00173     axi_ss.connect(sp, mp);
00174   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, and trisycl::vendor::xilinx::acap::aie::axi_stream_switch

Here is the call graph for this function:



9.1.2.17.3.11 input() [1/2] template<typename AIEDevice >

```
auto & trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::input (
            spl p ) [inline]
```

Get the input router port of the AXI stream switch.

Todo Factorize some code between core tile and shim tile

Parameters

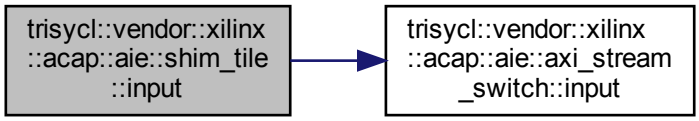
p	is the slave_port_layout for the stream

Definition at line 92 of file shim_tile.hpp.

```
00092                     {
00093     // No index validation required because of type safety
00094     return axi_ss.input(p);
00095   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AX and p.

Here is the call graph for this function:



9.1.2.17.3.12 input() [2/2] template<typename AIEDevice >

```
auto & trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::input (
            spl p )  [inline]
```

Get the input router port of the AXI stream switch.

Todo Factorize some code between core tile and shim tile

Parameters

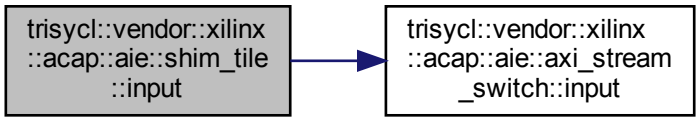
p	is the slave_port_layout for the stream

Definition at line 92 of file shim_tile.hpp.

```
00092                     {
00093     // No index validation required because of type safety
00094     return axi_ss.input(p);
00095   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AX and p.

Here is the call graph for this function:



9.1.2.17.3.13 output() [1/2] template<typename AIEDevice >

```
auto & trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::output (
            mpl p )  [inline]
```

Get the output router port of the AXI stream switch.

Parameters

p	is the master_port_layout for the stream

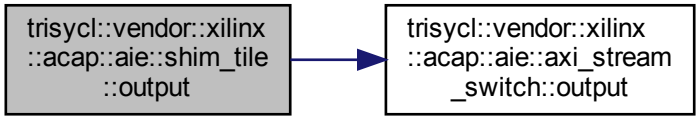
Definition at line 102 of file shim_tile.hpp.

```
00102                       {
00103     // No index validation required because of type safety
00104     return axi_ss.output(p);
00105   }
```

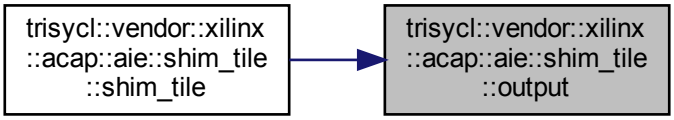
References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AX and p.

Referenced by trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::shim_tile().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.17.3.14 output() [2/2] template<typename AIEDevice >

```
auto & trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::output (
            mpl p )  [inline]
```

Get the output router port of the AXI stream switch.

Parameters

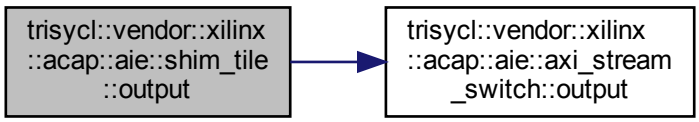
p	is the master_port_layout for the stream

Definition at line 102 of file shim_tile.hpp.

```
00102                                 {
00103     // No index validation required because of type safety
00104     return axi_ss.output(p);
00105   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AX and p.

Here is the call graph for this function:



9.1.2.17.3.15 start() [1/2] template<typename AIEDevice >

```
void trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::start (
             int x,
             ::trisycl::detail::fiber_pool & fiber_executor )  [inline]
```

Start the shim tile infrastructure associated to the AIE device.

Parameters

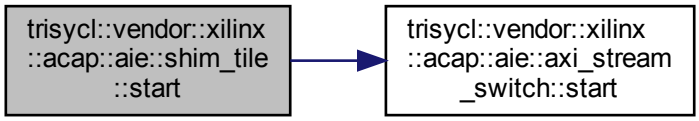
in	*x*	is the horizontal coordinate for this tile
in	*fiber_executor*	is the executor used to run infrastructure details

Definition at line 78 of file shim_tile.hpp.

```
00078                                                                 {
00079     x_coordinate = x;
00080 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00081     fe = &fiber_executor;
00082 #endif
00083     axi_ss.start(x, -1, fiber_executor);
00084   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AX x, and trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::x_coordinate.

Here is the call graph for this function:



9.1.2.17.3.16 start() [2/2] template<typename AIEDevice >

```
void trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::start (
            int x,
            ::trisycl::detail::fiber_pool & fiber_executor )  [inline]
```

Start the shim tile infrastructure associated to the AIE device.

Parameters

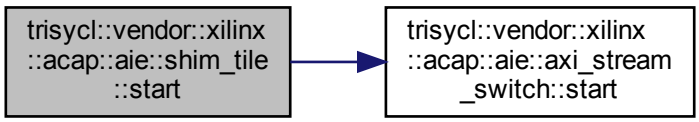
in	*x*	is the horizontal coordinate for this tile
in	*fiber_executor*	is the executor used to run infrastructure details

Definition at line 78 of file shim_tile.hpp.

```
00078                                                                 {
00079     x_coordinate = x;
00080 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00081     fe = &fiber_executor;
00082 #endif
00083     axi_ss.start(x, -1, fiber_executor);
00084   }
```

References trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AX
x, and trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::x_coordinate.

Here is the call graph for this function:



9.1.2.17.3.17 translate_input_port() [1/2] template<typename AIEDevice >

```
static auto trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::translate_input_port (
            int port )  [inline], [static]
```

Map the input BLI id/port to the shim AXI stream switch port.

Parameters

in	*port*	is the BLI id/port to use

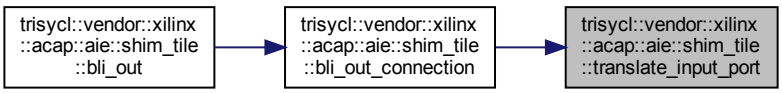
Definition at line 112 of file shim_tile.hpp.

```
00112                                          {
00113     return axi_ss_t::translate_port(port, spl::south_0, spl::south_last,
00114                                     "The BLI input port is out of range");
00115   }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< axi_ss_geo >::translate_port.

Referenced by trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_out_connection().

Here is the caller graph for this function:



9.1.2.17.3.18 translate_input_port() [2/2] template<typename AIEDevice >

```
static auto trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::translate_input_port (
            int port )  [inline], [static]
```

Map the input BLI id/port to the shim AXI stream switch port.

Parameters

in	*port*	is the BLI id/port to use

Definition at line 112 of file shim_tile.hpp.

```
00112                                          {
00113     return axi_ss_t::translate_port(port, spl::south_0, spl::south_last,
00114                                     "The BLI input port is out of range");
00115   }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< axi_ss_geo >::translate_port.

9.1.2.17.3.19 translate_output_port() [1/2] template<typename AIEDevice >

```
static auto trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::translate_output_port
(
            int port )  [inline], [static]
```

Map the input BLI id/port to the shim AXI stream switch port.

Parameters

in	*port*	is the BLI id/port to use

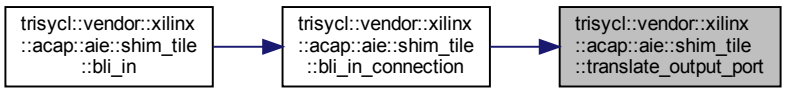
Definition at line 122 of file shim_tile.hpp.

```
00122                                       {
00123     return axi_ss_t::translate_port(port, mpl::south_0, mpl::south_last,
00124                                     "The BLI output port is out of range");
00125   }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< axi_ss_geo >::translate_port.

Referenced by trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_in_connection().

Here is the caller graph for this function:



9.1.2.17.3.20 translate_output_port() [2/2] template<typename AIEDevice >

```
static auto trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::translate_output_port
(
            int port )  [inline], [static]
```

Map the input BLI id/port to the shim AXI stream switch port.

Parameters

in	*port*	is the BLI id/port to use

Definition at line 122 of file shim_tile.hpp.

```
00122                                       {
00123     return axi_ss_t::translate_port(port, mpl::south_0, mpl::south_last,
00124                                     "The BLI output port is out of range");
00125   }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< axi_ss_geo >::translate_port.

9.1.2.17.4 Member Data Documentation

9.1.2.17.4.1 axi_ss template<typename AIEDevice >

axi_ss_t trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::axi_ss [private]

The AXI stream switch of the core tile.

Definition at line 47 of file shim_tile.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::bli_in_connection(), trisycl::vendor::xilinx::acap::aie::shim_til
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::connect(), trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::input()
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::output(), trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::shim_til
and trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::start().

9.1.2.17.4.2 future_work template<typename AIEDevice >

std::future< void > trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::future_work
[private]

Keep track of the std::thread execution in this tile.

Definition at line 57 of file shim_tile.hpp.

9.1.2.17.4.3 x_coordinate template<typename AIEDevice >

int trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::x_coordinate [private]

Keep the horizontal coordinate for debug purpose.

Definition at line 40 of file shim_tile.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::start().

9.1.2.17.4.4 y_coordinate template<typename AIEDevice >

int trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::y_coordinate = -1 [private]

Todo Remove? Keep the vertical coordinate for debug purpose

Todo Remove? Keep the vertical coordinate for debug purpose

Definition at line 44 of file shim_tile.hpp.

9.1.2.18 struct trisycl::vendor::xilinx::acap::aie::tile

template<typename AIE_Program, int X, int Y>
struct trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >

The AI Engine tile infrastructure defining the program of a tile.

This is the type you need to inherit from to define the program of a CGRA tile.

Parameters

AIE_Program	is the type representing the full CGRA with the programs and memory contents
X	is the horizontal coordinate of the memory module
Y	is the vertical coordinate of the memory module

Todo (Andrew Gozillon) FYI, until Intel fix the way they handle base classes in the SYCL compiler frontend this tile_base can never have non-default constructed data (on the device side implementation, the host should be fine to have whatever). As the hack I have in place just now only outlines a call to the base classes default constructor, it might be feasible to work around that though by calling a base class "fill" function in the tile constructor or tweaking the way the run function of a tile is outlined so that it outlines it as a copy of the full object rather than components of the object and reconstructing it inside the kernel (although you'll fall into the standard layout trap as you'll be passing a full object to the kernel invocation and it'd take some reworking of the Tile's main generation).

A proper fix for this is probably quite difficult as you'd need to take into account the base classes dependencies among-st other things like the constructor body not being available in the same TU, some discussion here: https://github.com/intel/llvm/issues/488

This is the type you need to inherit from to define the program of a CGRA tile.

Parameters

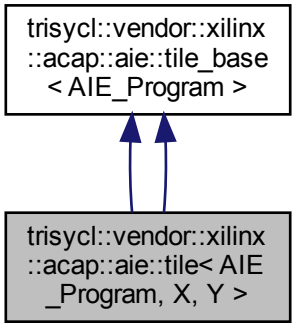
AIE_Program	is the type representing the full CGRA with the programs and memory contents
X	is the horizontal coordinate of the memory module
Y	is the vertical coordinate of the memory module

Todo (Andrew Gozillon) FYI, until Intel fix the way they handle base classes in the SYCL compiler frontend this tile_base can never have non-default constructed data (on the device side implementation, the host should be fine to have whatever). As the hack I have in place just now only outlines a call to the base classes default constructor, it might be feasible to work around that though by calling a base class "fill" function in the tile constructor or tweaking the way the run function of a tile is outlined so that it outlines it as a copy of the full object rather than components of the object and reconstructing it inside the kernel (although you'll fall into the standard layout trap as you'll be passing a full object to the kernel invocation and it'd take some reworking of the Tile's main generation).

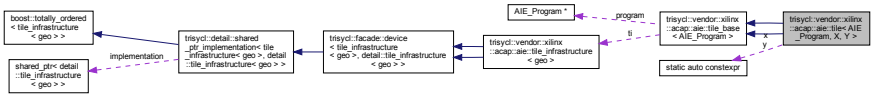
A proper fix for this is probably quite difficult as you'd need to take into account the base classes dependencies among-st other things like the constructor body not being available in the same TU, some discussion here: https://github.com/intel/llvm/issues/488

Definition at line 57 of file tile.hpp.

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >:



Public Types

- using geo = typename AIE_Program::geo

  The geography of the CGRA.
- using tb = tile_base< AIE_Program >

  Shortcut to the tile base class.
- using mem_t = typename AIE_Program::template tileable_memory< x, y >

  The type of the memory module native to the tile.
- using geo = typename AIE_Program::geo

  The geography of the CGRA.
- using tb = tile_base< AIE_Program >

  Shortcut to the tile base class.
- using mem_t = typename AIE_Program::template tileable_memory< x, y >

  The type of the memory module native to the tile.

Public Member Functions

- auto & mem_west ()

  Get the memory module on the West if it does exist.
- auto & mem_east ()

  Get the memory module on the East if it does exist.
- auto & mem_south ()

  Get the memory module on the South if it does exist.

- auto & mem_north ()

  Get the memory module on the North if it does exist.
- auto & mem ()

  The memory module native to the tile.
- template<int Dx, int Dy>
  auto & mem ()

  Get the memory module relative to the tile.
- template<typename T >
  auto get_cascade_stream_in ()

  Get a read accessor to the cascade stream input.
- template<typename T >
  auto get_cascade_stream_out ()

  Get a write accessor to the cascade stream output.
- void horizontal_barrier (int lock=14)

  An horizontal barrier using a lock.
- void vertical_barrier (int lock=15)

  A vertical barrier using a lock.
- void barrier ()

  Full barrier using the 2 locks by default.
- auto rx_dma (int port)

  Get access on a receiver DMA.
- auto tx_dma (int port)

  Get access on a transmit DMA.
- auto & mem_west ()

  Get the memory module on the West if it does exist.
- auto & mem_east ()

  Get the memory module on the East if it does exist.
- auto & mem_south ()

  Get the memory module on the South if it does exist.
- auto & mem_north ()

  Get the memory module on the North if it does exist.
- auto & mem ()

  The memory module native to the tile.
- template<int Dx, int Dy>
  auto & mem ()

  Get the memory module relative to the tile.
- template<typename T >
  auto get_cascade_stream_in ()

  Get a read accessor to the cascade stream input.
- template<typename T >
  auto get_cascade_stream_out ()

  Get a write accessor to the cascade stream output.
- void horizontal_barrier (int lock=14)

  An horizontal barrier using a lock.
- void vertical_barrier (int lock=15)

  A vertical barrier using a lock.
- void barrier ()

  Full barrier using the 2 locks by default.
- auto rx_dma (int port)

  Get access on a receiver DMA.
- auto tx_dma (int port)

  Get access on a transmit DMA.

Static Public Member Functions

- template<int Dim>
  static auto constexpr get_id ()

  Return the coordinate of the tile in the given dimension.
- static auto constexpr linear_id ()

  Return the linearized coordinate of the tile.
- static bool constexpr is_west_column ()

  Test if the tile is in the Western column.
- static bool constexpr is_east_column ()

  Test if the tile is in the Eastern column.
- static bool constexpr is_south_row ()

  Test if the tile is in the Southern row.
- static bool constexpr is_north_row ()

  Test if the tile is in the Northern row.
- static bool constexpr is_west_valid ()

  Test if the tile has a Western neighbor.
- static bool constexpr is_east_valid ()

  Test if the tile has a Eastern neighbor.
- static bool constexpr is_south_valid ()

  Test if the tile has a Southern neighbor.
- static bool constexpr is_south_west_valid ()

  Test if the tile has a South West neighbor.
- static bool constexpr is_south_east_valid ()

  Test if the tile has a South East neighbor.
- static bool constexpr is_north_valid ()

  Test if the tile has a Northern neighbor.
- static bool constexpr is_north_east_valid ()

  Test if the tile has a North East neighbor.
- static bool constexpr is_north_west_valid ()

  Test if the tile has a North West neighbor.
- static bool constexpr is_memory_module (int dx, int dy)

  Test if a memory module exists and is connected to this tile.
- static bool constexpr is_memory_module_west ()

  Test if a memory module exists on the West of this tile.
- static bool constexpr is_memory_module_east ()

  Test if a memory module exists on the East of this tile.
- static bool constexpr is_memory_module_south ()

  Test if a memory module exists on the South this tile.
- static bool constexpr is_memory_module_north ()

  Test if a memory module exists on the North this tile.
- static auto constexpr memory_module_linear_id (int dx, int dy)

  Compute the linear id of a memory module attached to this tile.
- static bool constexpr is_cascade_start ()

  Test if this tile owns the start of the cascade_stream.
- static bool constexpr is_cascade_end ()

  Test if this tile owns the end of the cascade_stream.
- static auto constexpr cascade_linear_id ()

  Compute the linearized coordinate of a tile along the cascade stream.
- template<int Dim>
  static auto constexpr get_id ()

Return the coordinate of the tile in the given dimension.

- static auto constexpr linear_id ()

  Return the linearized coordinate of the tile.
- static bool constexpr is_west_column ()

  Test if the tile is in the Western column.
- static bool constexpr is_east_column ()

  Test if the tile is in the Eastern column.
- static bool constexpr is_south_row ()

  Test if the tile is in the Southern row.
- static bool constexpr is_north_row ()

  Test if the tile is in the Northern row.
- static bool constexpr is_west_valid ()

  Test if the tile has a Western neighbor.
- static bool constexpr is_east_valid ()

  Test if the tile has a Eastern neighbor.
- static bool constexpr is_south_valid ()

  Test if the tile has a Southern neighbor.
- static bool constexpr is_south_west_valid ()

  Test if the tile has a South West neighbor.
- static bool constexpr is_south_east_valid ()

  Test if the tile has a South East neighbor.
- static bool constexpr is_north_valid ()

  Test if the tile has a Northern neighbor.
- static bool constexpr is_north_east_valid ()

  Test if the tile has a North East neighbor.
- static bool constexpr is_north_west_valid ()

  Test if the tile has a North West neighbor.
- static bool constexpr is_memory_module (int dx, int dy)

  Test if a memory module exists and is connected to this tile.
- static bool constexpr is_memory_module_west ()

  Test if a memory module exists on the West of this tile.
- static bool constexpr is_memory_module_east ()

  Test if a memory module exists on the East of this tile.
- static bool constexpr is_memory_module_south ()

  Test if a memory module exists on the South this tile.
- static bool constexpr is_memory_module_north ()

  Test if a memory module exists on the North this tile.
- static auto constexpr memory_module_linear_id (int dx, int dy)

  Compute the linear id of a memory module attached to this tile.
- static bool constexpr is_cascade_start ()

  Test if this tile owns the start of the cascade_stream.
- static bool constexpr is_cascade_end ()

  Test if this tile owns the end of the cascade_stream.
- static auto constexpr cascade_linear_id ()

  Compute the linearized coordinate of a tile along the cascade stream.

Static Public Attributes

- static auto constexpr x = X

  The horizontal tile coordinates in the CGRA grid (starting at 0 and increasing towards the East)
- static auto constexpr y = Y

  The vertical tile coordinates in the CGRA grid (starting at increasing towards the North)

Additional Inherited Members

9.1.2.18.1 Member Typedef Documentation

9.1.2.18.1.1 geo [1/2]

```
template<typename AIE_Program , int X, int Y>
using trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::geo = typename AIE_↩
Program::geo
```

The geography of the CGRA.

Definition at line 66 of file tile.hpp.

9.1.2.18.1.2 geo [2/2]

```
template<typename AIE_Program , int X, int Y>
using trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::geo = typename AIE_↩
Program::geo
```

The geography of the CGRA.

Definition at line 66 of file tile.hpp.

9.1.2.18.1.3 mem_t [1/2]

```
template<typename AIE_Program , int X, int Y>
using trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_t = typename AIE_↩
Program::template tileable_memory<x, y>
```

The type of the memory module native to the tile.

Definition at line 298 of file tile.hpp.

9.1.2.18.1.4 mem_t [2/2]

```
template<typename AIE_Program , int X, int Y>
using trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_t = typename AIE_↩
Program::template tileable_memory<x, y>
```

The type of the memory module native to the tile.

Definition at line 298 of file tile.hpp.

9.1.2.18.1.5 tb [1/2]

```
template<typename AIE_Program , int X, int Y>
using trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::tb = tile_base<AIE_↩
Program>
```

Shortcut to the tile base class.

Definition at line 69 of file tile.hpp.

9.1.2.18.1.6 tb [2/2] template<typename AIE_Program , int X, int Y>

using trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::tb = tile_base<AIE_↩ Program>

Shortcut to the tile base class.

Definition at line 69 of file tile.hpp.

9.1.2.18.2 Member Function Documentation

9.1.2.18.2.1 barrier() [1/2] template<typename AIE_Program , int X, int Y>

void trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::barrier ( ) [inline]

Full barrier using the 2 locks by default.

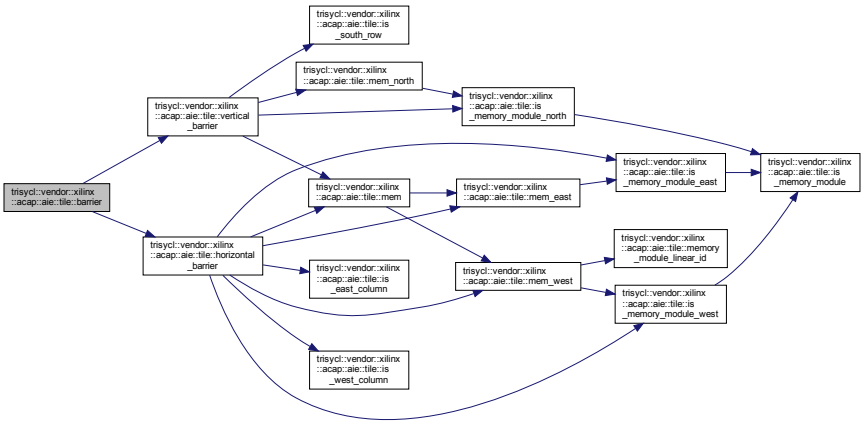
Implement a barrier across the full program by using horizontal_barrier() and vertical_barrier().

Definition at line 430 of file tile.hpp.

```
00430                 {
00431     horizontal_barrier();
00432     vertical_barrier();
00433   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::horizontal_barrier(), and trisycl::vendor::xilinx::acap::aie::tile<

Here is the call graph for this function:



9.1.2.18.2.2 barrier() [2/2] template<typename AIE_Program , int X, int Y>

void trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::barrier ( ) [inline]

Full barrier using the 2 locks by default.

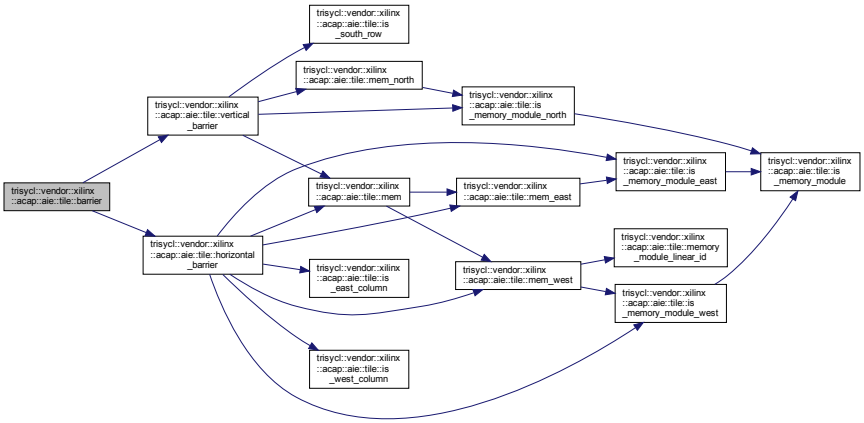
Implement a barrier across the full program by using horizontal_barrier() and vertical_barrier().

Definition at line 430 of file tile.hpp.

```
00430                 {
00431     horizontal_barrier();
00432     vertical_barrier();
00433   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::horizontal_barrier(), and trisycl::vendor::xilinx::acap::aie::tile<

Here is the call graph for this function:



9.1.2.18.2.3 cascade_linear_id() [1/2] template<typename AIE_Program , int X, int Y>

static auto constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::cascade↩
_linear_id ( ) [inline], [static], [constexpr]

Compute the linearized coordinate of a tile along the cascade stream.

Definition at line 315 of file tile.hpp.

```
00315                                 {
00316     return geo::cascade_linear_id(x, y);
00317   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

9.1.2.18.2.4 cascade_linear_id() [2/2] template<typename AIE_Program , int X, int Y>

static auto constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::cascade↩
_linear_id ( ) [inline], [static], [constexpr]

Compute the linearized coordinate of a tile along the cascade stream.

Definition at line 315 of file tile.hpp.

```
00315                                 {
00316     return geo::cascade_linear_id(x, y);
00317   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

9.1.2.18.2.5 get_cascade_stream_in() [1/2]

```
template<typename AIE_Program , int X, int Y>
template<typename T >
auto trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::get_cascade_stream_in ( )
[inline]
```

Get a read accessor to the cascade stream input.

Parameters

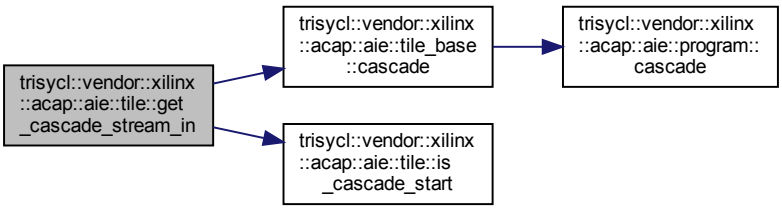
T	is the data type used to read from the cascade stream pipe
Target	is the access mode to the pipe. It is blocking by default

Definition at line 329 of file tile.hpp.

```
00329                                {
00330     static_assert(!is_cascade_start(), "You cannot access to the cascade stream"
00331                   " input on the tile that starts the stream");
00332     return tb::cascade().template get_cascade_stream_in<T>(x, y);
00333   }
```

References trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::cascade(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Progra trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

Here is the call graph for this function:



9.1.2.18.2.6 get_cascade_stream_in() [2/2]

```
template<typename AIE_Program , int X, int Y>
template<typename T >
auto trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::get_cascade_stream_in ( )
[inline]
```

Get a read accessor to the cascade stream input.

Parameters

T	is the data type used to read from the cascade stream pipe
Target	is the access mode to the pipe. It is blocking by default

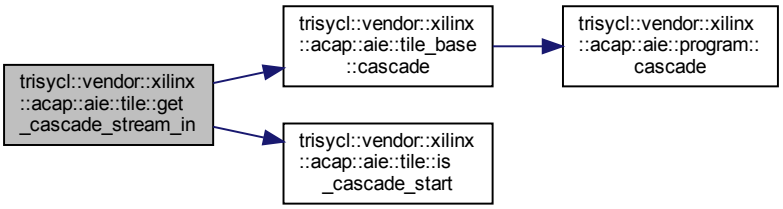
Definition at line 329 of file tile.hpp.

```
00329                                {
00330     static_assert(!is_cascade_start(), "You cannot access to the cascade stream"
00331                   " input on the tile that starts the stream");
```

```
00332     return tb::cascade().template get_cascade_stream_in<T>(x, y);
00333   }
```

References trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::cascade(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Progra
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

Here is the call graph for this function:



9.1.2.18.2.7 get_cascade_stream_out() [1/2] template<typename AIE_Program , int X, int Y>

```
template<typename T >
auto trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::get_cascade_stream_out ( )
[inline]
```

Get a write accessor to the cascade stream output.

Parameters

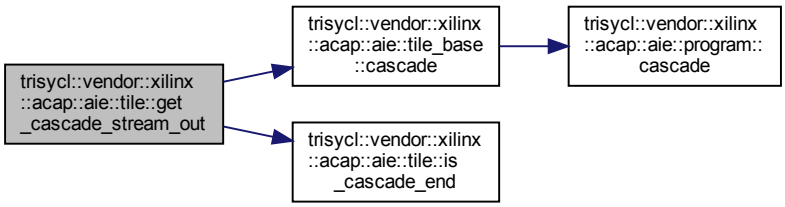
T	is the data type used to write to the cascade stream pipe
Target	is the access mode to the pipe. It is blocking by default

Definition at line 345 of file tile.hpp.

```
00345                                 {
00346     static_assert(!is_cascade_end(), "You cannot access to the cascade stream"
00347                   " output on the tile that starts the stream");
00348     return tb::cascade().template get_cascade_stream_out<T>(x, y);
00349   }
```

References trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::cascade(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Progra
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

Here is the call graph for this function:



9.1.2.18.2.8 get_cascade_stream_out() [2/2]

```
template<typename AIE_Program , int X, int Y>
template<typename T >
auto trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::get_cascade_stream_out ( )
[inline]
```

Get a write accessor to the cascade stream output.

Parameters

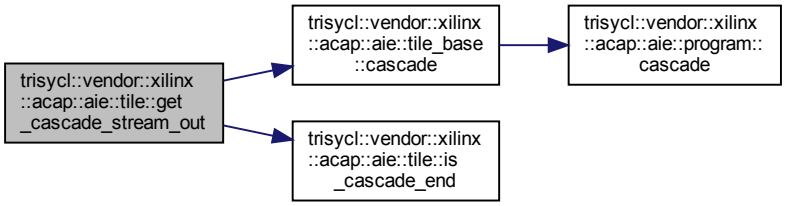
T	is the data type used to write to the cascade stream pipe
Target	is the access mode to the pipe. It is blocking by default

Definition at line 345 of file tile.hpp.

```
00345                                {
00346     static_assert(!is_cascade_end(), "You cannot access to the cascade stream"
00347                   " output on the tile that starts the stream");
00348     return tb::cascade().template get_cascade_stream_out<T>(x, y);
00349   }
```

References trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::cascade(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Progra
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

Here is the call graph for this function:



9.1.2.18.2.9 get_id() [1/2] template<typename AIE_Program , int X, int Y>

```
template<int Dim>
static auto constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::get_id (
)  [inline], [static], [constexpr]
```

Return the coordinate of the tile in the given dimension.

Parameters

Dim	is 0 for the horizontal (x) dimension and 1 for the vertical (y) dimension

Definition at line 78 of file tile.hpp.

```
00078                                {
00079     static_assert(0 <= Dim && Dim <= 1,
00080                   "The dimension has to be between 0 and 1");
00081     if constexpr (Dim == 0)
00082       return x;
00083     else
00084       return y;
00085   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

9.1.2.18.2.10 get_id() [2/2] template<typename AIE_Program , int X, int Y>

```
template<int Dim>
static auto constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::get_id (
)  [inline], [static], [constexpr]
```

Return the coordinate of the tile in the given dimension.

Parameters

Dim	is 0 for the horizontal (x) dimension and 1 for the vertical (y) dimension

Definition at line 78 of file tile.hpp.

```
00078                                {
00079     static_assert(0 <= Dim && Dim <= 1,
00080                   "The dimension has to be between 0 and 1");
00081     if constexpr (Dim == 0)
00082       return x;
00083     else
00084       return y;
00085   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

9.1.2.18.2.11 horizontal_barrier() [1/2] template<typename AIE_Program , int X, int Y>

```
void trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::horizontal_barrier (
             int lock = 14 )  [inline]
```

An horizontal barrier using a lock.

Implement a barrier across the tiles a line.

Parameters

in	*lock*	is the ME lock to use. The lock 14 is used by default

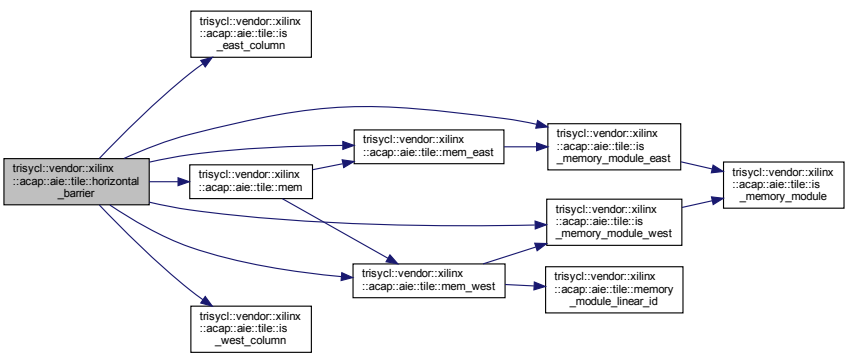
Definition at line 359 of file tile.hpp.

```
00359                                       {
00360     if constexpr (y & 1) {
00361       // Propagate a token from West to East and back
00362       if constexpr (!is_west_column()) {
00363         // Wait for the Western neighbour to be ready
00364         mem().lock(lock).acquire_with_value(true);
00365       }
00366       if constexpr (is_memory_module_east()) {
00367         mem_east().lock(lock).acquire_with_value(false);
00368         // Unleash the Eastern neighbour
00369         mem_east().lock(lock).release_with_value(true);
00370         // Wait for the Eastern neighbour to acknowledge
00371         mem_east().lock(lock).acquire_with_value(false);
00372        }
00373       if constexpr (!is_west_column()) {
00374         // Acknowledge to the Western neighbour
00375         mem().lock(lock).release_with_value(false);
00376       }
00377     } else {
00378       // Propagate a token from East to West and back
00379       if constexpr (!is_east_column()) {
00380         // Wait for the Eastern neighbour to be ready
00381         mem().lock(lock).acquire_with_value(true);
00382       }
00383       if constexpr (is_memory_module_west()) {
00384         mem_west().lock(lock).acquire_with_value(false);
00385         // Unleash the Western neighbour
00386         mem_west().lock(lock).release_with_value(true);
00387         // Wait for the Western neighbour to acknowledge
00388         mem_west().lock(lock).acquire_with_value(false);
00389        }
00390       if constexpr (!is_east_column()) {
00391         // Acknowledge to the Eastern neighbour
00392         mem().lock(lock).release_with_value(false);
00393       }
00394     }
00395   }
```

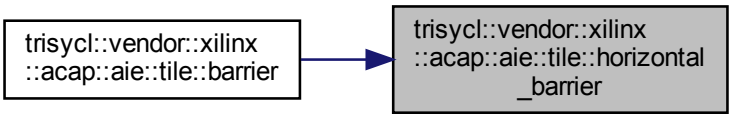
References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_east_column(), trisycl::vendor::xilinx::acap::aie::tile< AIE_
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_west(), trisycl::vendor::xilinx::acap::aie::tile< AIE_P
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_west(), and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X,

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::barrier().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.18.2.12 horizontal_barrier() [2/2]

```
template<typename AIE_Program , int X, int Y>
void trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::horizontal_barrier (
            int lock = 14 )  [inline]
```

An horizontal barrier using a lock.

Implement a barrier across the tiles a line.

Parameters

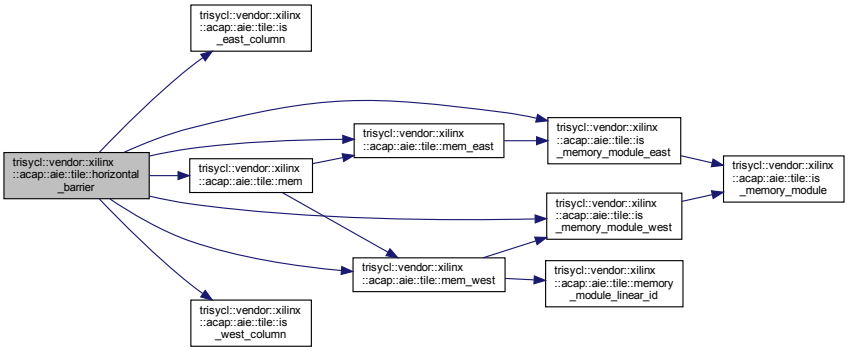
in	*lock*	is the ME lock to use. The lock 14 is used by default

Definition at line 359 of file tile.hpp.

```
00359                                          {
00360     if constexpr (y & 1) {
00361       // Propagate a token from West to East and back
00362       if constexpr (!is_west_column()) {
00363         // Wait for the Western neighbour to be ready
00364         mem().lock(lock).acquire_with_value(true);
00365       }
00366       if constexpr (is_memory_module_east()) {
00367         mem_east().lock(lock).acquire_with_value(false);
00368         // Unleash the Eastern neighbour
00369         mem_east().lock(lock).release_with_value(true);
00370         // Wait for the Eastern neighbour to acknowledge
00371         mem_east().lock(lock).acquire_with_value(false);
00372       }
00373       if constexpr (!is_west_column()) {
00374         // Acknowledge to the Western neighbour
00375         mem().lock(lock).release_with_value(false);
00376       }
00377     } else {
00378       // Propagate a token from East to West and back
00379       if constexpr (!is_east_column()) {
00380         // Wait for the Eastern neighbour to be ready
00381         mem().lock(lock).acquire_with_value(true);
00382       }
00383       if constexpr (is_memory_module_west()) {
00384         mem_west().lock(lock).acquire_with_value(false);
00385         // Unleash the Western neighbour
00386         mem_west().lock(lock).release_with_value(true);
00387         // Wait for the Western neighbour to acknowledge
00388         mem_west().lock(lock).acquire_with_value(false);
00389       }
00390       if constexpr (!is_east_column()) {
00391         // Acknowledge to the Eastern neighbour
00392         mem().lock(lock).release_with_value(false);
00393       }
00394     }
00395   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_east_column(), trisycl::vendor::xilinx::acap::aie::tile< AIE_
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_west(), trisycl::vendor::xilinx::acap::aie::tile< AIE_P
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_west(), and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X,

Here is the call graph for this function:



9.1.2.18.2.13 is_cascade_end() [1/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
cascade_end ( )  [inline], [static], [constexpr]
```

Test if this tile owns the end of the cascade_stream.

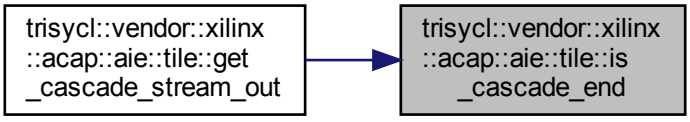
Definition at line 308 of file tile.hpp.

```
00308                                       {
00309     return geo::is_cascade_end(x, y);
00310   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::get_cascade_stream_out().

Here is the caller graph for this function:



9.1.2.18.2.14 is_cascade_end() [2/2] template<typename AIE_Program , int X, int Y>

static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
cascade_end ( ) [inline], [static], [constexpr]

Test if this tile owns the end of the cascade_stream.

Definition at line 308 of file tile.hpp.

```
00308                                        {
00309     return geo::is_cascade_end(x, y);
00310   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

9.1.2.18.2.15 is_cascade_start() [1/2] template<typename AIE_Program , int X, int Y>

static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
cascade_start ( ) [inline], [static], [constexpr]

Test if this tile owns the start of the cascade_stream.

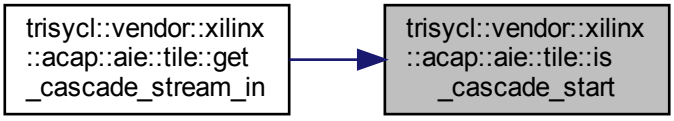
Definition at line 302 of file tile.hpp.

```
00302                                          {
00303     return geo::is_cascade_start(x, y);
00304   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::get_cascade_stream_in().

Here is the caller graph for this function:



9.1.2.18.2.16 is_cascade_start() [2/2] template<typename AIE_Program , int X, int Y>

static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
cascade_start ( ) [inline], [static], [constexpr]

Test if this tile owns the start of the cascade_stream.

Definition at line 302 of file tile.hpp.

```
00302                                          {
00303     return geo::is_cascade_start(x, y);
00304   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

9.1.2.18.2.17 is_east_column() [1/2] template<typename AIE_Program , int X, int Y>

static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_east_column ( ) [inline], [static], [constexpr]

Test if the tile is in the Eastern column.

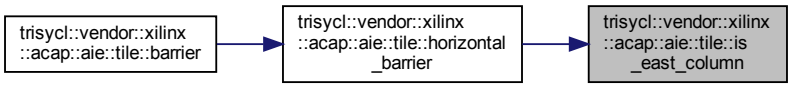
Definition at line 110 of file tile.hpp.

```
00110                                       {
00111     return geo::is_east_column(x);
00112   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x.

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::horizontal_barrier().

Here is the caller graph for this function:



9.1.2.18.2.18 is_east_column() [2/2] template<typename AIE_Program , int X, int Y>

static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_east_column ( ) [inline], [static], [constexpr]

Test if the tile is in the Eastern column.

Definition at line 110 of file tile.hpp.

```
00110                                       {
00111     return geo::is_east_column(x);
00112   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x.

9.1.2.18.2.19 is_east_valid() [1/2] template<typename AIE_Program , int X, int Y>

static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_east_valid ( ) [inline], [static], [constexpr]

Test if the tile has a Eastern neighbor.

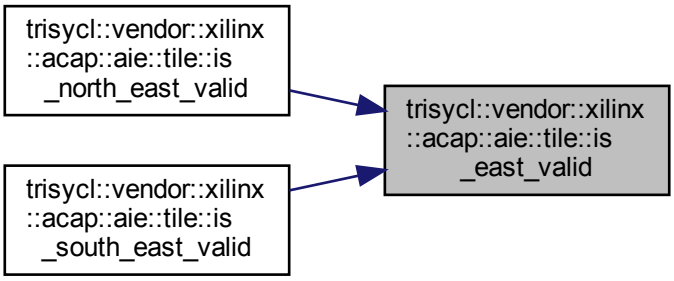
Definition at line 134 of file tile.hpp.

```
00134                                       {
00135     return !geo::is_east_column(x);
00136   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x.

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_north_east_valid(), and trisycl::vendor::xilinx::acap::aie

Here is the caller graph for this function:



9.1.2.18.2.20 is_east_valid() [2/2] template<typename AIE_Program , int X, int Y>

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_east↩
_valid ( )  [inline], [static], [constexpr]
```

Test if the tile has a Eastern neighbor.

Definition at line 134 of file tile.hpp.

```
00134                                       {
00135     return !geo::is_east_column(x);
00136   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x.

9.1.2.18.2.21 is_memory_module() [1/2] template<typename AIE_Program , int X, int Y>

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
memory_module (
              int dx,
              int dy )  [inline], [static], [constexpr]
```

Test if a memory module exists and is connected to this tile.

Parameters

in	*dx*	is the horizontal displacement (-1,0,+1)
in	*dy*	is the vertical displacement (-1,0,+1)

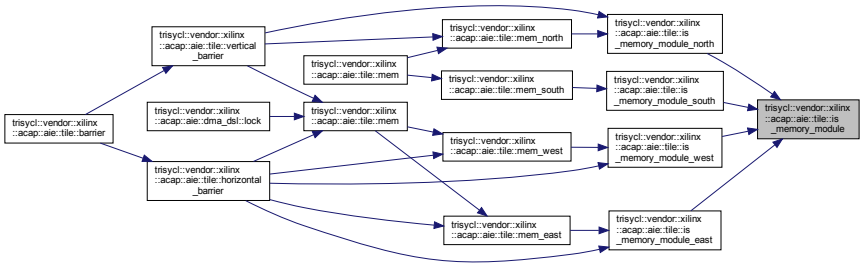
Definition at line 181 of file tile.hpp.

```
00181                                                       {
00182     return geo::is_memory_module(x, y, dx, dy);
00183   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_east(), trisycl::vendor::xilinx::acap::ai
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_south(), and trisycl::vendor::xilinx::acap::aie::tile< A

Here is the caller graph for this function:



9.1.2.18.2.22 is_memory_module() [2/2] template<typename AIE_Program , int X, int Y>

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
memory_module (
            int dx,
            int dy )  [inline], [static], [constexpr]
```

Test if a memory module exists and is connected to this tile.

Parameters

in	*dx*	is the horizontal displacement (-1,0,+1)
in	*dy*	is the vertical displacement (-1,0,+1)

Definition at line 181 of file tile.hpp.

```
00181                                                                    {
00182     return geo::is_memory_module(x, y, dx, dy);
00183   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

9.1.2.18.2.23 is_memory_module_east() [1/2] template<typename AIE_Program , int X, int Y>

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
memory_module_east ( )  [inline], [static], [constexpr]
```

Test if a memory module exists on the East of this tile.

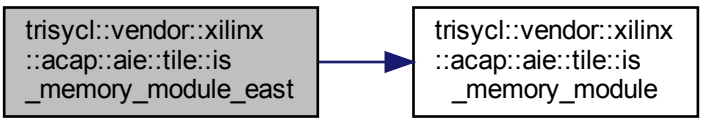
Definition at line 193 of file tile.hpp.

```
00193                                                {
00194     return is_memory_module(1, 0);
00195   }
```

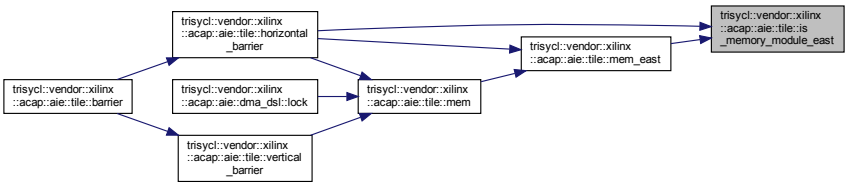
References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module().

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::horizontal_barrier(), and trisycl::vendor::xilinx::acap::aie::t

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.18.2.24 is_memory_module_east() [2/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
memory_module_east ( )  [inline], [static], [constexpr]
```

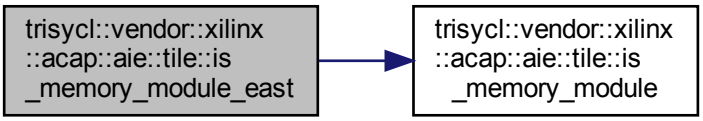
Test if a memory module exists on the East of this tile.

Definition at line 193 of file tile.hpp.

```
00193                                                {
00194     return is_memory_module(1, 0);
00195   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module().

Here is the call graph for this function:



9.1.2.18.2.25 is_memory_module_north() [1/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
memory_module_north ( )  [inline], [static], [constexpr]
```

Test if a memory module exists on the North this tile.

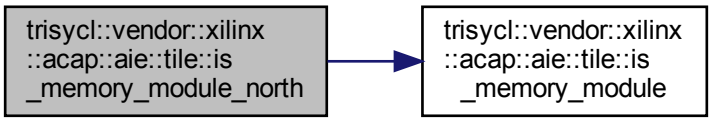
Definition at line 205 of file tile.hpp.

```
00205                                          {
00206     return is_memory_module(0, 1);
00207   }
```

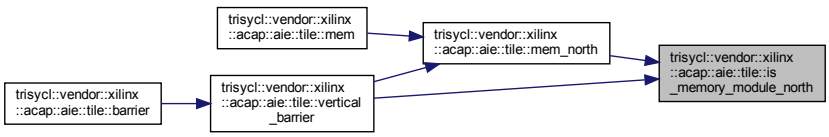
References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module().

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_north(), and trisycl::vendor::xilinx::acap::aie::tile< A

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.18.2.26 is_memory_module_north() [2/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
memory_module_north ( )  [inline], [static], [constexpr]
```

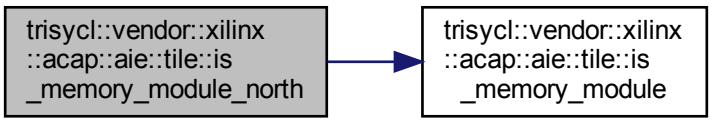
Test if a memory module exists on the North this tile.

Definition at line 205 of file tile.hpp.

```
00205                                          {
00206     return is_memory_module(0, 1);
00207   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module().

Here is the call graph for this function:



9.1.2.18.2.27 is_memory_module_south() [1/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
memory_module_south ( )  [inline], [static], [constexpr]
```

Test if a memory module exists on the South this tile.

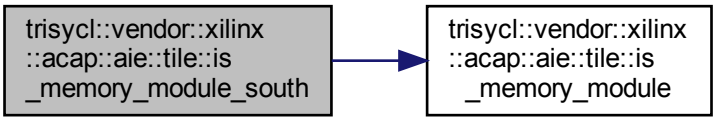
Definition at line 199 of file tile.hpp.

```
00199                                          {
00200     return is_memory_module(0, -1);
00201   }
```

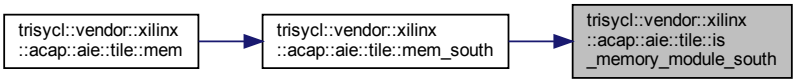
References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module().

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_south().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.18.2.28 is_memory_module_south() [2/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_south ( )  [inline], [static], [constexpr]
```

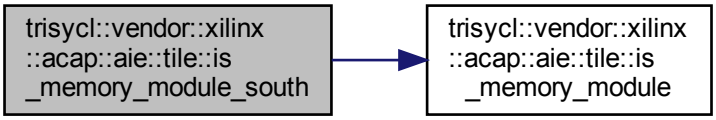
Test if a memory module exists on the South this tile.

Definition at line 199 of file tile.hpp.

```
00199                                                {
00200     return is_memory_module(0, -1);
00201   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module().

Here is the call graph for this function:



9.1.2.18.2.29 is_memory_module_west() [1/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_west ( )  [inline], [static], [constexpr]
```

Test if a memory module exists on the West of this tile.

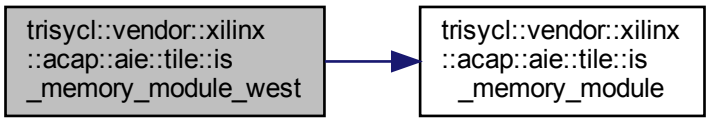
Definition at line 187 of file tile.hpp.

```
00187                                                {
00188     return is_memory_module(-1, 0);
00189   }
```

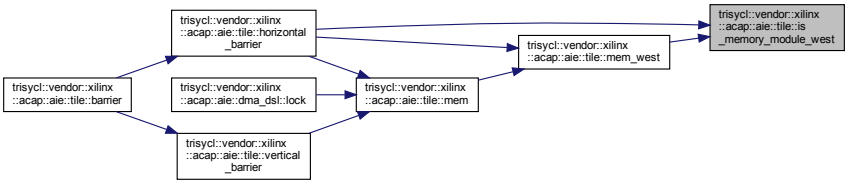
References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module().

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::horizontal_barrier(), and trisycl::vendor::xilinx::acap::aie::t

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.18.2.30 is_memory_module_west() [2/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
memory_module_west ( )  [inline], [static], [constexpr]
```

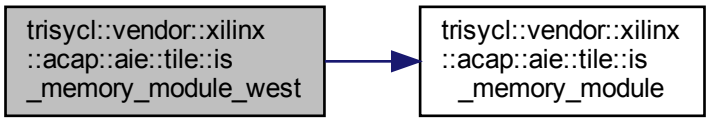
Test if a memory module exists on the West of this tile.

Definition at line 187 of file tile.hpp.

```
00187                                                {
00188     return is_memory_module(-1, 0);
00189   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module().

Here is the call graph for this function:



9.1.2.18.2.31 is_north_east_valid() [1/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
north_east_valid ( )  [inline], [static], [constexpr]
```

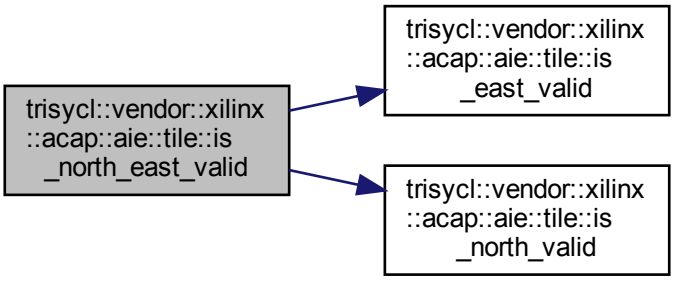
Test if the tile has a North East neighbor.

Definition at line 164 of file tile.hpp.

```
00164                                                {
00165     return is_north_valid() && is_east_valid();
00166   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_east_valid(), and trisycl::vendor::xilinx::acap::aie::tile< AIE

Here is the call graph for this function:



9.1.2.18.2.32 is_north_east_valid() [2/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
north_east_valid ( )  [inline], [static], [constexpr]
```

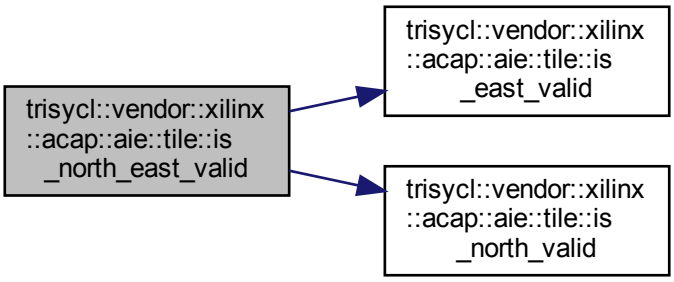
Test if the tile has a North East neighbor.

Definition at line 164 of file tile.hpp.

```
00164                                                {
00165     return is_north_valid() && is_east_valid();
00166   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_east_valid(), and trisycl::vendor::xilinx::acap::aie::tile< AIE

Here is the call graph for this function:



9.1.2.18.2.33 is_north_row() [1/2] template<typename AIE_Program , int X, int Y>

static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_north_row ( ) [inline], [static], [constexpr]

Test if the tile is in the Northern row.

Definition at line 122 of file tile.hpp.

```
00122                                    {
00123     return geo::is_north_row(y);
00124   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

9.1.2.18.2.34 is_north_row() [2/2] template<typename AIE_Program , int X, int Y>

static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_north_row ( ) [inline], [static], [constexpr]

Test if the tile is in the Northern row.

Definition at line 122 of file tile.hpp.

```
00122                                    {
00123     return geo::is_north_row(y);
00124   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

9.1.2.18.2.35 is_north_valid() [1/2] template<typename AIE_Program , int X, int Y>

static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_north_valid ( ) [inline], [static], [constexpr]

Test if the tile has a Northern neighbor.

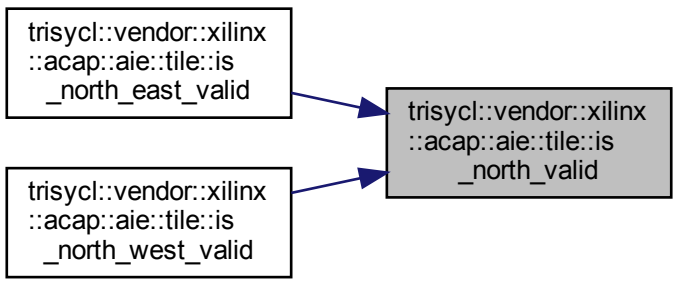
Definition at line 158 of file tile.hpp.

```
00158                                      {
00159     return !geo::is_north_row(y);
00160   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_north_east_valid(), and trisycl::vendor::xilinx::acap::aie

Here is the caller graph for this function:



9.1.2.18.2.36 is_north_valid() [2/2] template<typename AIE_Program , int X, int Y>

static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_north_valid ( ) [inline], [static], [constexpr]

Test if the tile has a Northern neighbor.

Definition at line 158 of file tile.hpp.

```
00158                                       {
00159     return !geo::is_north_row(y);
00160   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

9.1.2.18.2.37 is_north_west_valid() [1/2] template<typename AIE_Program , int X, int Y>

static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_north_west_valid ( ) [inline], [static], [constexpr]

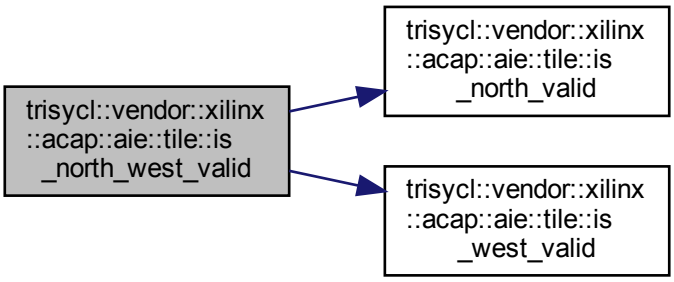
Test if the tile has a North West neighbor.

Definition at line 170 of file tile.hpp.

```
00170                                            {
00171     return is_north_valid() && is_west_valid();
00172   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_north_valid(), and trisycl::vendor::xilinx::acap::aie::tile< AI

Here is the call graph for this function:



9.1.2.18.2.38 is_north_west_valid() [2/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
north_west_valid ( )  [inline], [static], [constexpr]
```

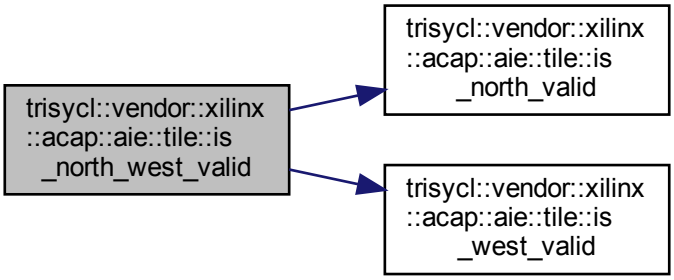
Test if the tile has a North West neighbor.

Definition at line 170 of file tile.hpp.

```
00170                                          {
00171     return is_north_valid() && is_west_valid();
00172   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_north_valid(), and trisycl::vendor::xilinx::acap::aie::tile< AI

Here is the call graph for this function:



9.1.2.18.2.39 is_south_east_valid() [1/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
south_east_valid ( )  [inline], [static], [constexpr]
```

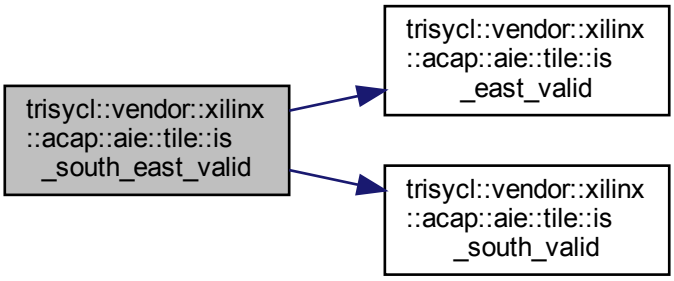
Test if the tile has a South East neighbor.

Definition at line 152 of file tile.hpp.

```
00152                                          {
00153     return is_south_valid() && is_east_valid();
00154   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_east_valid(), and trisycl::vendor::xilinx::acap::aie::tile< AI

Here is the call graph for this function:



9.1.2.18.2.40 is_south_east_valid() [2/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
south_east_valid ( )  [inline], [static], [constexpr]
```

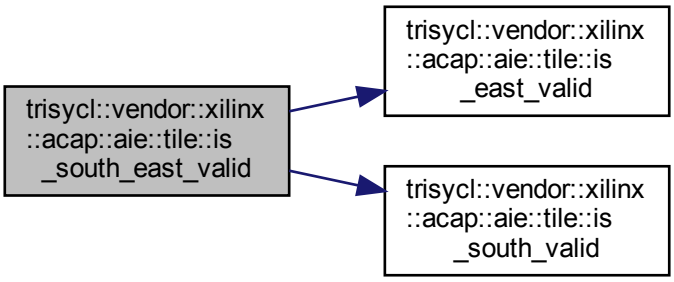
Test if the tile has a South East neighbor.

Definition at line 152 of file tile.hpp.

```
00152                                                {
00153     return is_south_valid() && is_east_valid();
00154   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_east_valid(), and trisycl::vendor::xilinx::acap::aie::tile< AIE

Here is the call graph for this function:



9.1.2.18.2.41 is_south_row() [1/2] template<typename AIE_Program , int X, int Y>

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
south_row ( )  [inline], [static], [constexpr]
```

Test if the tile is in the Southern row.

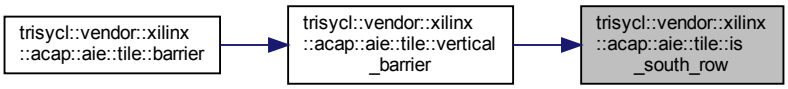
Definition at line 116 of file tile.hpp.

```
00116                                    {
00117     return geo::is_south_row(y);
00118   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::vertical_barrier().

Here is the caller graph for this function:



9.1.2.18.2.42 is_south_row() [2/2] template<typename AIE_Program , int X, int Y>

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
south_row ( )  [inline], [static], [constexpr]
```

Test if the tile is in the Southern row.

Definition at line 116 of file tile.hpp.

```
00116                                    {
00117     return geo::is_south_row(y);
00118   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

9.1.2.18.2.43 is_south_valid() [1/2] template<typename AIE_Program , int X, int Y>

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
south_valid ( )  [inline], [static], [constexpr]
```

Test if the tile has a Southern neighbor.

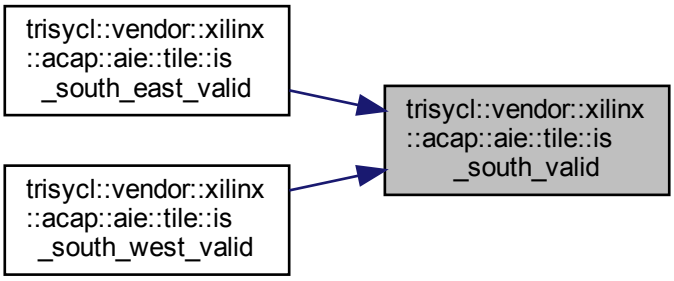
Definition at line 140 of file tile.hpp.

```
00140                                      {
00141     return !geo::is_south_row(y);
00142   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_south_east_valid(), and trisycl::vendor::xilinx::acap::aie

Here is the caller graph for this function:



9.1.2.18.2.44 is_south_valid() [2/2] template<typename AIE_Program , int X, int Y>

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
south_valid ( )  [inline], [static], [constexpr]
```

Test if the tile has a Southern neighbor.

Definition at line 140 of file tile.hpp.

```
00140                                       {
00141     return !geo::is_south_row(y);
00142   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

9.1.2.18.2.45 is_south_west_valid() [1/2] template<typename AIE_Program , int X, int Y>

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
south_west_valid ( )  [inline], [static], [constexpr]
```

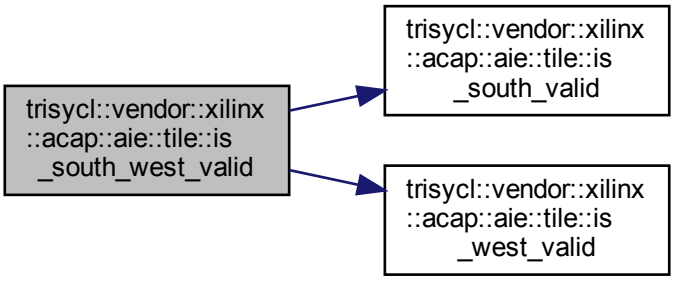
Test if the tile has a South West neighbor.

Definition at line 146 of file tile.hpp.

```
00146                                            {
00147     return is_south_valid() && is_west_valid();
00148   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_south_valid(), and trisycl::vendor::xilinx::acap::aie::tile< A

Here is the call graph for this function:



9.1.2.18.2.46 is_south_west_valid() [2/2]

```
template<typename AIE_Program , int X, int Y>
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_↩
south_west_valid ( )  [inline], [static], [constexpr]
```

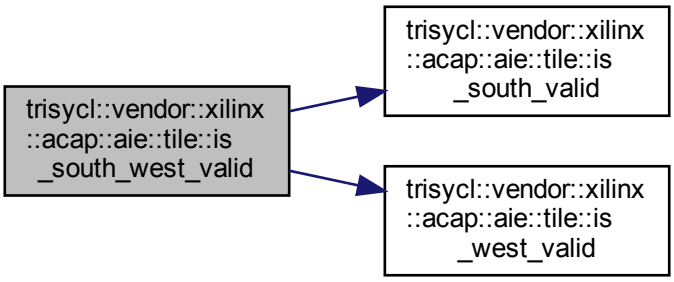
Test if the tile has a South West neighbor.

Definition at line 146 of file tile.hpp.

```
00146                                                {
00147     return is_south_valid() && is_west_valid();
00148   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_south_valid(), and trisycl::vendor::xilinx::acap::aie::tile< A

Here is the call graph for this function:



9.1.2.18.2.47 is_west_column() [1/2] template<typename AIE_Program , int X, int Y>

static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_west_column ( ) [inline], [static], [constexpr]

Test if the tile is in the Western column.

Todo Need something more modern than Boost.Operator to implement trisycl::id

Look for example at https://github.com/taocpp/operators/issues/9 static auto constexpr get_id() { return trisycl::id<2> { x, y }; }

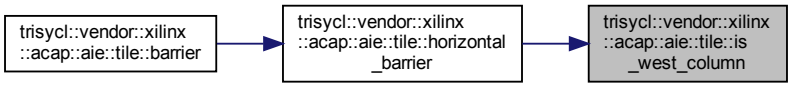
Definition at line 104 of file tile.hpp.

```
00104                                       {
00105     return geo::is_west_column(x);
00106   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x.

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::horizontal_barrier().

Here is the caller graph for this function:



9.1.2.18.2.48 is_west_column() [2/2] template<typename AIE_Program , int X, int Y>

static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_west_column ( ) [inline], [static], [constexpr]

Test if the tile is in the Western column.

Todo Need something more modern than Boost.Operator to implement trisycl::id

Look for example at https://github.com/taocpp/operators/issues/9 static auto constexpr get_id() { return trisycl::id<2> { x, y }; }

Definition at line 104 of file tile.hpp.

```
00104                                       {
00105     return geo::is_west_column(x);
00106   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x.

9.1.2.18.2.49 is_west_valid() [1/2] template<typename AIE_Program , int X, int Y>

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_west↩
_valid ( )  [inline], [static], [constexpr]
```

Test if the tile has a Western neighbor.

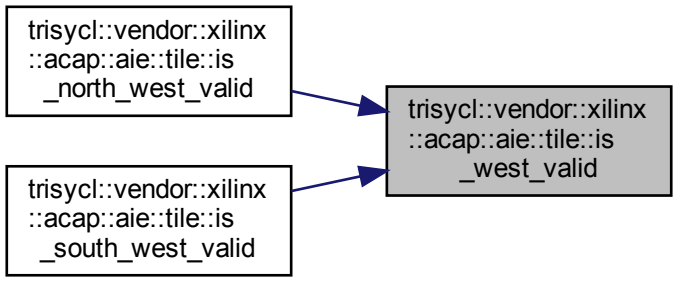
Definition at line 128 of file tile.hpp.

```
00128                                    {
00129     return !geo::is_west_column(x);
00130   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x.

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_north_west_valid(), and trisycl::vendor::xilinx::acap::aie

Here is the caller graph for this function:



9.1.2.18.2.50 is_west_valid() [2/2] template<typename AIE_Program , int X, int Y>

```
static bool constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_west↩
_valid ( )  [inline], [static], [constexpr]
```

Test if the tile has a Western neighbor.

Definition at line 128 of file tile.hpp.

```
00128                                    {
00129     return !geo::is_west_column(x);
00130   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x.

9.1.2.18.2.51 linear_id() [1/2] template<typename AIE_Program , int X, int Y>

static auto constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::linear_id ( ) [inline], [static], [constexpr]

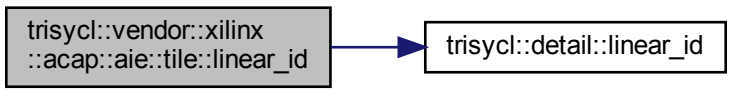
Return the linearized coordinate of the tile.

Definition at line 89 of file tile.hpp.

```
00089                                    {
00090     return geo::linear_id(x, y);
00091   }
```

References trisycl::detail::linear_id(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::a

Here is the call graph for this function:



9.1.2.18.2.52 linear_id() [2/2] template<typename AIE_Program , int X, int Y>

static auto constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::linear_id ( ) [inline], [static], [constexpr]

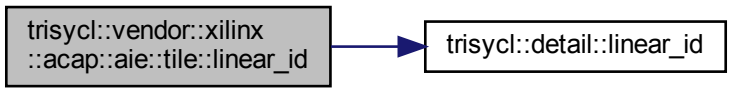
Return the linearized coordinate of the tile.

Definition at line 89 of file tile.hpp.

```
00089                                    {
00090     return geo::linear_id(x, y);
00091   }
```

References trisycl::detail::linear_id(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::a

Here is the call graph for this function:



9.1.2.18.2.53 mem() [1/4] template<typename AIE_Program , int X, int Y>

auto & trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem ( ) [inline]

The memory module native to the tile.

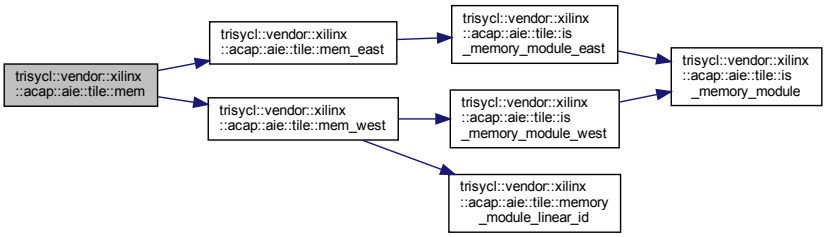
Definition at line 260 of file tile.hpp.

```
00260             {
00261     if constexpr (y & 1)
00262       return mem_west();
00263     else
00264       return mem_east();
00265   }
```

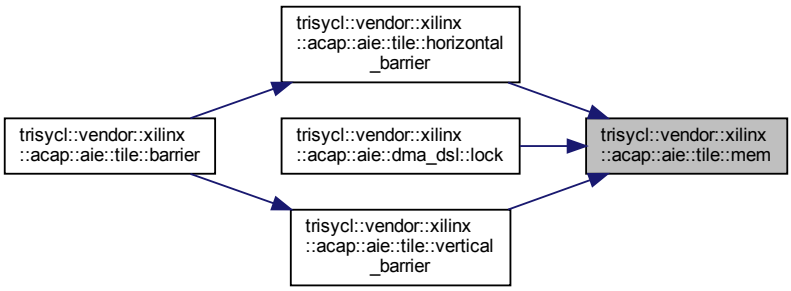
References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_east(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Prog and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::horizontal_barrier(), trisycl::vendor::xilinx::acap::aie::dma and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::vertical_barrier().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.18.2.54 mem() [2/4] template<typename AIE_Program , int X, int Y>

template<int Dx, int Dy>

auto & trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem ( ) [inline]

Get the memory module relative to the tile.

Parameters

in	*Dx*	is the horizontal offset relative to the current core tile
in	*Dy*	is the vertical offset relative to the current core tile

Returns

the memory tile object

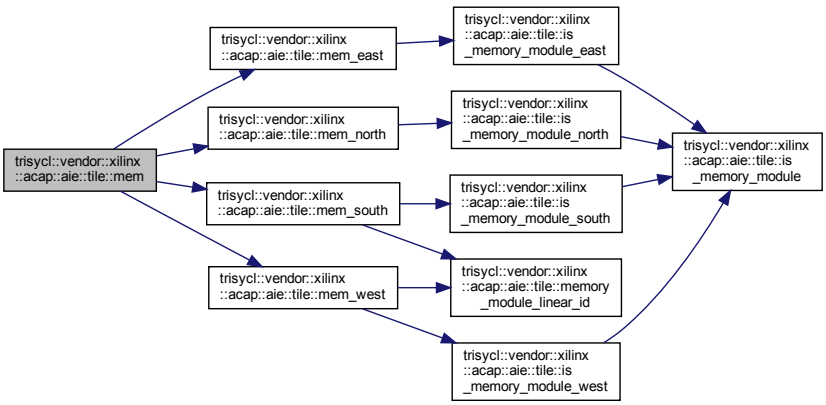
Note that since a core tile has a 4-neighbor connectivity, at least one of the offset needs to be 0 and the other to be either 1 or -1.

Definition at line 281 of file tile.hpp.

```
00281               {
00282     static_assert(geo::is_valid_memory_module_offset(Dx, Dy),
00283                   "Note that since a core tile has a 4-neighbor connectivity,"
00284                   " one of the offset needs to be 0 and the other to be either"
00285                   " 1 or -1");
00286     if constexpr (Dx == -1)
00287       return mem_west();
00288     else if constexpr (Dx == 1)
00289       return mem_east();
00290     else if constexpr (Dy == -1)
00291       return mem_south();
00292     else
00293       return mem_north();
00294   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_east(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Prog
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_south(), and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

Here is the call graph for this function:



9.1.2.18.2.55 mem() [3/4] template<typename AIE_Program , int X, int Y>

```
auto & trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem ( )  [inline]
```

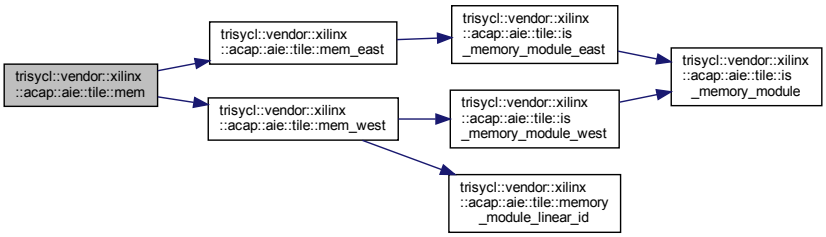
The memory module native to the tile.

Definition at line 260 of file tile.hpp.

```
00260             {
00261     if constexpr (y & 1)
00262       return mem_west();
00263     else
00264       return mem_east();
00265   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_east(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Prog and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y.

Here is the call graph for this function:



9.1.2.18.2.56 mem() [4/4] template<typename AIE_Program , int X, int Y>

```
template<int Dx, int Dy>
auto & trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem ( )  [inline]
```

Get the memory module relative to the tile.

Parameters

in	*Dx*	is the horizontal offset relative to the current core tile
in	*Dy*	is the vertical offset relative to the current core tile

Returns

the memory tile object

Note that since a core tile has a 4-neighbor connectivity, at least one of the offset needs to be 0 and the other to be either 1 or -1.

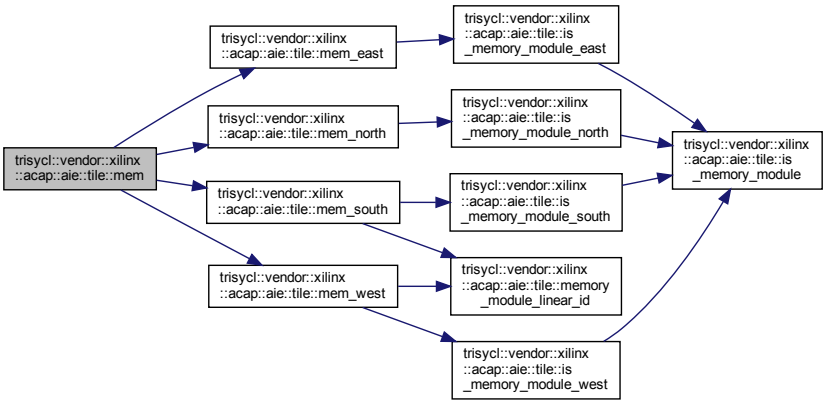
Definition at line 281 of file tile.hpp.

```
00281             {
00282     static_assert(geo::is_valid_memory_module_offset(Dx, Dy),
00283                   "Note that since a core tile has a 4-neighbor connectivity,"
00284                   " one of the offset needs to be 0 and the other to be either"
00285                   " 1 or -1");
00286     if constexpr (Dx == -1)
```

```
00287       return mem_west();
00288     else if constexpr (Dx == 1)
00289       return mem_east();
00290     else if constexpr (Dy == -1)
00291       return mem_south();
00292     else
00293       return mem_north();
00294   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_east(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Prog
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_south(), and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

Here is the call graph for this function:



9.1.2.18.2.57 mem_east() [1/2]

```
template<typename AIE_Program , int X, int Y>
auto & trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_east ( )  [inline]
```

Get the memory module on the East if it does exist.

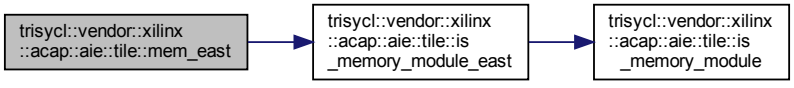
Definition at line 232 of file tile.hpp.

```
00232                    {
00233     static_assert(is_memory_module_east(), "There is no memory module"
00234                   " on the East of this tile in the Eastern column and"
00235                   " on an odd row");
00236     return tb::program->template
00237       memory_module<memory_module_linear_id(1, 0)>();
00238   }
```

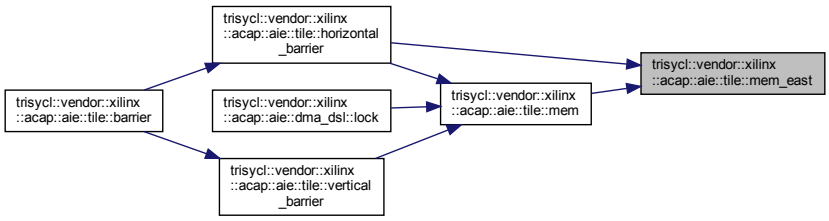
References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_east(), and trisycl::vendor::xilinx::acap::

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::horizontal_barrier(), and trisycl::vendor::xilinx::acap::aie::t

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.18.2.58 mem_east() [2/2] template<typename AIE_Program , int X, int Y>

```
auto & trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_east ( )  [inline]
```

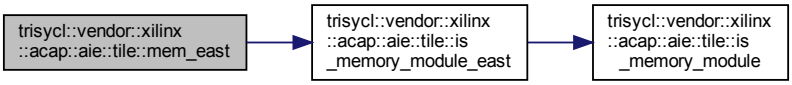
Get the memory module on the East if it does exist.

Definition at line 232 of file tile.hpp.

```
00232                     {
00233     static_assert(is_memory_module_east(), "There is no memory module"
00234                   " on the East of this tile in the Eastern column and"
00235                   " on an odd row");
00236     return tb::program->template
00237       memory_module<memory_module_linear_id(1, 0)>();
00238   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_east(), and trisycl::vendor::xilinx::acap::

Here is the call graph for this function:



9.1.2.18.2.59 mem_north() [1/2] template<typename AIE_Program , int X, int Y>

```
auto & trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_north ( )  [inline]
```

Get the memory module on the North if it does exist.

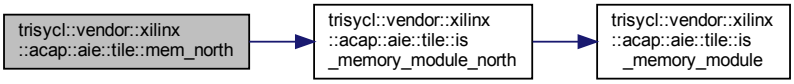
Definition at line 251 of file tile.hpp.

```
00251                     {
00252     static_assert(is_memory_module_north(), "There is no memory module"
00253                   " above the Northern tile row");
00254     return tb::program->template
00255       memory_module<memory_module_linear_id(0, 1)>();
00256   }
```

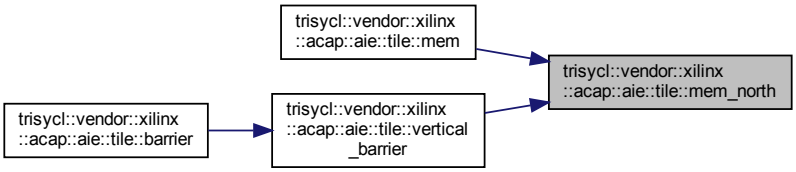
References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_north(), and trisycl::vendor::xilinx::acap:

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem(), and trisycl::vendor::xilinx::acap::aie::tile< AIE_Pro

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.18.2.60 mem_north() [2/2] template<typename AIE_Program , int X, int Y>

```
auto & trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_north ( )  [inline]
```

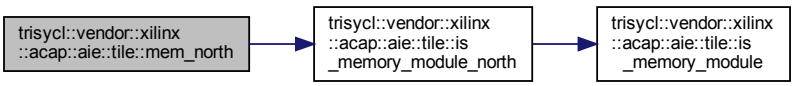
Get the memory module on the North if it does exist.

Definition at line 251 of file tile.hpp.

```
00251                     {
00252     static_assert(is_memory_module_north(), "There is no memory module"
00253                   " above the Northern tile row");
00254     return tb::program->template
00255       memory_module<memory_module_linear_id(0, 1)>();
00256   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_north(), and trisycl::vendor::xilinx::acap:

Here is the call graph for this function:



9.1.2.18.2.61 mem_south() [1/2] template<typename AIE_Program , int X, int Y>

```
auto & trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_south ( )  [inline]
```

Get the memory module on the South if it does exist.

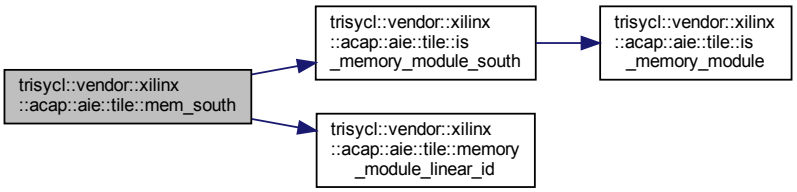
Definition at line 242 of file tile.hpp.

```
00242                       {
00243     static_assert(is_memory_module_south(), "There is no memory module"
00244                   " below the Southern tile row");
00245     return tb::program->template
00246       memory_module<memory_module_linear_id(0, -1)>();
00247   }
```

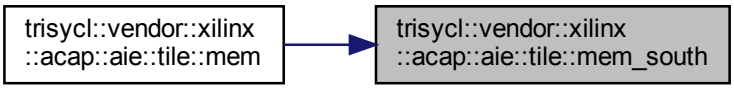
References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_south(), trisycl::vendor::xilinx::acap::aie and trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::program.

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.18.2.62 mem_south() [2/2] template<typename AIE_Program , int X, int Y>

```
auto & trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_south ( )  [inline]
```

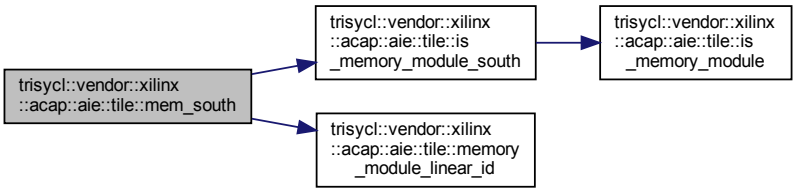
Get the memory module on the South if it does exist.

Definition at line 242 of file tile.hpp.

```
00242                       {
00243     static_assert(is_memory_module_south(), "There is no memory module"
00244                   " below the Southern tile row");
00245     return tb::program->template
00246       memory_module<memory_module_linear_id(0, -1)>();
00247   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_south(), trisycl::vendor::xilinx::acap::aie
and trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::program.

Here is the call graph for this function:



9.1.2.18.2.63 mem_west() [1/2] template<typename AIE_Program , int X, int Y>

```
auto & trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_west ( )  [inline]
```

Get the memory module on the West if it does exist.

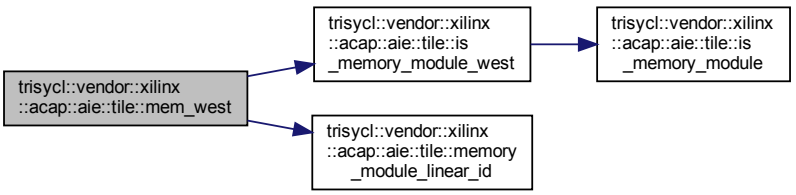
Definition at line 222 of file tile.hpp.

```
00222                     {
00223       static_assert(is_memory_module_west(), "There is no memory module"
00224                     " on the West of this tile in the Western column and"
00225                     " on an even row");
00226       return tb::program->template
00227         memory_module<memory_module_linear_id(-1, 0)>();
00228     }
```

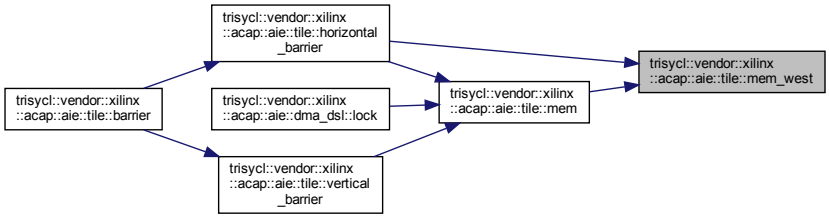
References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_west(), trisycl::vendor::xilinx::acap::aie::
and trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::program.

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::horizontal_barrier(), and trisycl::vendor::xilinx::acap::aie::t

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.18.2.64 mem_west() [2/2]

```
template<typename AIE_Program , int X, int Y>
auto & trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_west ( )  [inline]
```

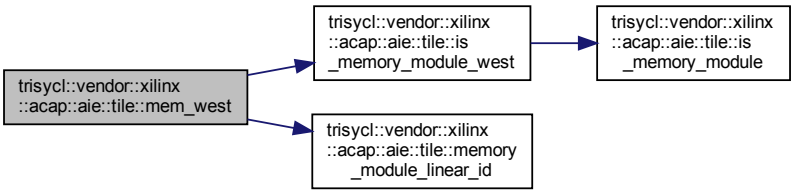
Get the memory module on the West if it does exist.

Definition at line 222 of file tile.hpp.

```
00222                     {
00223       static_assert(is_memory_module_west(), "There is no memory module"
00224                     " on the West of this tile in the Western column and"
00225                     " on an even row");
00226       return tb::program->template
00227         memory_module<memory_module_linear_id(-1, 0)>();
00228     }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_west(), trisycl::vendor::xilinx::acap::aie::
and trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::program.

Here is the call graph for this function:



9.1.2.18.2.65 memory_module_linear_id() [1/2]

```
template<typename AIE_Program , int X, int Y>
static auto constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::memory_↩
module_linear_id (
            int dx,
            int dy )  [inline], [static], [constexpr]
```

Compute the linear id of a memory module attached to this tile.

Parameters

in	*dx*	is the horizontal displacement (-1,0,+1)
in	*dy*	is the vertical displacement (-1,0,+1)

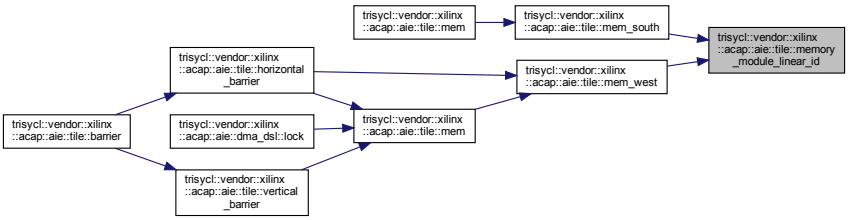
Definition at line 216 of file tile.hpp.

```
00216                                                                    {
00217     return geo::  memory_module_linear_id(x, y, dx, dy);
00218   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_south(), and trisycl::vendor::xilinx::acap::aie::tile< A

Here is the caller graph for this function:



9.1.2.18.2.66 memory_module_linear_id() [2/2]

```
template<typename AIE_Program , int X, int Y>
static auto constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::memory_↩
module_linear_id (
            int dx,
            int dy )  [inline], [static], [constexpr]
```

Compute the linear id of a memory module attached to this tile.

Parameters

in	*dx*	is the horizontal displacement (-1,0,+1)
in	*dy*	is the vertical displacement (-1,0,+1)

Definition at line 216 of file tile.hpp.

```
00216                                                                    {
00217     return geo::  memory_module_linear_id(x, y, dx, dy);
00218   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x, and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

9.1.2.18.2.67 rx_dma() [1/2]

```
template<typename AIE_Program , int X, int Y>
auto trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::rx_dma (
            int port )  [inline]
```

Get access on a receiver DMA.

Parameters

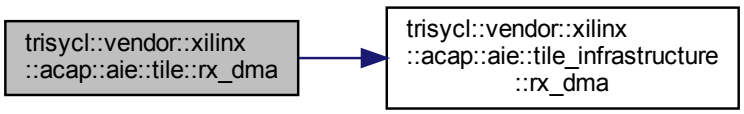
in	*port*	specifies which DMA to access, starting at 0

Definition at line 439 of file tile.hpp.

```
00439                                 {
00440     return dma_dsl { *this, this->ti.rx_dma(port) };
00441   }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::rx_dma(), and trisycl::vendor::xilinx::acap::aie::tile_bas

Here is the call graph for this function:



9.1.2.18.2.68 rx_dma() [2/2]

```
template<typename AIE_Program , int X, int Y>
auto trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::rx_dma (
            int port )  [inline]
```

Get access on a receiver DMA.

Parameters

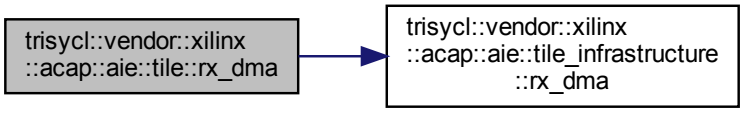
in	*port*	specifies which DMA to access, starting at 0

Definition at line 439 of file tile.hpp.

```
00439                                 {
00440     return dma_dsl { *this, this->ti.rx_dma(port) };
00441   }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::rx_dma(), and trisycl::vendor::xilinx::acap::aie::tile_bas

Here is the call graph for this function:



9.1.2.18.2.69 tx_dma() [1/2]

```
template<typename AIE_Program , int X, int Y>
auto trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::tx_dma (
            int port )  [inline]
```

Get access on a transmit DMA.

Parameters

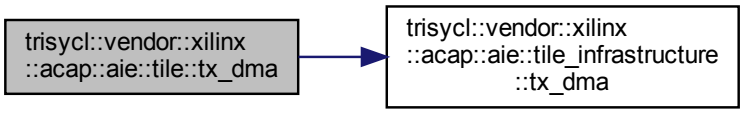
in	*port*	specifies which DMA to access, starting at 0

Definition at line 447 of file tile.hpp.

```
00447                                {
00448     return dma_dsl { *this, this->ti.tx_dma(port) };
00449   }
```

References trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::ti, and trisycl::vendor::xilinx::acap::aie::tile_infrastructure< G

Here is the call graph for this function:



9.1.2.18.2.70 tx_dma() [2/2]

```
template<typename AIE_Program , int X, int Y>
auto trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::tx_dma (
            int port )  [inline]
```

Get access on a transmit DMA.

Parameters

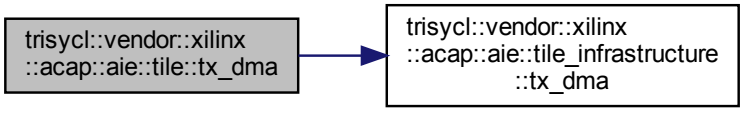
in	*port*	specifies which DMA to access, starting at 0

Definition at line 447 of file tile.hpp.

```
00447                                {
00448     return dma_dsl { *this, this->ti.tx_dma(port) };
00449   }
```

References trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::ti, and trisycl::vendor::xilinx::acap::aie::tile_infrastructure< G

Here is the call graph for this function:



9.1.2.18.2.71 vertical_barrier() [1/2]

```
template<typename AIE_Program , int X, int Y>
void trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::vertical_barrier (
            int lock = 15 )  [inline]
```

A vertical barrier using a lock.

Implement a barrier across the tiles a line.

Parameters

in	*lock*	is the ME lock to use. The lock 15 is used by default

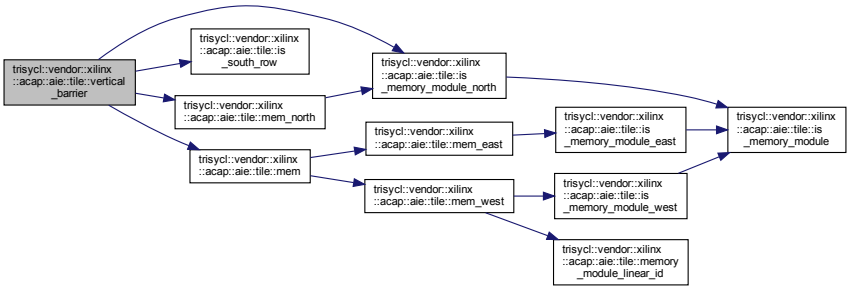
Definition at line 405 of file tile.hpp.

```
00405                                       {
00406     // Propagate a token from South to North and back
00407     if constexpr (!is_south_row()) {
00408       // Wait for the Southern neighbour to be ready
00409       mem().lock(lock).acquire_with_value(true);
00410     }
00411     if constexpr (is_memory_module_north()) {
00412       mem_north().lock(lock).acquire_with_value(false);
00413       // Unleash the Northern neighbour
00414       mem_north().lock(lock).release_with_value(true);
00415       // Wait for the Northern neighbour to acknowledge
00416       mem_north().lock(lock).acquire_with_value(false);
00417     }
00418     if constexpr (!is_south_row()) {
00419       // Acknowledge to the Southern neighbour
00420       mem().lock(lock).release_with_value(false);
00421     }
00422   }
```

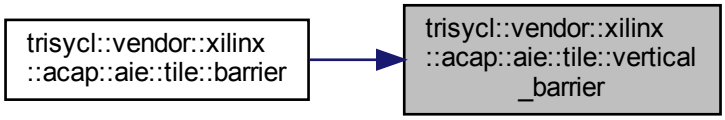
References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_north(), trisycl::vendor::xilinx::acap::aie:
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem(), and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::barrier().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.18.2.72 vertical_barrier() [2/2] template<typename AIE_Program , int X, int Y>

```
void trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::vertical_barrier (
            int lock = 15 )  [inline]
```

A vertical barrier using a lock.

Implement a barrier across the tiles a line.

Parameters

in	*lock*	is the ME lock to use. The lock 15 is used by default

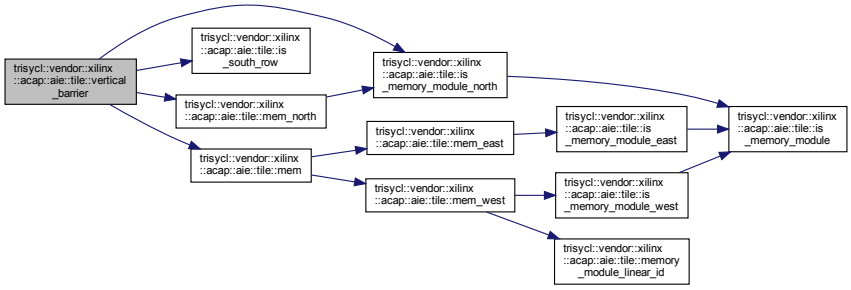
Definition at line 405 of file tile.hpp.

```
00405                                       {
00406     // Propagate a token from South to North and back
00407     if constexpr (!is_south_row()) {
00408       // Wait for the Southern neighbour to be ready
00409       mem().lock(lock).acquire_with_value(true);
00410     }
00411     if constexpr (is_memory_module_north()) {
00412       mem_north().lock(lock).acquire_with_value(false);
00413       // Unleash the Northern neighbour
00414       mem_north().lock(lock).release_with_value(true);
00415       // Wait for the Northern neighbour to acknowledge
00416       mem_north().lock(lock).acquire_with_value(false);
00417     }
```

```
00418     if constexpr (!is_south_row()) {
00419       // Acknowledge to the Southern neighbour
00420       mem().lock(lock).release_with_value(false);
00421     }
00422   }
```

References trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module_north(), trisycl::vendor::xilinx::acap::aie:
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem(), and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::

Here is the call graph for this function:



9.1.2.18.3 Member Data Documentation

9.1.2.18.3.1 x `template<typename AIE_Program , int X, int Y>`

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::x = X
[static], [constexpr]
```

The horizontal tile coordinates in the CGRA grid (starting at 0 and increasing towards the East)

Definition at line 60 of file tile.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::cascade_linear_id(), trisycl::vendor::xilinx::acap::aie::tile<
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::get_cascade_stream_out(), trisycl::vendor::xilinx::acap::aie::tile< AIE_P
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_cascade_end(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_east_column(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_memory_module(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Prograr
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_west_valid(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y
and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::memory_module_linear_id().

9.1.2.18.3.2 y `template<typename AIE_Program , int X, int Y>`

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::y = Y
[static], [constexpr]
```

The vertical tile coordinates in the CGRA grid (starting at increasing towards the North)

Definition at line 63 of file tile.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::cascade_linear_id(), trisycl::vendor::xilinx::acap::aie::tile<
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::get_cascade_stream_out(), trisycl::vendor::xilinx::acap::aie::tile< AIE_P
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::horizontal_barrier(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Program,
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_cascade_start(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Program,
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_north_row(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::is_south_row(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::linear_id(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::r
and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::memory_module_linear_id().

9.1.2.19 class trisycl::vendor::xilinx::acap::aie::tile_base

template<typename AIE_Program>
class trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >

The AI Engine tile infrastructure common to all the tiles.

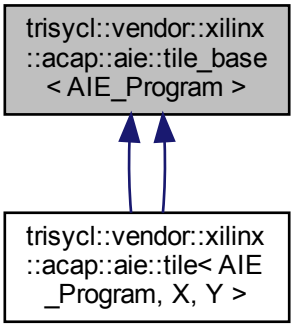
This allows some type erasure while accessing the common tile infrastructure.

Parameters

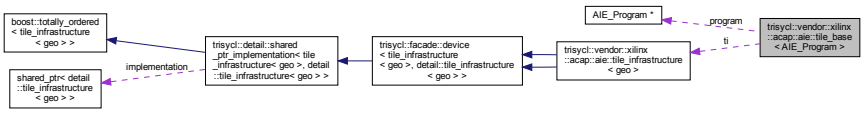
AIE_Program	is the type representing the full CGRA program with the tile programs and memory contents

Definition at line 32 of file tile_base.hpp.

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >:



Public Member Functions

- void run ()

  Provide a run member function that does nothing so it is possible to write a minimum AI Engine program that does nothing.

- template<typename Work >
  void single_task (Work &&f)

  Submit a callable on this tile.

- void wait ()

  Wait for the execution of the callable on this tile.
- auto & cascade ()

  Access the cascade connections.
- auto & in_connection (int port)

  Get the user input connection from the AXI stream switch.
- auto & out_connection (int port)

  Get the user output connection to the AXI stream switch.
- auto & in (int port)

  Get the user input port from the AXI stream switch.
- auto & out (int port)

  Get the user output port to the AXI stream switch.
- void set_program (AIE_Program &p)

  Store a way to access to the program.
- void set_tile_infrastructure (tile_infrastructure< geo > t)

  Store a way to access to hardware infrastructure of the tile.
- void run ()

  Provide a run member function that does nothing so it is possible to write a minimum AI Engine program that does nothing.
- template<typename Work >
  void single_task (Work &&f)

  Submit a callable on this tile.
- void wait ()

  Wait for the execution of the callable on this tile.
- auto & cascade ()

  Access the cascade connections.
- auto & in_connection (int port)

  Get the user input connection from the AXI stream switch.
- auto & out_connection (int port)

  Get the user output connection to the AXI stream switch.
- auto & in (int port)

  Get the user input port from the AXI stream switch.
- auto & out (int port)

  Get the user output port to the AXI stream switch.
- void set_program (AIE_Program &p)

  Store a way to access to the program.
- void set_tile_infrastructure (tile_infrastructure< geo > t)

  Store a way to access to hardware infrastructure of the tile.

Protected Attributes

- AIE_Program * program

  Keep a reference to the AIE_Program with the full tile and memory view.
- tile_infrastructure< geo > ti

  Keep a reference to the tile_infrastructure hardware features.

Private Types

- using geo = typename AIE_Program::device::geo
- using geo = typename AIE_Program::device::geo

9.1.2.19.1 Member Typedef Documentation

9.1.2.19.1.1 geo [1/2] template<typename AIE_Program >

```
using trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::geo = typename AIE_↩
Program::device::geo  [private]
```

Definition at line 34 of file tile_base.hpp.

9.1.2.19.1.2 geo [2/2] template<typename AIE_Program >

```
using trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::geo = typename AIE_↩
Program::device::geo  [private]
```

Definition at line 34 of file tile_base.hpp.

9.1.2.19.2 Member Function Documentation

9.1.2.19.2.1 cascade() [1/2] template<typename AIE_Program >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::cascade ( )  [inline]
```

Access the cascade connections.

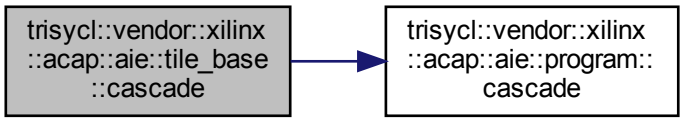
Definition at line 72 of file tile_base.hpp.

```
00072                 {
00073     // \todo the cascade should be part of the tile infrastructure instead
00074     return program->cascade();
00075   }
```

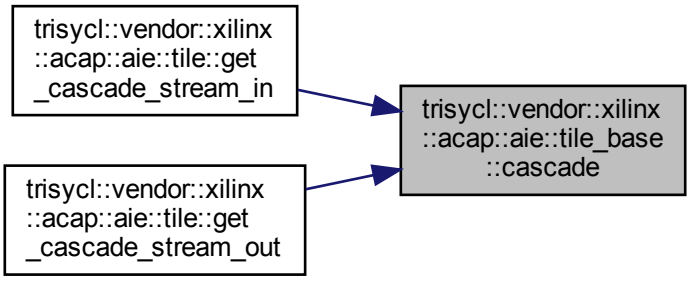
References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::cascade().

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::get_cascade_stream_in(), and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::get_cascade_stream_out().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.19.2.2 cascade() [2/2] template<typename AIE_Program >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::cascade ( )  [inline]
```

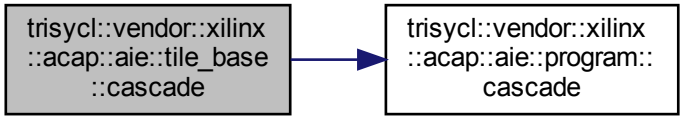
Access the cascade connections.

Definition at line 72 of file tile_base.hpp.

```
00072                 {
00073     // \todo the cascade should be part of the tile infrastructure instead
00074     return program->cascade();
00075   }
```

References trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::cascade().

Here is the call graph for this function:



9.1.2.19.2.3 in() [1/2] template<typename AIE_Program >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::in (
            int port )  [inline]
```

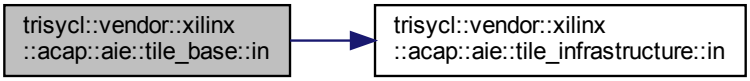
Get the user input port from the AXI stream switch.

Definition at line 97 of file tile_base.hpp.

```
00097                     {
00098     return ti.in(port);
00099   }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::in(), and trisycl::vendor::xilinx::acap::aie::tile_base< A

Here is the call graph for this function:



9.1.2.19.2.4 in() [2/2] template<typename AIE_Program >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::in (
            int port )  [inline]
```

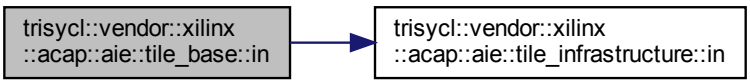
Get the user input port from the AXI stream switch.

Definition at line 97 of file tile_base.hpp.

```
00097                     {
00098     return ti.in(port);
00099   }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::in(), and trisycl::vendor::xilinx::acap::aie::tile_base< A

Here is the call graph for this function:



9.1.2.19.2.5 in_connection() [1/2] template<typename AIE_Program >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::in_connection (
            int port )  [inline]
```

Get the user input connection from the AXI stream switch.

Parameters

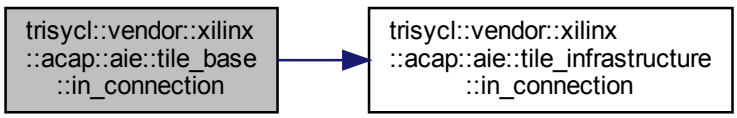
in	*port*	is the port to use

Definition at line 82 of file tile_base.hpp.

```
00082                                 {
00083     return ti.in_connection(port);
00084   }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::in_connection(), and trisycl::vendor::xilinx::acap::aie::t

Here is the call graph for this function:



9.1.2.19.2.6 in_connection() [2/2] template<typename AIE_Program >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::in_connection (
            int port )  [inline]
```

Get the user input connection from the AXI stream switch.

Parameters

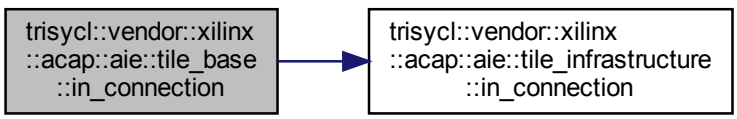
in	*port*	is the port to use

Definition at line 82 of file tile_base.hpp.

```
00082                                 {
00083     return ti.in_connection(port);
00084   }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::in_connection(), and trisycl::vendor::xilinx::acap::aie::t

Here is the call graph for this function:



9.1.2.19.2.7 out() [1/2] template<typename AIE_Program >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::out (
            int port )  [inline]
```

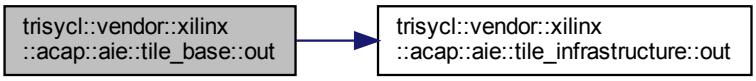
Get the user output port to the AXI stream switch.

Definition at line 103 of file tile_base.hpp.

```
00103                           {
00104     return ti.out(port);
00105   }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::out(), and trisycl::vendor::xilinx::acap::aie::tile_base<

Here is the call graph for this function:



9.1.2.19.2.8 out() [2/2] template<typename AIE_Program >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::out (
            int port )  [inline]
```

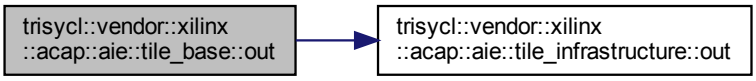
Get the user output port to the AXI stream switch.

Definition at line 103 of file tile_base.hpp.

```
00103                           {
00104     return ti.out(port);
00105   }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::out(), and trisycl::vendor::xilinx::acap::aie::tile_base<

Here is the call graph for this function:



9.1.2.19.2.9 out_connection() [1/2] template<typename AIE_Program >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::out_connection (
            int port )  [inline]
```

Get the user output connection to the AXI stream switch.

Parameters

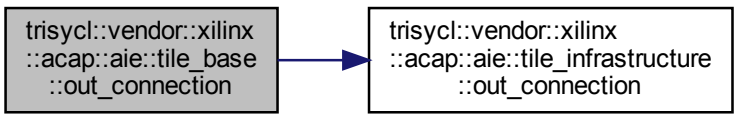
in	*port*	is port to use

Definition at line 91 of file tile_base.hpp.

```
00091                                 {
00092     return ti.out_connection(port);
00093   }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::out_connection(), and trisycl::vendor::xilinx::acap::aie:

Here is the call graph for this function:



9.1.2.19.2.10 out_connection() [2/2] template<typename AIE_Program >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::out_connection (
            int port )  [inline]
```

Get the user output connection to the AXI stream switch.

Parameters

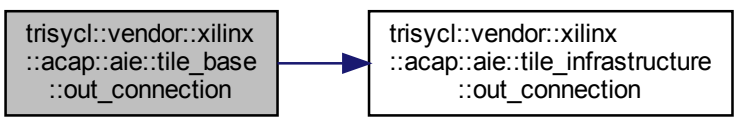
in	*port*	is port to use

Definition at line 91 of file tile_base.hpp.

```
00091                                 {
00092     return ti.out_connection(port);
00093   }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::out_connection(), and trisycl::vendor::xilinx::acap::aie:

Here is the call graph for this function:



9.1.2.19.2.11 run() [1/2] template<typename AIE_Program >

void trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::run ( ) [inline]

Provide a run member function that does nothing so it is possible to write a minimum AI Engine program that does nothing.

Note that even if this function is not virtual, in the common case a programmer implements it to specify the program executed by a tile

Definition at line 54 of file tile_base.hpp.

```
00054             {
00055   }
```

9.1.2.19.2.12 run() [2/2] template<typename AIE_Program >

void trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::run ( ) [inline]

Provide a run member function that does nothing so it is possible to write a minimum AI Engine program that does nothing.

Note that even if this function is not virtual, in the common case a programmer implements it to specify the program executed by a tile

Definition at line 54 of file tile_base.hpp.

```
00054             {
00055   }
```

9.1.2.19.2.13 set_program() [1/2] template<typename AIE_Program >

void trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::set_program (
AIE_Program & *p*) [inline]

Store a way to access to the program.

Definition at line 109 of file tile_base.hpp.

```
00109                                   {
00110     program = &p;
00111   }
```

References p.

9.1.2.19.2.14 set_program() [2/2] template<typename AIE_Program >

void trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::set_program (
AIE_Program & *p*) [inline]

Store a way to access to the program.

Definition at line 109 of file tile_base.hpp.

```
00109                                   {
00110     program = &p;
00111   }
```

References p.

9.1.2.19.2.15 set_tile_infrastructure() [1/2] template<typename AIE_Program >

```
void trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::set_tile_infrastructure (
            tile_infrastructure< geo > t )  [inline]
```

Store a way to access to hardware infrastructure of the tile.

Definition at line 115 of file tile_base.hpp.

```
00115                                                    {
00116     ti = t;
00117   }
```

References trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::ti.

9.1.2.19.2.16 set_tile_infrastructure() [2/2] template<typename AIE_Program >

```
void trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::set_tile_infrastructure (
            tile_infrastructure< geo > t )  [inline]
```

Store a way to access to hardware infrastructure of the tile.

Definition at line 115 of file tile_base.hpp.

```
00115                                                    {
00116     ti = t;
00117   }
```

References trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::ti.

9.1.2.19.2.17 single_task() [1/2] template<typename AIE_Program >

```
template<typename Work >
void trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::single_task (
            Work && f )  [inline]
```

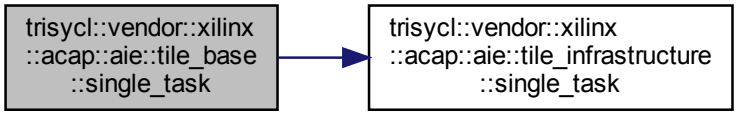
Submit a callable on this tile.

Definition at line 60 of file tile_base.hpp.

```
00060                                {
00061     ti.single_task(std::forward<Work>(f));
00062   }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::single_task(), and trisycl::vendor::xilinx::acap::aie::tile_

Here is the call graph for this function:



9.1.2.19.2.18 single_task() [2/2] template<typename AIE_Program >

```
template<typename Work >
void trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::single_task (
            Work && f )  [inline]
```

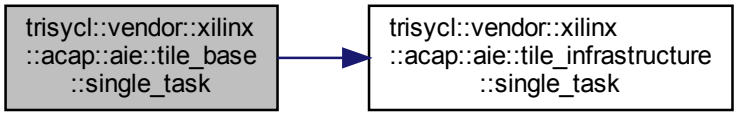
Submit a callable on this tile.

Definition at line 60 of file tile_base.hpp.

```
00060                                {
00061     ti.single_task(std::forward<Work>(f));
00062   }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::single_task(), and trisycl::vendor::xilinx::acap::aie::tile_

Here is the call graph for this function:



9.1.2.19.2.19 wait() [1/2] template<typename AIE_Program >

```
void trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::wait ( )  [inline]
```

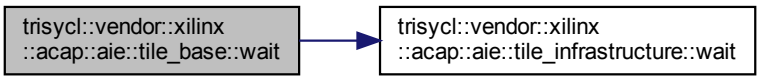
Wait for the execution of the callable on this tile.

Definition at line 66 of file tile_base.hpp.

```
00066                {
00067     ti.wait();
00068   }
```

References trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::ti, and trisycl::vendor::xilinx::acap::aie::tile_infrastructure< G

Here is the call graph for this function:



9.1.2.19.2.20 wait() [2/2] template<typename AIE_Program >

void trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::wait ( ) [inline]

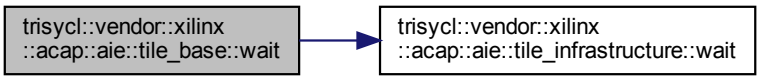
Wait for the execution of the callable on this tile.

Definition at line 66 of file tile_base.hpp.

```
00066               {
00067     ti.wait();
00068   }
```

References trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::ti, and trisycl::vendor::xilinx::acap::aie::tile_infrastructure< G

Here is the call graph for this function:



9.1.2.19.3 Member Data Documentation

9.1.2.19.3.1 program template<typename AIE_Program >

AIE_Program * trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::program [protected]

Keep a reference to the AIE_Program with the full tile and memory view.

Todo can probably be removed

Todo can probably be removed

Definition at line 40 of file tile_base.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_east(), trisycl::vendor::xilinx::acap::aie::tile< AIE_P trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::mem_south(), and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X

9.1.2.19.3.2 ti template<typename AIE_Program >

tile_infrastructure< geo > trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::ti [protected]

Keep a reference to the tile_infrastructure hardware features.

Definition at line 43 of file tile_base.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::in(), trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Pro trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::out(), trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::out_ trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::rx_dma(), trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::s trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::single_task(), trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y and trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::wait().

9.1.2.20 class trisycl::vendor::xilinx::acap::aie::tile_infrastructure

template<typename Geography>
class trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >

The AI Engine tile infrastructure common to all the tiles.

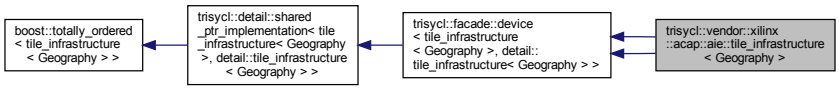
This allows some type erasure while accessing the common tile infrastructure.

Parameters

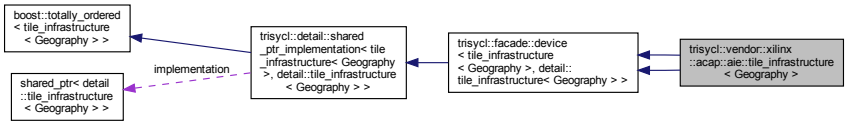
Geography	is the `geography` type representing the full layout of the architecture

Definition at line 42 of file tile_infrastructure.hpp.

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >:



Public Types

- using facade_t = facade::device< tile_infrastructure< Geography >, dti >

  The façade used to implement part of the user-facing type.
- using geo = Geography
- using axi_ss_geo = typename dti::axi_ss_geo
- using mpl = typename dti::mpl
- using spl = typename dti::spl
- using axi_ss_t = typename dti::axi_ss_t
- using facade_t = facade::device< tile_infrastructure< Geography >, dti >

  The façade used to implement part of the user-facing type.
- using geo = Geography
- using axi_ss_geo = typename dti::axi_ss_geo
- using mpl = typename dti::mpl
- using spl = typename dti::spl
- using axi_ss_t = typename dti::axi_ss_t

Public Member Functions

- tile_infrastructure (int x, int y, auto &dev, ::trisycl::detail::fiber_pool &fiber_executor)

  Start the tile infrastructure associated to the AIE device.
- tile_infrastructure ()=default
- int x ()

  Get the horizontal coordinate.
- int y ()

Get the vertical coordinate.

- auto & mem ()

  Access to the common infrastructure part of a tile memory.

- template<typename Invocable >
  void submit (Invocable &&f) const

  Submit a command group to the tile.

- auto & in_connection (int port)

  Get the user input connection from the AXI stream switch.

- auto & out_connection (int port)

  Get the user output connection to the AXI stream switch.

- auto & in (int port)

  Get the user input port from the AXI stream switch.

- auto & out (int port)

  Get the user output port to the AXI stream switch.

- auto & rx_dma (int id)

  Get access to a receiver DMA.

- auto & tx_dma (int id)

  Get access to a transmit DMA.

- auto & input (spl p)

  Get the input router port of the AXI stream switch.

- auto & output (mpl p)

  Get the output router port of the AXI stream switch.

- template<typename Work >
  auto & single_task (Work &&f)

  Launch a callable on this tile.

- auto & wait ()

  Wait for the execution of the callable on this tile.

- auto & connect (typename geo::core_axi_stream_switch::slave_port_layout sp, typename geo::core_axi_↩
  stream_switch::master_port_layout mp)

  Configure a connection of the core tile AXI stream switch.

- void display (latex::context &c) const

  Display the tile to a LaTeX context.

- tile_infrastructure (int x, int y, auto &dev, ::trisycl::detail::fiber_pool &fiber_executor)

  Start the tile infrastructure associated to the AIE device.

- tile_infrastructure ()=default

- int x ()

  Get the horizontal coordinate.

- int y ()

  Get the vertical coordinate.

- auto & mem ()

  Access to the common infrastructure part of a tile memory.

- template<typename Invocable >
  void submit (Invocable &&f) const

  Submit a command group to the tile.

- auto & in_connection (int port)

  Get the user input connection from the AXI stream switch.

- auto & out_connection (int port)

  Get the user output connection to the AXI stream switch.

- auto & in (int port)

  Get the user input port from the AXI stream switch.

- auto & out (int port)

Get the user output port to the AXI stream switch.

- auto & rx_dma (int id)

  Get access to a receiver DMA.
- auto & tx_dma (int id)

  Get access to a transmit DMA.
- auto & input (spl p)

  Get the input router port of the AXI stream switch.
- auto & output (mpl p)

  Get the output router port of the AXI stream switch.
- template<typename Work >
  auto & single_task (Work &&f)

  Launch a callable on this tile.
- auto & wait ()

  Wait for the execution of the callable on this tile.
- auto & connect (typename geo::core_axi_stream_switch::slave_port_layout sp, typename geo::core_axi_↩ stream_switch::master_port_layout mp)

  Configure a connection of the core tile AXI stream switch.
- void display (latex::context &c) const

  Display the tile to a LaTeX context.

Static Public Member Functions

- static constexpr int x_size ()

  Get the horizontal number of tiles.
- static constexpr int y_size ()

  Get the vertical number of tiles.
- static vec< int, 2 > display_size ()

  Compute the size of the graphics representation of the tile.
- static constexpr int x_size ()

  Get the horizontal number of tiles.
- static constexpr int y_size ()

  Get the vertical number of tiles.
- static vec< int, 2 > display_size ()

  Compute the size of the graphics representation of the tile.

Private Types

- using dti = detail::tile_infrastructure< Geography >

  The type encapsulating the implementation.
- using dti = detail::tile_infrastructure< Geography >

  The type encapsulating the implementation.

Additional Inherited Members

9.1.2.20.1 Member Typedef Documentation

9.1.2.20.1.1 axi_ss_geo [1/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::axi_ss_geo =
typename dti::axi_ss_geo
```

Definition at line 53 of file tile_infrastructure.hpp.

9.1.2.20.1.2 axi_ss_geo [2/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::axi_ss_geo =
typename dti::axi_ss_geo
```

Definition at line 53 of file tile_infrastructure.hpp.

9.1.2.20.1.3 axi_ss_t [1/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::axi_ss_t = typename
dti::axi_ss_t
```

Definition at line 56 of file tile_infrastructure.hpp.

9.1.2.20.1.4 axi_ss_t [2/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::axi_ss_t = typename
dti::axi_ss_t
```

Definition at line 56 of file tile_infrastructure.hpp.

9.1.2.20.1.5 dti [1/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::dti = detail::tile_infrastructure
[private]
```

The type encapsulating the implementation.

Definition at line 46 of file tile_infrastructure.hpp.

9.1.2.20.1.6 dti [2/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::dti = detail::tile_infrastructure
[private]
```

The type encapsulating the implementation.

Definition at line 46 of file tile_infrastructure.hpp.

9.1.2.20.1.7 facade_t [1/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::facade_t = facade::device<tile_in
dti>
```

The façade used to implement part of the user-facing type.

Definition at line 50 of file tile_infrastructure.hpp.

9.1.2.20.1.8 facade_t [2/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::facade_t = facade::device<tile_in
dti>
```

The façade used to implement part of the user-facing type.

Definition at line 50 of file tile_infrastructure.hpp.

9.1.2.20.1.9 geo [1/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::geo = Geography
```

Definition at line 52 of file tile_infrastructure.hpp.

9.1.2.20.1.10 geo [2/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::geo = Geography
```

Definition at line 52 of file tile_infrastructure.hpp.

9.1.2.20.1.11 mpl [1/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::mpl = typename
dti::mpl
```

Definition at line 54 of file tile_infrastructure.hpp.

9.1.2.20.1.12 mpl [2/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::mpl = typename
dti::mpl
```

Definition at line 54 of file tile_infrastructure.hpp.

9.1.2.20.1.13 spl [1/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::spl = typename
dti::spl
```

Definition at line 55 of file tile_infrastructure.hpp.

9.1.2.20.1.14 spl [2/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::spl = typename
dti::spl
```

Definition at line 55 of file tile_infrastructure.hpp.

9.1.2.20.2 Constructor & Destructor Documentation

9.1.2.20.2.1 tile_infrastructure() [1/4] template<typename Geography >

```
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::tile_infrastructure (
            int x,
            int y,
            auto & dev,
            ::trisycl::detail::fiber_pool & fiber_executor )  [inline]
```

Start the tile infrastructure associated to the AIE device.

Parameters

in	*x*	is the horizontal coordinate for this tile
in	*y*	is the vertical coordinate for this tile
in	*dev*	is the aie::detail::device used to control hardware when using real hardware and provide some debug information from inside the tile_infrastructure. Use auto concept here to avoid explicit type causing circular dependency
in	*fiber_executor*	is the executor used to run infrastructure details

Definition at line 75 of file tile_infrastructure.hpp.

```
00077       :  facade_t { std::make_shared<dti>(x, y, dev, fiber_executor) } {}
```

9.1.2.20.2.2 tile_infrastructure() [2/4] template<typename Geography >

```
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::tile_infrastructure ( )
[default]
```

9.1.2.20.2.3 tile_infrastructure() [3/4] template<typename Geography >

```
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::tile_infrastructure (
            int x,
            int y,
            auto & dev,
            ::trisycl::detail::fiber_pool & fiber_executor )  [inline]
```

Start the tile infrastructure associated to the AIE device.

Parameters

in	*x*	is the horizontal coordinate for this tile
in	*y*	is the vertical coordinate for this tile
in	*dev*	is the aie::detail::device used to control hardware when using real hardware and provide some debug information from inside the tile_infrastructure. Use auto concept here to avoid explicit type causing circular dependency
in	*fiber_executor*	is the executor used to run infrastructure details

Definition at line 75 of file tile_infrastructure.hpp.

```
00077       :  facade_t { std::make_shared<dti>(x, y, dev, fiber_executor) } {}
```

9.1.2.20.2.4 tile_infrastructure() [4/4] template<typename Geography >

trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::tile_infrastructure ( ) [default]

9.1.2.20.3 Member Function Documentation

9.1.2.20.3.1 connect() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::connect (
            typename geo::core_axi_stream_switch::slave_port_layout sp,
            typename geo::core_axi_stream_switch::master_port_layout mp ) [inline]
```

Configure a connection of the core tile AXI stream switch.

Definition at line 198 of file tile_infrastructure.hpp.

```
00199                                                                    {
00200     implementation->connect(sp, mp);
00201     // To allow chaining commands
00202     return *this;
00203   }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.2 connect() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::connect (
            typename geo::core_axi_stream_switch::slave_port_layout sp,
            typename geo::core_axi_stream_switch::master_port_layout mp ) [inline]
```

Configure a connection of the core tile AXI stream switch.

Definition at line 198 of file tile_infrastructure.hpp.

```
00199                                                                    {
00200     implementation->connect(sp, mp);
00201     // To allow chaining commands
00202     return *this;
00203   }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.3 display() [1/2] template<typename Geography >

```
void trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::display (
            latex::context & c ) const  [inline]
```

Display the tile to a LaTeX context.

Definition at line 209 of file tile_infrastructure.hpp.

```
00209 { implementation->display(c); }
```

References c, and trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geograp

9.1.2.20.3.4 display() [2/2] template<typename Geography >

```
void trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::display (
            latex::context & c ) const  [inline]
```

Display the tile to a LaTeX context.

Definition at line 209 of file tile_infrastructure.hpp.

```
00209 { implementation->display(c); }
```

References c, and trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geograp

9.1.2.20.3.5 display_size() [1/2] template<typename Geography >

```
static vec< int, 2 > trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >↩
::display_size ( )  [inline], [static]
```

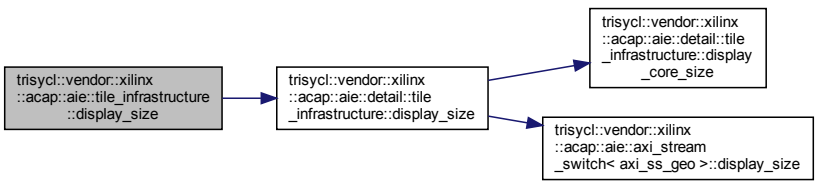
Compute the size of the graphics representation of the tile.

Definition at line 206 of file tile_infrastructure.hpp.

```
00206 { return dti::display_size(); }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display_size().

Here is the call graph for this function:



9.1.2.20.3.6 display_size() [2/2] template<typename Geography >

```
static vec< int, 2 > trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >↩
::display_size ( )  [inline], [static]
```

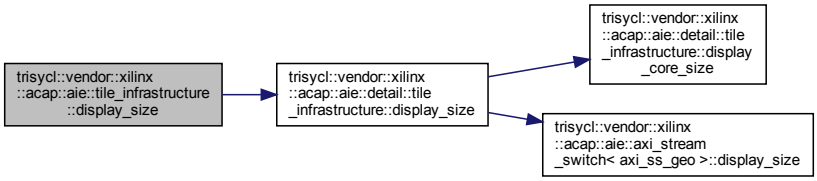
Compute the size of the graphics representation of the tile.

Definition at line 206 of file tile_infrastructure.hpp.

```
00206 { return dti::display_size(); }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display_size().

Here is the call graph for this function:



9.1.2.20.3.7 in() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::in (
            int port )  [inline]
```

Get the user input port from the AXI stream switch.

Parameters

in	*port*	is the port to use

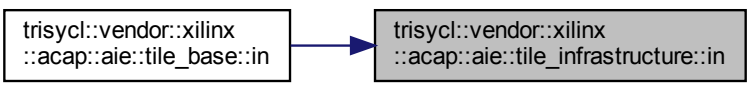
Definition at line 135 of file tile_infrastructure.hpp.

```
00135 { return implementation->in(port); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

Referenced by trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::in().

Here is the caller graph for this function:



9.1.2.20.3.8 in() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::in (
            int port )  [inline]
```

Get the user input port from the AXI stream switch.

Parameters

in	*port*	is the port to use

Definition at line 135 of file tile_infrastructure.hpp.

```
00135 { return implementation->in(port); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.9 in_connection() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::in_connection (
            int port )  [inline]
```

Get the user input connection from the AXI stream switch.

Parameters

in	*port*	is the port to use

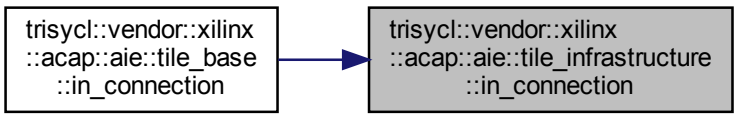
Definition at line 121 of file tile_infrastructure.hpp.

```
00121 { return implementation->in_connection(port); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

Referenced by trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::in_connection().

Here is the caller graph for this function:



9.1.2.20.3.10 in_connection() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::in_connection (
            int port )  [inline]
```

Get the user input connection from the AXI stream switch.

Parameters

in	*port*	is the port to use

Definition at line 121 of file tile_infrastructure.hpp.

```
00121 { return implementation->in_connection(port); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.11 input() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::input (
            spl p )  [inline]
```

Get the input router port of the AXI stream switch.

Parameters

p	is the slave_port_layout for the stream

Definition at line 157 of file tile_infrastructure.hpp.

```
00157 { return implementation->input(p); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > > and p.

9.1.2.20.3.12 input() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::input (
            spl p )  [inline]
```

Get the input router port of the AXI stream switch.

Parameters

p	is the slave_port_layout for the stream

Definition at line 157 of file tile_infrastructure.hpp.

```
00157 { return implementation->input(p); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > > and p.

9.1.2.20.3.13 mem() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::mem ( )  [inline]
```

Access to the common infrastructure part of a tile memory.

Definition at line 96 of file tile_infrastructure.hpp.

```
00096               {
00097     return implementation->mem();
00098   }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.14 mem() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::mem ( )  [inline]
```

Access to the common infrastructure part of a tile memory.

Definition at line 96 of file tile_infrastructure.hpp.

```
00096               {
00097     return implementation->mem();
00098   }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.15 out() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::out (
            int port )  [inline]
```

Get the user output port to the AXI stream switch.

Parameters

in	*port*	is the port to use

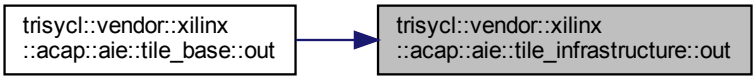
Definition at line 141 of file tile_infrastructure.hpp.

```
00141 { return implementation->out(port); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

Referenced by trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::out().

Here is the caller graph for this function:



9.1.2.20.3.16 out() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::out (
            int port )  [inline]
```

Get the user output port to the AXI stream switch.

Parameters

in	*port*	is the port to use

Definition at line 141 of file tile_infrastructure.hpp.

```
00141 { return implementation->out(port); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.17 out_connection() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::out_connection (
            int port )  [inline]
```

Get the user output connection to the AXI stream switch.

Parameters

in	*port*	is port to use

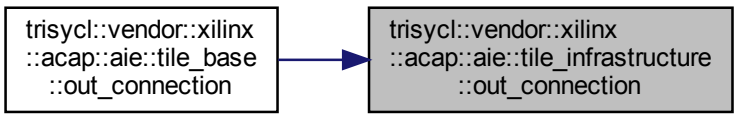
Definition at line 127 of file tile_infrastructure.hpp.

```
00127                                    {
00128     return implementation->out_connection(port);
00129   }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

Referenced by trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::out_connection().

Here is the caller graph for this function:



9.1.2.20.3.18 out_connection() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::out_connection (
            int port )  [inline]
```

Get the user output connection to the AXI stream switch.

Parameters

in	*port*	is port to use

Definition at line 127 of file tile_infrastructure.hpp.

```
00127                                   {
00128     return implementation->out_connection(port);
00129   }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.19 output() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::output (
            mpl p )  [inline]
```

Get the output router port of the AXI stream switch.

Parameters

p	is the master_port_layout for the stream

Definition at line 163 of file tile_infrastructure.hpp.

```
00163 { return implementation->output(p); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > > and p.

9.1.2.20.3.20 output() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::output (
            mpl p )  [inline]
```

Get the output router port of the AXI stream switch.

Parameters

p	is the master_port_layout for the stream

Definition at line 163 of file tile_infrastructure.hpp.

```
00163 { return implementation->output(p); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > > and p.

9.1.2.20.3.21 rx_dma() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::rx_dma (
            int id )  [inline]
```

Get access to a receiver DMA.

Parameters

in	*id*	specifies which DMA to access

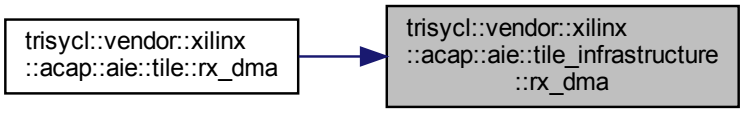
Definition at line 146 of file tile_infrastructure.hpp.

```
00146 { return implementation->rx_dma(id); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::rx_dma().

Here is the caller graph for this function:



9.1.2.20.3.22 rx_dma() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::rx_dma (
            int id )  [inline]
```

Get access to a receiver DMA.

Parameters

in	*id*	specifies which DMA to access

Definition at line 146 of file tile_infrastructure.hpp.

```
00146 { return implementation->rx_dma(id); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.23 single_task() [1/2] template<typename Geography >

```
template<typename Work >
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::single_task (
            Work && f )  [inline]
```

Launch a callable on this tile.

Recycle the tile_infrastructure shared_ptr as a fake tile handle which is copyable.

Todo In a device implementation we should have a real tile_handler type making sense on the device instead of just ∗this

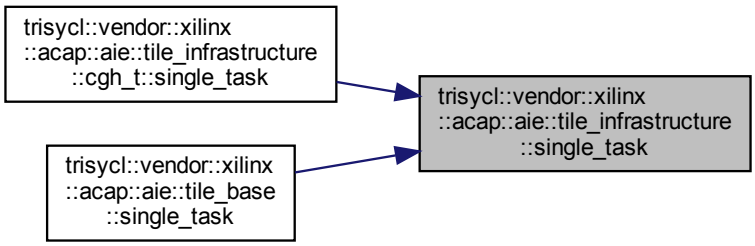
Definition at line 166 of file tile_infrastructure.hpp.

```
00166                                                    {
00167     /* Add an immediate lambda call to avoid a warning about capturing
00168 this when non using it */
00169     auto kernel = [&] {
00170       if constexpr (requires { f(); })
00171         /* If the invocable is not interested by the handler, do not
00172 provide it.  Use a copy of the callable to make the
00173 separation from host to device memory very clear.  */
00174         return [work = f] () mutable { return work(); };
00175       else
00176 /** Recycle the tile_infrastructure shared_ptr as a fake tile
00177 handle which is copyable.
00178
00179 \todo In a device implementation we should have a real
00180 tile_handler type making sense on the device instead of
00181 just *this */
00182         return [this, work = f] () mutable { return work(*this); };
00183     }();
00184
00185     implementation->single_task(kernel);
00186     // To allow chaining commands
00187     return *this;
00188   }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

Referenced by trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t::single_task(), and trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::single_task().

Here is the caller graph for this function:



9.1.2.20.3.24 single_task() [2/2]

```
template<typename Geography >
template<typename Work >
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::single_task (
            Work && f )  [inline]
```

Launch a callable on this tile.

Recycle the tile_infrastructure shared_ptr as a fake tile handle which is copyable.

Todo In a device implementation we should have a real tile_handler type making sense on the device instead of just ∗this

Definition at line 166 of file tile_infrastructure.hpp.

```
00166                                                                    {
00167     /* Add an immediate lambda call to avoid a warning about capturing
00168 this when non using it */
00169     auto kernel = [&] {
00170       if constexpr (requires { f(); })
00171         /* If the invocable is not interested by the handler, do not
00172 provide it.  Use a copy of the callable to make the
00173 separation from host to device memory very clear.  */
00174         return [work = f] () mutable { return work(); };
00175       else
00176 /** Recycle the tile_infrastructure shared_ptr as a fake tile
00177 handle which is copyable.
00178
00179 \todo In a device implementation we should have a real
00180 tile_handler type making sense on the device instead of
00181 just *this */
00182         return [this, work = f] () mutable { return work(*this); };
00183     }();
00184
00185     implementation->single_task(kernel);
00186     // To allow chaining commands
00187     return *this;
00188   }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.25 submit() [1/2] template<typename Geography >

```
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::submit (
              Invocable && f ) const  [inline]
```

Submit a command group to the tile.

Definition at line 112 of file tile_infrastructure.hpp.

```
00112                                    {
00113     cgh_t cgh { *this };
00114     std::forward<Invocable>(f)(cgh);
00115   }
```

9.1.2.20.3.26 submit() [2/2] template<typename Geography >

```
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::submit (
              Invocable && f ) const  [inline]
```

Submit a command group to the tile.

Definition at line 112 of file tile_infrastructure.hpp.

```
00112                                    {
00113     cgh_t cgh { *this };
00114     std::forward<Invocable>(f)(cgh);
00115   }
```

9.1.2.20.3.27 tx_dma() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::tx_dma (
              int id )  [inline]
```

Get access to a transmit DMA.

Parameters

in	*id*	specifies which DMA to access

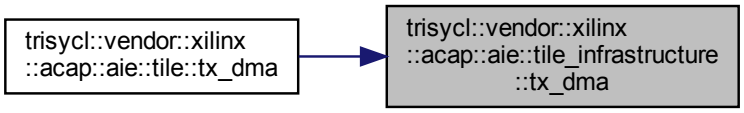
Definition at line 151 of file tile_infrastructure.hpp.

```
00151 { return implementation->tx_dma(id); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

Referenced by trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::tx_dma().

Here is the caller graph for this function:



9.1.2.20.3.28 tx_dma() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::tx_dma (
            int id )  [inline]
```

Get access to a transmit DMA.

Parameters

in	*id*	specifies which DMA to access

Definition at line 151 of file tile_infrastructure.hpp.

```
00151 { return implementation->tx_dma(id); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.29 wait() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::wait ( )  [inline]
```

Wait for the execution of the callable on this tile.

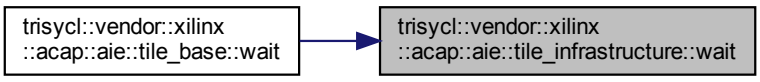
Definition at line 191 of file tile_infrastructure.hpp.

```
00191                {
00192     implementation->wait();
00193     // To allow chaining commands
00194     return *this;
00195   }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

Referenced by trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::wait().

Here is the caller graph for this function:



9.1.2.20.3.30 wait() [2/2] template<typename Geography >

auto & trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::wait ( ) [inline]

Wait for the execution of the callable on this tile.

Definition at line 191 of file tile_infrastructure.hpp.

```
00191                {
00192     implementation->wait();
00193     // To allow chaining commands
00194     return *this;
00195   }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.31 x() [1/2] template<typename Geography >

int trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::x ( ) [inline]

Get the horizontal coordinate.

Definition at line 84 of file tile_infrastructure.hpp.

```
00084 { return implementation->x(); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.32 x() [2/2] template<typename Geography >

int trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::x ( ) [inline]

Get the horizontal coordinate.

Definition at line 84 of file tile_infrastructure.hpp.

```
00084 { return implementation->x(); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.33 x_size() [1/2] template<typename Geography >

```
static constexpr int trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::x↩
_size ( )  [inline], [static], [constexpr]
```

Get the horizontal number of tiles.

Definition at line 90 of file tile_infrastructure.hpp.

```
00090 { return geo::x_size; }
```

9.1.2.20.3.34 x_size() [2/2] template<typename Geography >

```
static constexpr int trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::x↩
_size ( )  [inline], [static], [constexpr]
```

Get the horizontal number of tiles.

Definition at line 90 of file tile_infrastructure.hpp.

```
00090 { return geo::x_size; }
```

9.1.2.20.3.35 y() [1/2] template<typename Geography >

```
int trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::y ( )  [inline]
```

Get the vertical coordinate.

Definition at line 87 of file tile_infrastructure.hpp.

```
00087 { return implementation->y(); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.36 y() [2/2] template<typename Geography >

```
int trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::y ( )  [inline]
```

Get the vertical coordinate.

Definition at line 87 of file tile_infrastructure.hpp.

```
00087 { return implementation->y(); }
```

References trisycl::detail::shared_ptr_implementation< tile_infrastructure< Geography >, detail::tile_infrastructure< Geography > >

9.1.2.20.3.37 y_size() [1/2] template<typename Geography >

```
static constexpr int trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::y↩
_size ( )  [inline], [static], [constexpr]
```

Get the vertical number of tiles.

Definition at line 93 of file tile_infrastructure.hpp.

```
00093 { return geo::y_size; }
```

9.1.2.20.3.38 y_size() [2/2] template<typename Geography >

```
static constexpr int trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::y↩
_size ( )  [inline], [static], [constexpr]
```

Get the vertical number of tiles.

Definition at line 93 of file tile_infrastructure.hpp.

```
00093 { return geo::y_size; }
```

9.1.2.21 class trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure

template<typename Geography>
class trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >

The AI Engine tile infrastructure common to all the tiles.

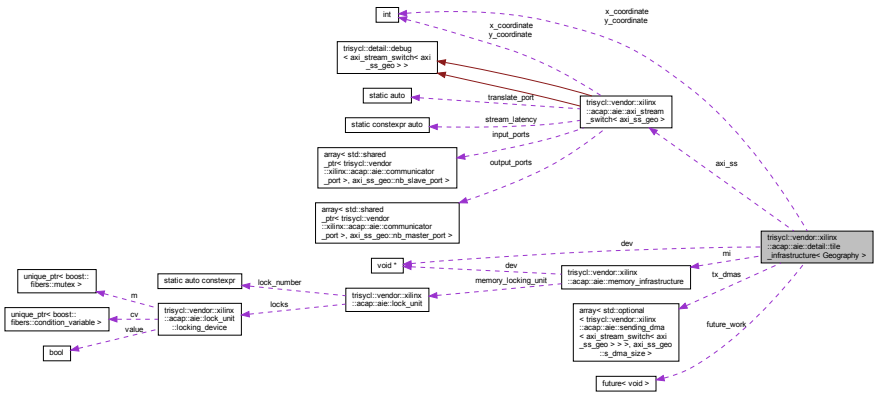
This allows some type erasure while accessing the common tile infrastructure.

Parameters

AIE	is the type representing the full CGRA with the programs and memory contents

Definition at line 51 of file tile_infrastructure.hpp.

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >:



Public Types

- using geo = Geography
- using axi_ss_geo = typename geo::core_axi_stream_switch
- using mpl = typename axi_ss_geo::master_port_layout
- using spl = typename axi_ss_geo::slave_port_layout
- using axi_ss_t = axi_stream_switch< axi_ss_geo >
- using geo = Geography
- using axi_ss_geo = typename geo::core_axi_stream_switch
- using mpl = typename axi_ss_geo::master_port_layout
- using spl = typename axi_ss_geo::slave_port_layout
- using axi_ss_t = axi_stream_switch< axi_ss_geo >

Public Member Functions

- tile_infrastructure (int x, int y, auto &dev, ::trisycl::detail::fiber_pool &fiber_executor)

  Start the tile infrastructure associated to the AIE device tile.
- int x ()

  Get the horizontal coordinate.
- int y ()

  Get the vertical coordinate.
- auto & mem ()

  Access to the common infrastructure part of a tile memory.
- auto & in_connection (int port)

  Get the user input connection from the AXI stream switch.
- auto & out_connection (int port)

  Get the user output connection to the AXI stream switch.
- auto & in (int port)

  Get the user input port from the AXI stream switch.
- auto & out (int port)

  Get the user output port to the AXI stream switch.
- auto & rx_dma (int id)

  Get access to a receiver DMA.
- auto & tx_dma (int id)

  Get access to a transmit DMA.
- auto & input (spl p)

  Get the input router port of the AXI stream switch.
- auto & output (mpl p)

  Get the output router port of the AXI stream switch.
- template<typename Work >
  void single_task (Work &&f)

  Launch an invocable on this tile.
- void wait ()

  Wait for the execution of the callable on this tile.
- void connect (typename geo::core_axi_stream_switch::slave_port_layout sp, typename geo::core_axi_↩ stream_switch::master_port_layout mp)

  Configure a connection of the core tile AXI stream switch.
- void display (latex::context &c) const

  Display the tile to a LaTeX context.
- tile_infrastructure (int x, int y, auto &dev, ::trisycl::detail::fiber_pool &fiber_executor)

  Start the tile infrastructure associated to the AIE device tile.
- int x ()

  Get the horizontal coordinate.
- int y ()

  Get the vertical coordinate.
- auto & mem ()

  Access to the common infrastructure part of a tile memory.
- auto & in_connection (int port)

  Get the user input connection from the AXI stream switch.
- auto & out_connection (int port)

  Get the user output connection to the AXI stream switch.
- auto & in (int port)

  Get the user input port from the AXI stream switch.
- auto & out (int port)

Get the user output port to the AXI stream switch.

- auto & rx_dma (int id)

  Get access to a receiver DMA.
- auto & tx_dma (int id)

  Get access to a transmit DMA.
- auto & input (spl p)

  Get the input router port of the AXI stream switch.
- auto & output (mpl p)

  Get the output router port of the AXI stream switch.
- template<typename Work >
  void single_task (Work &&f)

  Launch an invocable on this tile.
- void wait ()

  Wait for the execution of the callable on this tile.
- void connect (typename geo::core_axi_stream_switch::slave_port_layout sp, typename geo::core_axi_↩ stream_switch::master_port_layout mp)

  Configure a connection of the core tile AXI stream switch.
- void display (latex::context &c) const

  Display the tile to a LaTeX context.

Static Public Member Functions

- static vec< int, 2 > display_core_size ()

  Compute the size of the graphics representation of the processor.
- static vec< int, 2 > display_size ()

  Compute the size of the graphics representation of the tile.
- static vec< int, 2 > display_core_size ()

  Compute the size of the graphics representation of the processor.
- static vec< int, 2 > display_size ()

  Compute the size of the graphics representation of the tile.

Static Private Member Functions

- static auto translate_input_port (int port)

  Map the user input port number to the AXI stream switch port.
- static auto translate_output_port (int port)

  Map the user output port number to the AXI stream switch port.
- static auto translate_input_port (int port)

  Map the user input port number to the AXI stream switch port.
- static auto translate_output_port (int port)

  Map the user output port number to the AXI stream switch port.

Private Attributes

- int x_coordinate

  Keep the horizontal coordinate.
- int y_coordinate

  Keep the vertical coordinate.
- void * dev

  Keep track of the aie::detail::device for hardware resource control in device mode or for debugging purpose for better messages.
- axi_ss_t axi_ss

  The AXI stream switch of the core tile.
- aie::memory_infrastructure mi

  Keep track of all the infrastructure tile memories of this device.
- std::array< std::optional< sending_dma< axi_ss_t > >, axi_ss_geo::s_dma_size > tx_dmas

  Sending DMAs.
- std::future< void > future_work

  Keep track of the std::thread execution in this tile.

9.1.2.21.1 Member Typedef Documentation

9.1.2.21.1.1 axi_ss_geo [1/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss_geo
= typename geo::core_axi_stream_switch
```

Definition at line 55 of file tile_infrastructure.hpp.

9.1.2.21.1.2 axi_ss_geo [2/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss_geo
= typename geo::core_axi_stream_switch
```

Definition at line 55 of file tile_infrastructure.hpp.

9.1.2.21.1.3 axi_ss_t [1/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss_t =
axi_stream_switch<axi_ss_geo>
```

Definition at line 58 of file tile_infrastructure.hpp.

9.1.2.21.1.4 axi_ss_t [2/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss_t =
axi_stream_switch<axi_ss_geo>
```

Definition at line 58 of file tile_infrastructure.hpp.

9.1.2.21.1.5 geo [1/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::geo =
Geography
```

Definition at line 54 of file tile_infrastructure.hpp.

9.1.2.21.1.6 geo [2/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::geo =
Geography
```

Definition at line 54 of file tile_infrastructure.hpp.

9.1.2.21.1.7 mpl [1/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::mpl =
typename axi_ss_geo::master_port_layout
```

Definition at line 56 of file tile_infrastructure.hpp.

9.1.2.21.1.8 mpl [2/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::mpl =
typename axi_ss_geo::master_port_layout
```

Definition at line 56 of file tile_infrastructure.hpp.

9.1.2.21.1.9 spl [1/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::spl =
typename axi_ss_geo::slave_port_layout
```

Definition at line 57 of file tile_infrastructure.hpp.

9.1.2.21.1.10 spl [2/2] template<typename Geography >

```
using trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::spl =
typename axi_ss_geo::slave_port_layout
```

Definition at line 57 of file tile_infrastructure.hpp.

9.1.2.21.2 Constructor & Destructor Documentation

9.1.2.21.2.1 tile_infrastructure() [1/2] template<typename Geography >

```
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::tile_infrastructure
(
            int x,
            int y,
            auto & dev,
            ::trisycl::detail::fiber_pool & fiber_executor )  [inline]
```

Start the tile infrastructure associated to the AIE device tile.

Parameters

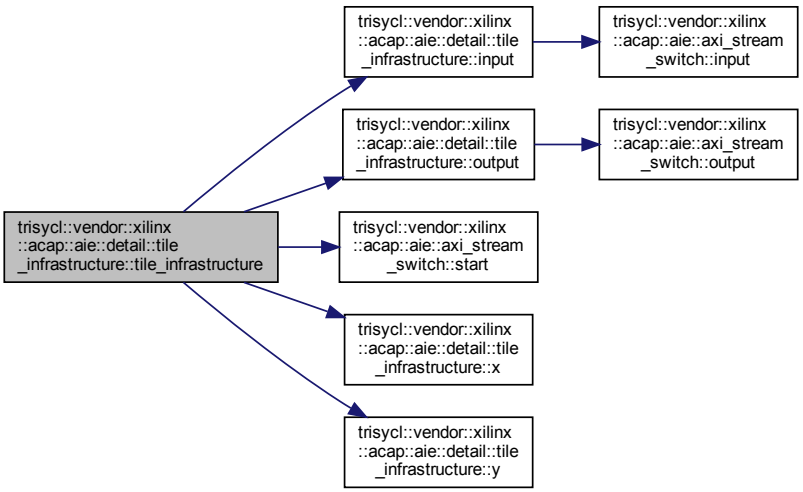
in	*x*	is the horizontal coordinate for this tile
in	*y*	is the vertical coordinate for this tile
in	*dev*	is the aie::detail::device used to control hardware when using real hardware and provide some debug information from inside the tile_infrastructure. Use auto concept here to avoid explicit type causing circular dependency
in	*fiber_executor*	is the executor used to run infrastructure details

Definition at line 132 of file tile_infrastructure.hpp.

```
00134       :  x_coordinate { x }
00135       , y_coordinate { y }
00136       , dev { &dev }
00137       , mi { dev }
00138 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00139       , fe { &fiber_executor }
00140 #endif
00141   {
00142     // Connect the core receivers to its AXI stream switch
00143     for (auto p :  views::enum_type(mpl::me_0, mpl::me_last))
00144       output(p) =
00145           std::make_shared<port_receiver<axi_ss_t»(axi_ss, "core_receiver");
00146     axi_ss.start(x, y, fiber_executor);
00147     /* Create the core tile receiver DMAs and make them directly the
00148 switch output ports */
00149     for (auto p :  axi_ss_geo::m_dma_range)
00150       output(p) =
00151           std::make_shared<receiving_dma<axi_ss_t»(axi_ss, fiber_executor);
00152     /* Create the core tile sender DMAs and connect them internally to
00153 their switch input ports */
00154     for (const auto& [d, p] :
00155          ranges::views::zip(tx_dmas, axi_ss_geo::s_dma_range))
00156       d.emplace(fiber_executor, input(p));
00157   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, trisycl::views::enum_type, trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::input(), trisycl::vendor::xilinx::acap::aie::detail::tile_infrastruc p, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::start(), trisycl::vendor::xilinx::acap::aie::detail::tile_i trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::x(), and trisycl::vendor::xilinx::acap::aie::detail::tile_infrastru

Here is the call graph for this function:



9.1.2.21.2.2 tile_infrastructure() [2/2] template<typename Geography >

```
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::tile_infrastructure
(
            int x,
            int y,
            auto & dev,
            ::trisycl::detail::fiber_pool & fiber_executor )  [inline]
```

Start the tile infrastructure associated to the AIE device tile.

Parameters

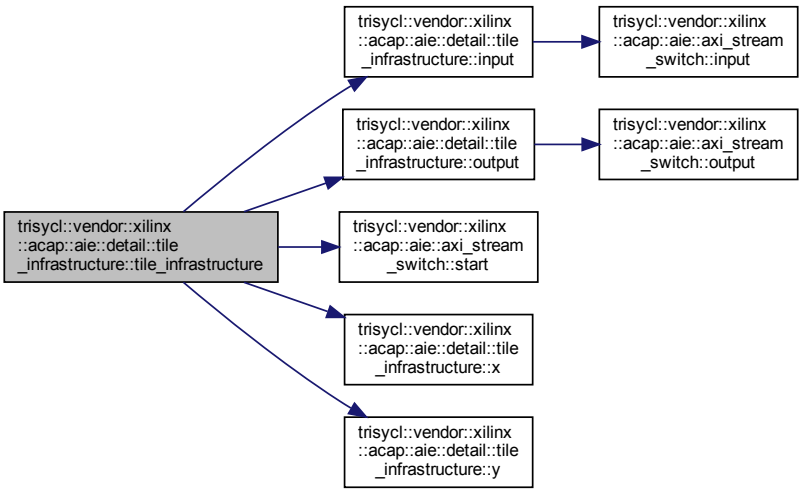
in	*x*	is the horizontal coordinate for this tile
in	*y*	is the vertical coordinate for this tile
in	*dev*	is the aie::detail::device used to control hardware when using real hardware and provide some debug information from inside the tile_infrastructure. Use auto concept here to avoid explicit type causing circular dependency
in	*fiber_executor*	is the executor used to run infrastructure details

Definition at line 132 of file tile_infrastructure.hpp.

```
00134       :  x_coordinate { x }
00135       , y_coordinate { y }
00136       , dev { &dev }
00137       , mi { dev }
00138 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00139       , fe { &fiber_executor }
00140 #endif
00141   {
00142     // Connect the core receivers to its AXI stream switch
00143     for (auto p :  views::enum_type(mpl::me_0, mpl::me_last))
00144       output(p) =
00145           std::make_shared<port_receiver<axi_ss_t»(axi_ss, "core_receiver");
00146     axi_ss.start(x, y, fiber_executor);
00147     /* Create the core tile receiver DMAs and make them directly the
00148 switch output ports */
00149     for (auto p :  axi_ss_geo::m_dma_range)
00150       output(p) =
00151           std::make_shared<receiving_dma<axi_ss_t»(axi_ss, fiber_executor);
00152     /* Create the core tile sender DMAs and connect them internally to
00153 their switch input ports */
00154     for (const auto& [d, p] :
00155          ranges::views::zip(tx_dmas, axi_ss_geo::s_dma_range))
00156       d.emplace(fiber_executor, input(p));
00157   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, trisycl::views::enum_type, trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::input(), trisycl::vendor::xilinx::acap::aie::detail::tile_infrastruc
p, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::start(), trisycl::vendor::xilinx::acap::aie::detail::tile_i
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::x(), and trisycl::vendor::xilinx::acap::aie::detail::tile_infrastru

Here is the call graph for this function:



9.1.2.21.3 Member Function Documentation

9.1.2.21.3.1 connect() [1/2] template<typename Geography >

```
void trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::connect (
            typename geo::core_axi_stream_switch::slave_port_layout sp,
            typename geo::core_axi_stream_switch::master_port_layout mp )  [inline]
```

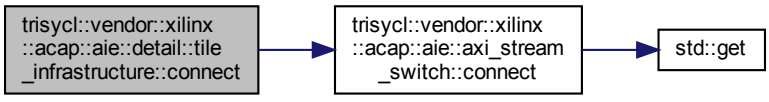
Configure a connection of the core tile AXI stream switch.

Definition at line 262 of file tile_infrastructure.hpp.

```
00263                                                                    {
00264     axi_ss.connect(sp, mp);
00265   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, and trisycl::vendor::xilinx::acap::aie::axi_

Here is the call graph for this function:



9.1.2.21.3.2 connect() [2/2] template<typename Geography >

```
void trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::connect (
            typename geo::core_axi_stream_switch::slave_port_layout sp,
            typename geo::core_axi_stream_switch::master_port_layout mp )  [inline]
```

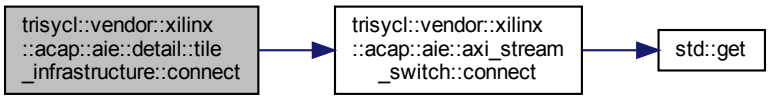
Configure a connection of the core tile AXI stream switch.

Definition at line 262 of file tile_infrastructure.hpp.

```
00263                                                                    {
00264     axi_ss.connect(sp, mp);
00265   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, and trisycl::vendor::xilinx::acap::aie::axi_

Here is the call graph for this function:



9.1.2.21.3.3 display() [1/2] template<typename Geography >

```
void trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display (
            latex::context & c ) const  [inline]
```

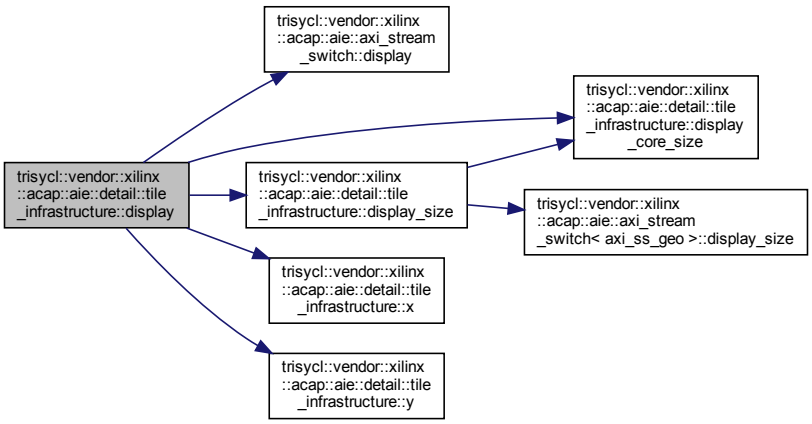
Display the tile to a LaTeX context.

Definition at line 281 of file tile_infrastructure.hpp.

```
00281                                                                    {
00282     auto get_tikz_coordinate = [&](auto x, auto y) {
00283       auto const [x_size, y_size] = display_size();
00284       return (boost::format { "(%1%,%2%)" }
00285               // Scale real LaTeX coordinate to fit
00286               % c.scale(x_coordinate * x_size + x) %
00287               c.scale(y_coordinate * y_size + y))
00288           .str();
00289     };
00290     c.add((boost::format { "  \\begin{scope}[name prefix = TileX%1%Y%2%]" } %
00291            x_coordinate % y_coordinate)
00292               .str());
00293     axi_ss.display(c, display_core_size(), get_tikz_coordinate);
00294
00295     // Connect the core receivers to its AXI stream switch
00296     for (auto [i, p] :  axi_ss_geo::m_me_range | ranges::views::enumerate) {
00297       c.add(
00298           (boost::format { R"(
00299 \coordinate(CoreIn%1%) at %2%;
00300 \node[rotate=90,anchor=east](CoreIn%1%Label) at %2% {in(%1%)};
00301 \draw (node cs:name=MMe%1%)
00302 -| (node cs:name=CoreIn%1%);)" } %
00303 i %
00304 get_tikz_coordinate(i, ranges::distance(axi_ss_geo::m_me_range) + 1))
00305 .str());
00306 };
00307 // Connect the core senders to its AXI stream switch
00308     for (auto [i, p] :  axi_ss_geo::s_me_range | ranges::views::enumerate) {
00309       c.add(
00310           (boost::format { R"(
00311 \coordinate(CoreOut%1%) at %2%;
00312 \node[anchor=east](CoreOut%1%Label) at %2%  {out(%1%)};
00313 \draw (node cs:name=CoreOut%1%)
00314 -| (node cs:name=SMe%1%);)" } %
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, c, trisycl::vendor::xilinx::acap::aie::axi_s
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display_core_size(), trisycl::vendor::xilinx::acap::aie::detail::
i, p, trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::x(), trisycl::vendor::xilinx::acap::aie::detail::tile_infrastru
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::y(), and trisycl::vendor::xilinx::acap::aie::detail::tile_infrastru

Here is the call graph for this function:



9.1.2.21.3.4 display() [2/2] template<typename Geography >

```
void trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display (
            latex::context & c ) const  [inline]
```

Display the tile to a LaTeX context.

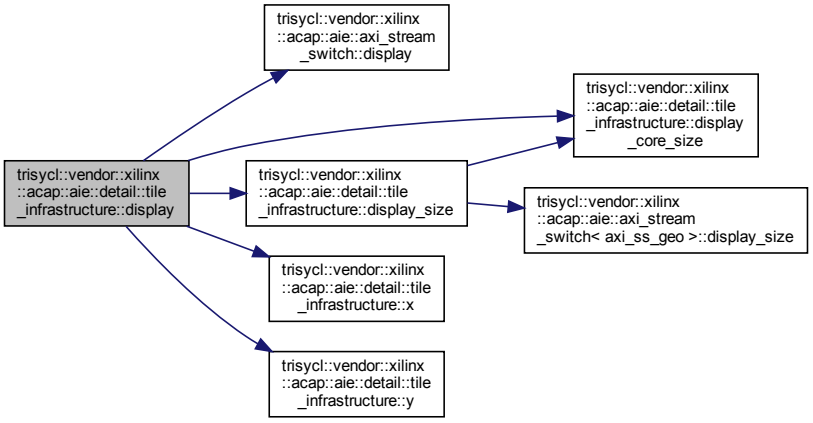
Definition at line 281 of file tile_infrastructure.hpp.

```
00281                                       {
00282     auto get_tikz_coordinate = [&](auto x, auto y) {
00283       auto const [x_size, y_size] = display_size();
00284       return (boost::format { "(%1%,%2%)" }
00285               // Scale real LaTeX coordinate to fit
00286               % c.scale(x_coordinate * x_size + x) %
00287               c.scale(y_coordinate * y_size + y))
00288           .str();
00289     };
00290     c.add((boost::format { "  \\begin{scope}[name prefix = TileX%1%Y%2%]" } %
00291            x_coordinate % y_coordinate)
00292               .str());
00293     axi_ss.display(c, display_core_size(), get_tikz_coordinate);
00294
00295     // Connect the core receivers to its AXI stream switch
00296     for (auto [i, p] :  axi_ss_geo::m_me_range | ranges::views::enumerate) {
00297       c.add(
00298           (boost::format { R"(
00299 \coordinate(CoreIn%1%) at %2%;
00300 \node[rotate=90,anchor=east](CoreIn%1%Label) at %2% {in(%1%)};
00301 \draw (node cs:name=MMe%1%)
00302 -| (node cs:name=CoreIn%1%);)" } %
00303 i %
00304 get_tikz_coordinate(i, ranges::distance(axi_ss_geo::m_me_range) + 1))
00305 .str());
00306 };
00307 // Connect the core senders to its AXI stream switch
00308     for (auto [i, p] :  axi_ss_geo::s_me_range | ranges::views::enumerate) {
00309       c.add(
00310           (boost::format { R"(
00311 \coordinate(CoreOut%1%) at %2%;
00312 \node[anchor=east](CoreOut%1%Label) at %2%  {out(%1%)};
```

```
00313 \draw (node cs:name=CoreOut%1%)
00314 -| (node cs:name=SMe%1%);)" } %
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, c, trisycl::vendor::xilinx::acap::aie::axi_s
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display_core_size(), trisycl::vendor::xilinx::acap::aie::detail::
i, p, trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::x(), trisycl::vendor::xilinx::acap::aie::detail::tile_infrastru
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::y(), and trisycl::vendor::xilinx::acap::aie::detail::tile_infrastru

Here is the call graph for this function:



9.1.2.21.3.5 display_core_size() [1/2]

```
template<typename Geography >
static vec< int, 2 > trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography
>::display_core_size ( )  [inline], [static]
```

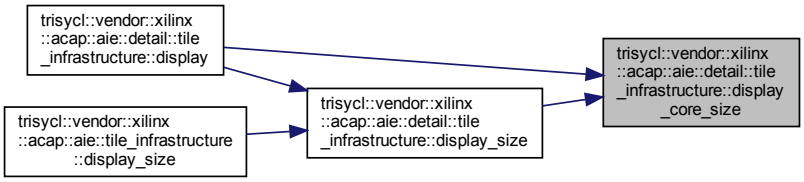
Compute the size of the graphics representation of the processor.

Definition at line 268 of file tile_infrastructure.hpp.

```
00268                                        {
00269     // This is the minimum rectangle fitting all the processor outputs & inputs
00270     return { 1 + ranges::distance(axi_ss_geo::m_me_range),
00271              1 + ranges::distance(axi_ss_geo::s_me_range) };
00272   }
```

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display(), and trisycl::vendor::xilinx::acap::aie

Here is the caller graph for this function:



9.1.2.21.3.6 display_core_size() [2/2] template<typename Geography >

static vec< int, 2 > trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display_core_size ( ) [inline], [static]

Compute the size of the graphics representation of the processor.

Definition at line 268 of file tile_infrastructure.hpp.

```
00268                                          {
00269     // This is the minimum rectangle fitting all the processor outputs & inputs
00270     return { 1 + ranges::distance(axi_ss_geo::m_me_range),
00271              1 + ranges::distance(axi_ss_geo::s_me_range) };
00272   }
```

9.1.2.21.3.7 display_size() [1/2] template<typename Geography >

static vec< int, 2 > trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display_size ( ) [inline], [static]

Compute the size of the graphics representation of the tile.

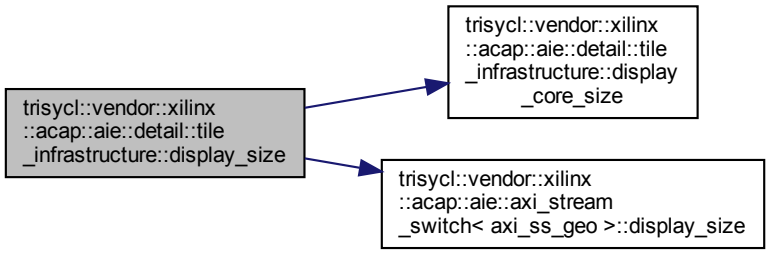
Definition at line 275 of file tile_infrastructure.hpp.

```
00275                                        {
00276     // Just the sum of the size of its content
00277     return display_core_size() + axi_ss_t::display_size();
00278   }
```

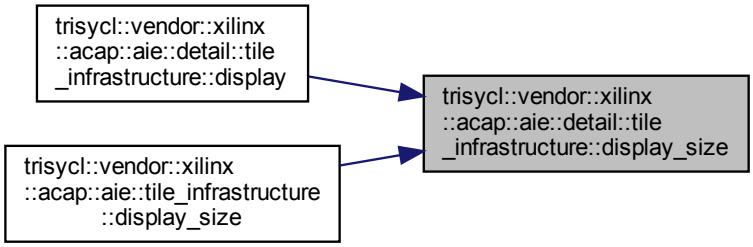
References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display_core_size(), and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< axi_ss_geo >::display_size().

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display(), and trisycl::vendor::xilinx::acap::aie

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.21.3.8 display_size() [2/2] template<typename Geography >

static vec< int, 2 > trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display_size ( ) [inline], [static]

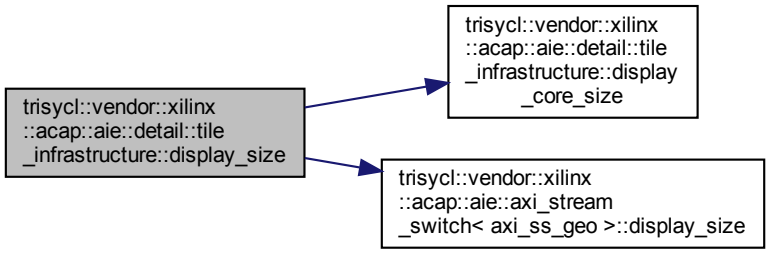
Compute the size of the graphics representation of the tile.

Definition at line 275 of file tile_infrastructure.hpp.

```
00275                                        {
00276     // Just the sum of the size of its content
00277     return display_core_size() + axi_ss_t::display_size();
00278   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display_core_size(), and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< axi_ss_geo >::display_size().

Here is the call graph for this function:



9.1.2.21.3.9 in() [1/2] template<typename Geography >

auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::in (int *port*) [inline]

Get the user input port from the AXI stream switch.

Parameters

in	*port*	is the port to use

Definition at line 194 of file tile_infrastructure.hpp.

```
00194                                  {
00195     TRISYCL_DUMP_T("in(" << port << ") on tile(" << x_coordinate << ','
00196                          << y_coordinate << ')');
00197     return *in_connection(port);
00198   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::in_connection(), TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::x_coordinate, and trisycl::vendor::xilinx::acap::aie::detail::til

Here is the call graph for this function:



9.1.2.21.3.10 in() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::in (
            int port )  [inline]
```

Get the user input port from the AXI stream switch.

Parameters

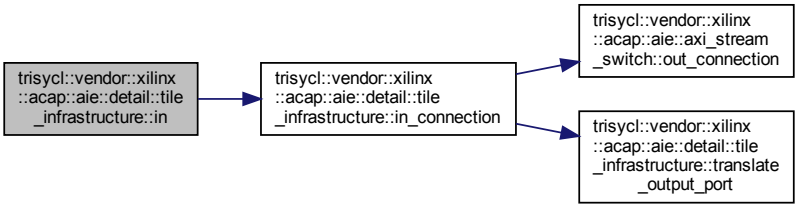
in	*port*	is the port to use

Definition at line 194 of file tile_infrastructure.hpp.

```
00194                     {
00195     TRISYCL_DUMP_T("in(" « port « ") on tile(" « x_coordinate « ','
00196                         « y_coordinate « ')');
00197     return *in_connection(port);
00198   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::in_connection(), TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::x_coordinate, and trisycl::vendor::xilinx::acap::aie::detail::til

Here is the call graph for this function:



9.1.2.21.3.11 in_connection() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::in_↩
connection (
            int port )  [inline]
```

Get the user input connection from the AXI stream switch.

Parameters

in	*port*	is the port to use

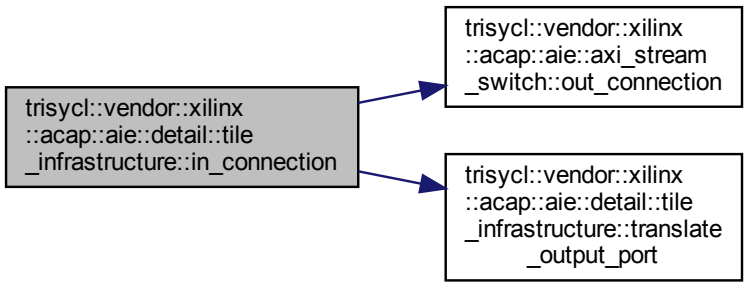
Definition at line 174 of file tile_infrastructure.hpp.

```
00174                                  {
00175     /* The input port for the core is actually the corresponding
00176 output on the switch */
00177     return axi_ss.out_connection(translate_output_port(port));
00178   }
```

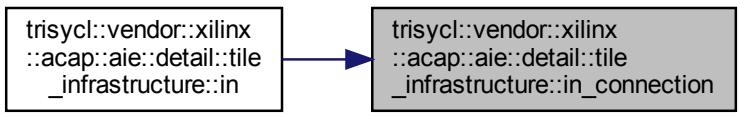
References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stre and trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::translate_output_port().

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::in().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.21.3.12 in_connection() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::in_↩
connection (
            int port )  [inline]
```

Get the user input connection from the AXI stream switch.

Parameters

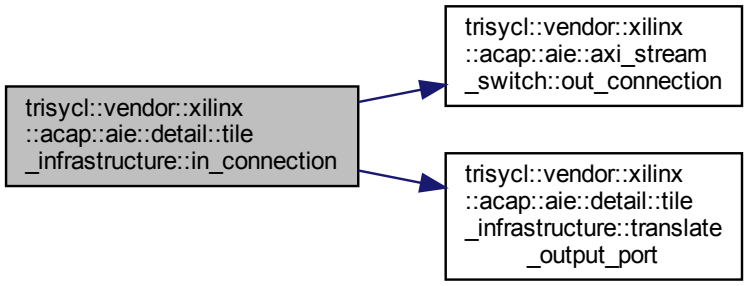
in	*port*	is the port to use

Definition at line 174 of file tile_infrastructure.hpp.

```
00174                                    {
00175     /* The input port for the core is actually the corresponding
00176 output on the switch */
00177     return axi_ss.out_connection(translate_output_port(port));
00178   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stre and trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::translate_output_port().

Here is the call graph for this function:



9.1.2.21.3.13 input() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::input (
            spl p )  [inline]
```

Get the input router port of the AXI stream switch.

Parameters

p	is the slave_port_layout for the stream

Definition at line 230 of file tile_infrastructure.hpp.

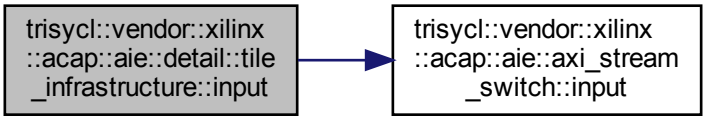
```
00230                                {
00231     // No index validation required because of type safety
00232     return axi_ss.input(p);
```

```
00233   }
```

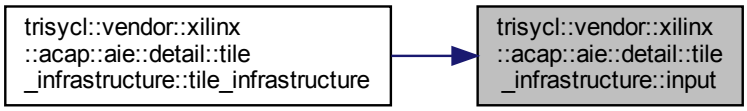
References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stre and p.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::tile_infrastructure().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.21.3.14 input() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::input (
            spl p )  [inline]
```

Get the input router port of the AXI stream switch.

Parameters

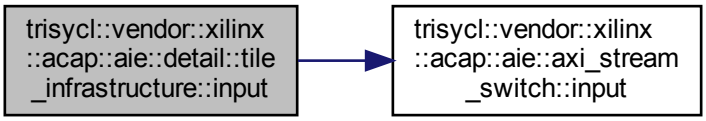
p	is the slave_port_layout for the stream

Definition at line 230 of file tile_infrastructure.hpp.

```
00230                     {
00231     // No index validation required because of type safety
00232     return axi_ss.input(p);
00233   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stre and p.

Here is the call graph for this function:



9.1.2.21.3.15 mem() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::mem ( )
[inline]
```

Access to the common infrastructure part of a tile memory.

Definition at line 166 of file tile_infrastructure.hpp.

```
00166               {
00167     return mi;
00168   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::mi.

9.1.2.21.3.16 mem() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::mem ( )
[inline]
```

Access to the common infrastructure part of a tile memory.

Definition at line 166 of file tile_infrastructure.hpp.

```
00166               {
00167     return mi;
00168   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::mi.

9.1.2.21.3.17 out() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::out (
            int port )  [inline]
```

Get the user output port to the AXI stream switch.

Parameters

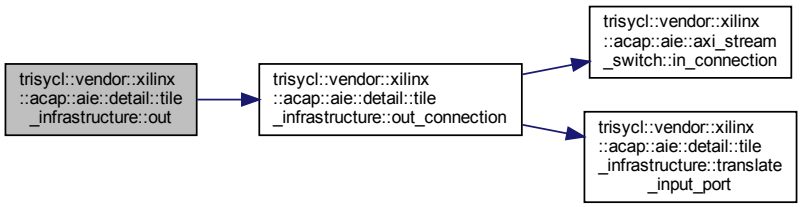
in	*port*	is the port to use

Definition at line 204 of file tile_infrastructure.hpp.

```
00204                         {
00205     TRISYCL_DUMP_T("out(" « port « ") on tile(" « x_coordinate « ','
00206                            « y_coordinate « ')');
00207     return *out_connection(port);
00208   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::out_connection(), TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::x_coordinate, and trisycl::vendor::xilinx::acap::aie::detail::til

Here is the call graph for this function:



9.1.2.21.3.18 out() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::out (
            int port )  [inline]
```

Get the user output port to the AXI stream switch.

Parameters

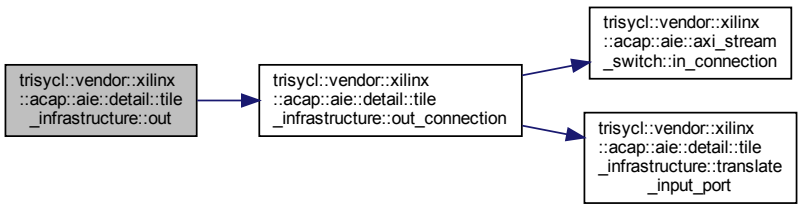
in	*port*	is the port to use

Definition at line 204 of file tile_infrastructure.hpp.

```
00204                         {
00205     TRISYCL_DUMP_T("out(" « port « ") on tile(" « x_coordinate « ','
00206                            « y_coordinate « ')');
00207     return *out_connection(port);
00208   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::out_connection(), TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::x_coordinate, and trisycl::vendor::xilinx::acap::aie::detail::til

Here is the call graph for this function:



9.1.2.21.3.19 out_connection() [1/2]

```
template<typename Geography >
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::out_↩
connection (
            int port )  [inline]
```

Get the user output connection to the AXI stream switch.

Parameters

in	*port*	is port to use

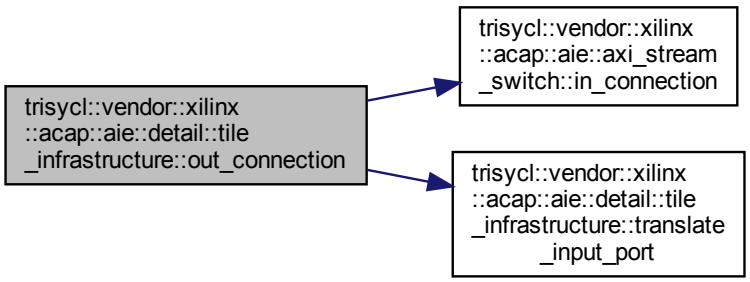
Definition at line 184 of file tile_infrastructure.hpp.

```
00184                                    {
00185     /* The output port for the core is actually the corresponding
00186 input on the switch */
00187     return axi_ss.in_connection(translate_input_port(port));
00188   }
```

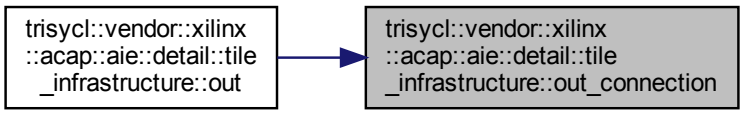
References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stre and trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::translate_input_port().

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::out().

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.21.3.20 out_connection() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::out_↩
connection (
            int port )  [inline]
```

Get the user output connection to the AXI stream switch.

Parameters

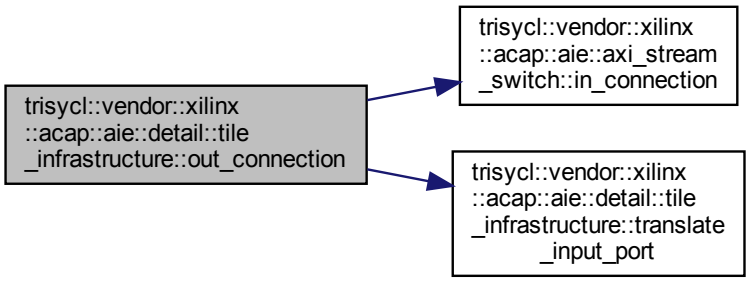
in	*port*	is port to use

Definition at line 184 of file tile_infrastructure.hpp.

```
00184                                    {
00185     /* The output port for the core is actually the corresponding
00186 input on the switch */
00187     return axi_ss.in_connection(translate_input_port(port));
00188   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stre and trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::translate_input_port().

Here is the call graph for this function:



9.1.2.21.3.21 output() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::output (
            mpl p )  [inline]
```

Get the output router port of the AXI stream switch.

Parameters

p	is the master_port_layout for the stream

Definition at line 239 of file tile_infrastructure.hpp.

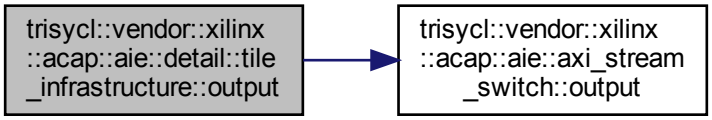
```
00239                                    {
00240     // No index validation required because of type safety
00241     return axi_ss.output(p);
```

```
00242   }
```

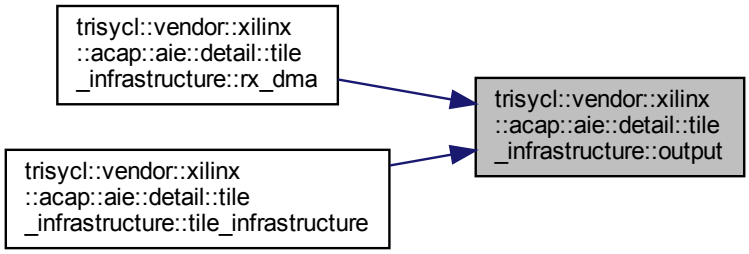
References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stre and p.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::rx_dma(), and trisycl::vendor::xilinx::acap::ai

Here is the call graph for this function:



Here is the caller graph for this function:



9.1.2.21.3.22 output() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::output (
            mpl p )  [inline]
```

Get the output router port of the AXI stream switch.

Parameters

p	is the master_port_layout for the stream

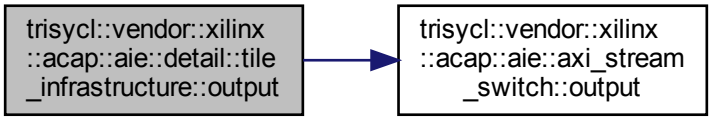
Definition at line 239 of file tile_infrastructure.hpp.

```
00239                       {
00240     // No index validation required because of type safety
```

```
00241     return axi_ss.output(p);
00242   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stre
and p.

Here is the call graph for this function:



9.1.2.21.3.23 rx_dma() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::rx_dma (
            int id )  [inline]
```

Get access to a receiver DMA.

Parameters

in	*id*	specifies which DMA to access

The output of the switch is actually a receiving DMA, so we can view it as a DMA

Definition at line 213 of file tile_infrastructure.hpp.

```
00213                       {
00214 /** The output of the switch is actually a receiving DMA, so we
00215 can view it as a DMA */
00216     return static_cast<receiving_dma<axi_ss_t>&>(*output(
00217         axi_ss_t::translate_port(id, mpl::dma_0, mpl::dma_last,
00218                                  "The receiver DMA port is out of range")));
00219   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::output(), and trisycl::vendor::xilinx::acap::aie::ax

Here is the call graph for this function:



9.1.2.21.3.24 rx_dma() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::rx_dma (
            int id )  [inline]
```

Get access to a receiver DMA.

Parameters

in	*id*	specifies which DMA to access

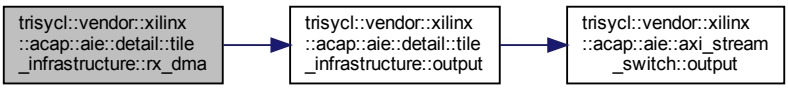
The output of the switch is actually a receiving DMA, so we can view it as a DMA

Definition at line 213 of file tile_infrastructure.hpp.

```
00213                                {
00214 /** The output of the switch is actually a receiving DMA, so we
00215 can view it as a DMA */
00216     return static_cast<receiving_dma<axi_ss_t>&>(*output(
00217         axi_ss_t::translate_port(id, mpl::dma_0, mpl::dma_last,
00218                                  "The receiver DMA port is out of range")));
00219   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::output(), and trisycl::vendor::xilinx::acap::aie::ax

Here is the call graph for this function:



9.1.2.21.3.25 single_task() [1/2] template<typename Geography >

```
template<typename Work >
void trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::single_task
(
            Work && f )  [inline]
```

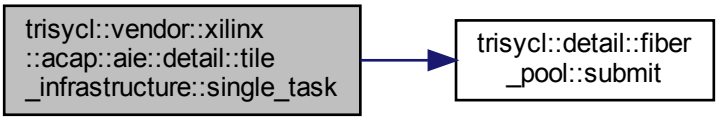
Launch an invocable on this tile.

Definition at line 245 of file tile_infrastructure.hpp.

```
00245                                                  {
00246     if (future_work.valid())
00247       throw std::logic_error("Something is already running on this tile!");
00248     // Launch the tile program immediately on a new executor engine
00249 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00250     future_work = fe->submit(std::forward<Work>(f));
00251 #else
00252     future_work = std::async(std::launch::async, std::forward<Work>(f));
00253 #endif
00254   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::future_work, and trisycl::detail::fiber_pool::subm

Here is the call graph for this function:



9.1.2.21.3.26 single_task() [2/2] template<typename Geography >

```
template<typename Work >
void trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::single_task
(
            Work && f )  [inline]
```

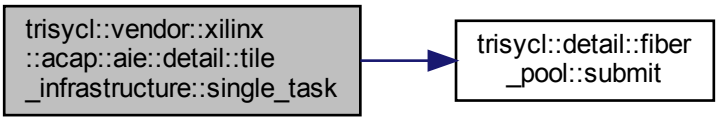
Launch an invocable on this tile.

Definition at line 245 of file tile_infrastructure.hpp.

```
00245                                                {
00246     if (future_work.valid())
00247       throw std::logic_error("Something is already running on this tile!");
00248     // Launch the tile program immediately on a new executor engine
00249 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00250     future_work = fe->submit(std::forward<Work>(f));
00251 #else
00252     future_work = std::async(std::launch::async, std::forward<Work>(f));
00253 #endif
00254   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::future_work, and trisycl::detail::fiber_pool::subm

Here is the call graph for this function:



9.1.2.21.3.27 translate_input_port() [1/2] template<typename Geography >

```
static auto trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >↩
::translate_input_port (
            int port )  [inline], [static], [private]
```

Map the user input port number to the AXI stream switch port.

Parameters

in	*port*	is the user port to use

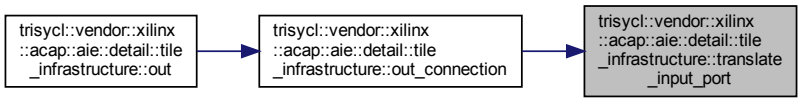
Definition at line 103 of file tile_infrastructure.hpp.

```
00103                                       {
00104     return axi_ss_t::translate_port(port, spl::me_0, spl::me_last,
00105                                     "The core input port is out of range");
00106   }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< axi_ss_geo >::translate_port.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::out_connection().

Here is the caller graph for this function:



9.1.2.21.3.28 translate_input_port() [2/2] template<typename Geography >

```
static auto trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >↩
::translate_input_port (
            int port )  [inline], [static], [private]
```

Map the user input port number to the AXI stream switch port.

Parameters

in	*port*	is the user port to use

Definition at line 103 of file tile_infrastructure.hpp.

```
00103                                       {
00104     return axi_ss_t::translate_port(port, spl::me_0, spl::me_last,
00105                                     "The core input port is out of range");
00106   }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< axi_ss_geo >::translate_port.

9.1.2.21.3.29 translate_output_port() [1/2] template<typename Geography >

```
static auto trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >↩
::translate_output_port (
            int port )  [inline], [static], [private]
```

Map the user output port number to the AXI stream switch port.

Parameters

in	*port*	is the user port to use

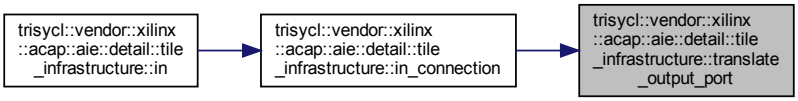
Definition at line 112 of file tile_infrastructure.hpp.

```
00112                                       {
00113     return axi_ss_t::translate_port(port, mpl::me_0, mpl::me_last,
00114                                     "The core output port is out of range");
00115   }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< axi_ss_geo >::translate_port.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::in_connection().

Here is the caller graph for this function:



9.1.2.21.3.30 translate_output_port() [2/2] template<typename Geography >

```
static auto trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >↩
::translate_output_port (
            int port )  [inline], [static], [private]
```

Map the user output port number to the AXI stream switch port.

Parameters

in	*port*	is the user port to use

Definition at line 112 of file tile_infrastructure.hpp.

```
00112                                       {
00113     return axi_ss_t::translate_port(port, mpl::me_0, mpl::me_last,
00114                                     "The core output port is out of range");
00115   }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< axi_ss_geo >::translate_port.

9.1.2.21.3.31 tx_dma() [1/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::tx_dma (
            int id )  [inline]
```

Get access to a transmit DMA.

Parameters

in	*id*	specifies which DMA to access

Definition at line 224 of file tile_infrastructure.hpp.

```
00224 { return *tx_dmas.at(id); }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::tx_dmas.

9.1.2.21.3.32 tx_dma() [2/2] template<typename Geography >

```
auto & trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::tx_dma (
            int id ) [inline]
```

Get access to a transmit DMA.

Parameters

in	*id*	specifies which DMA to access

Definition at line 224 of file tile_infrastructure.hpp.

```
00224 { return *tx_dmas.at(id); }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::tx_dmas.

9.1.2.21.3.33 wait() [1/2] template<typename Geography >

```
void trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::wait ( )
[inline]
```

Wait for the execution of the callable on this tile.

Definition at line 257 of file tile_infrastructure.hpp.

```
00257               { if (future_work.valid())
00258       future_work.get();
00259   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::future_work.

9.1.2.21.3.34 wait() [2/2] template<typename Geography >

```
void trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::wait ( )
[inline]
```

Wait for the execution of the callable on this tile.

Definition at line 257 of file tile_infrastructure.hpp.

```
00257               { if (future_work.valid())
00258       future_work.get();
00259   }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::future_work.

9.1.2.21.3.35 x() [1/2] template<typename Geography >

```
int trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::x ( )  [inline]
```

Get the horizontal coordinate.

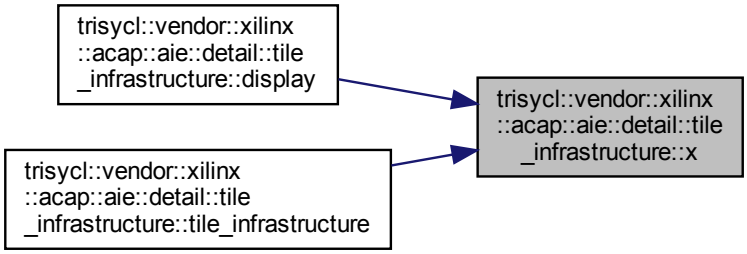
Definition at line 160 of file tile_infrastructure.hpp.

```
00160 { return x_coordinate; }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::x_coordinate.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display(), and trisycl::vendor::xilinx::acap::aie

Here is the caller graph for this function:



9.1.2.21.3.36 x() [2/2] template<typename Geography >

```
int trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::x ( )  [inline]
```

Get the horizontal coordinate.

Definition at line 160 of file tile_infrastructure.hpp.

```
00160 { return x_coordinate; }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::x_coordinate.

9.1.2.21.3.37 y() [1/2] template<typename Geography >

```
int trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::y ( )  [inline]
```

Get the vertical coordinate.

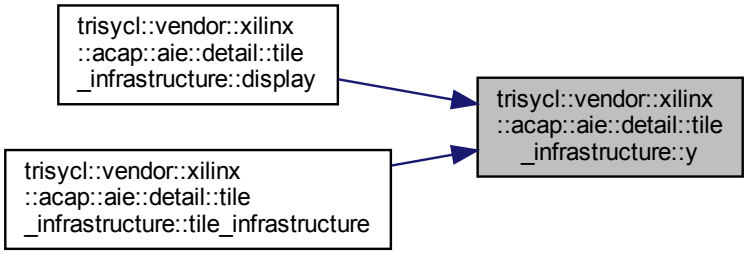
Definition at line 163 of file tile_infrastructure.hpp.

```
00163 { return y_coordinate; }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::y_coordinate.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display(), and trisycl::vendor::xilinx::acap::aie

Here is the caller graph for this function:



9.1.2.21.3.38 y() [2/2] template<typename Geography >

```
int trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::y ( )  [inline]
```

Get the vertical coordinate.

Definition at line 163 of file tile_infrastructure.hpp.

```
00163 { return y_coordinate; }
```

References trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::y_coordinate.

9.1.2.21.4 Member Data Documentation

9.1.2.21.4.1 axi_ss template<typename Geography >

```
axi_ss_t trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::axi_ss
[private]
```

The AXI stream switch of the core tile.

Definition at line 76 of file tile_infrastructure.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::connect(), trisycl::vendor::xilinx::acap::aie::d
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::in_connection(), trisycl::vendor::xilinx::acap::aie::detail::tile_
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::out_connection(), trisycl::vendor::xilinx::acap::aie::detail::tile
and trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::tile_infrastructure().

9.1.2.21.4.2 dev template<typename Geography >

```
void * trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::dev  [private]
```

Keep track of the aie::detail::device for hardware resource control in device mode or for debugging purpose for better messages.

Use void* for now to avoid cyclic header dependencies for now instead of the aie::detail::device

Definition at line 73 of file tile_infrastructure.hpp.

9.1.2.21.4.3 future_work

`template<typename Geography >`
`std::future< void > trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::future_work [private]`

Keep track of the std::thread execution in this tile.

Definition at line 96 of file tile_infrastructure.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::single_task(), and trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::wait().

9.1.2.21.4.4 mi

`template<typename Geography >`
`aie::memory_infrastructure trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::mi [private]`

Keep track of all the infrastructure tile memories of this device.

Definition at line 80 of file tile_infrastructure.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::mem().

9.1.2.21.4.5 tx_dmas

`template<typename Geography >`
`std::array< std::optional< sending_dma< axi_ss_t > >, axi_ss_geo::s_dma_size > trisycl::vendor::xilinx::acap Geography >::tx_dmas [private]`

Sending DMAs.

Use std::optional to postpone initialization

Definition at line 86 of file tile_infrastructure.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::tile_infrastructure(), and trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::tx_dma().

9.1.2.21.4.6 x_coordinate

`template<typename Geography >`
`int trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::x_coordinate [private]`

Keep the horizontal coordinate.

Definition at line 62 of file tile_infrastructure.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display(), trisycl::vendor::xilinx::acap::aie::de trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::out(), and trisycl::vendor::xilinx::acap::aie::detail::tile_infrast

9.1.2.21.4.7 y_coordinate `template<typename Geography >`

`int trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::y_coordinate [private]`

Keep the vertical coordinate.

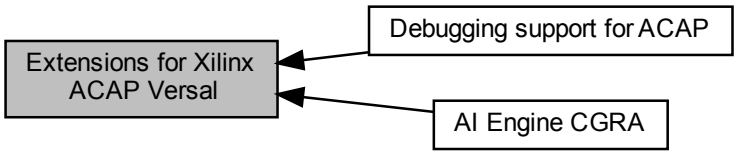
Definition at line 65 of file tile_infrastructure.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::display(), trisycl::vendor::xilinx::acap::aie::de trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::out(), and trisycl::vendor::xilinx::acap::aie::detail::tile_infrast

9.2 Extensions for Xilinx ACAP Versal

Some classes and functions to support devices including CPU, FPGA & AI Engine CGRA.

Collaboration diagram for Extensions for Xilinx ACAP Versal:



Modules

- AI Engine CGRA

  Extensions to support explicit AI Engine system-wide programming in C++.
- Debugging support for ACAP

  Some classes and functions to help debuging ACAP applications, such as verifying BSP forward progress.

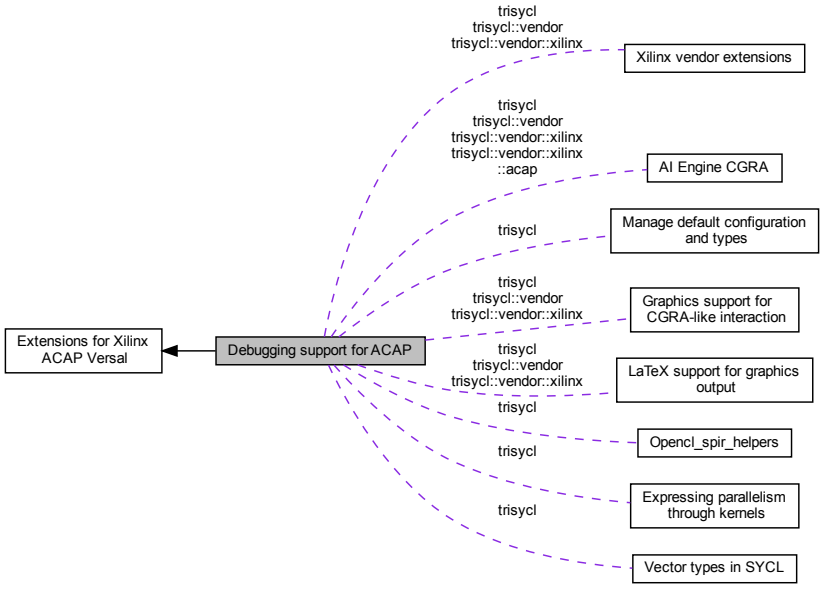
9.2.1 Detailed Description

Some classes and functions to support devices including CPU, FPGA & AI Engine CGRA.

9.3 Debugging support for ACAP

Some classes and functions to help debuging ACAP applications, such as verifying BSP forward progress.

Collaboration diagram for Debugging support for ACAP:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::debug

Classes

- class trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >

  Verify that an SPMD program has a bulk-synchronous behaviour across all the tiles of the CGRA. More...

9.3.1 Detailed Description

Some classes and functions to help debuging ACAP applications, such as verifying BSP forward progress.

9.3.2 Class Documentation

9.3.2.1 class trisycl::vendor::xilinx::acap::debug::bsp_checker

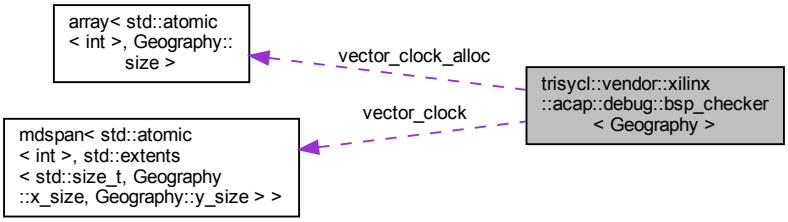
template<typename Geography>
class trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >

Verify that an SPMD program has a bulk-synchronous behaviour across all the tiles of the CGRA.

If there are some discrepancies, display the tile id which is too late or too early.

Definition at line 41 of file debug.hpp.

Collaboration diagram for trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >:



Public Member Functions

- void check (int x, int y)

  Check that none of the tiles executing this are separated by more than one execution step each other.
- void check (int x, int y)

  Check that none of the tiles executing this are separated by more than one execution step each other.

Private Attributes

- std::array< std::atomic< int >, Geography::size > vector_clock_alloc

  A vector clock to compare differerent tiles together.
- std::mdspan< std::atomic< int >, std::extents< std::size_t, Geography::x_size, Geography::y_size > > vector_clock { &vector_clock_alloc[0] }

9.3.2.1.1 Member Function Documentation

9.3.2.1.1.1 check() [1/2] template<typename Geography >

```
void trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >::check (
              int x,
              int y )  [inline]
```

Check that none of the tiles executing this are separated by more than one execution step each other.

This functions has to be called in a BSP synchronous way by all the tiles.

Definition at line 55 of file debug.hpp.

```
00055                                {
00056     ++vector_clock[x, y];
00057     /* While the algorithm execution is not atomic, it is enough to
00058 find some asynchronous bugs */
00059     auto [min_element, max_element] = std::minmax_element(
00060         vector_clock_alloc.cbegin(), vector_clock_alloc.cend());
00061
00062     if (*max_element - *min_element > 1) {
00063       TRISYCL_DUMP_T(std::dec « "compute(" « x « ',' « y
00064                               « ") vector clock min = " « *min_element
00065                               « ", max = " « *max_element);
00066       std::terminate();
00067     }
00068   }
```

References TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >::vector_clock, trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >::vector_clock_alloc, x, and trisycl::y.

9.3.2.1.1.2 check() [2/2] template<typename Geography >

```
void trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >::check (
              int x,
              int y )  [inline]
```

Check that none of the tiles executing this are separated by more than one execution step each other.

This functions has to be called in a BSP synchronous way by all the tiles.

Definition at line 55 of file debug.hpp.

```
00055                                {
00056     ++vector_clock[x, y];
00057     /* While the algorithm execution is not atomic, it is enough to
00058 find some asynchronous bugs */
00059     auto [min_element, max_element] = std::minmax_element(
00060         vector_clock_alloc.cbegin(), vector_clock_alloc.cend());
00061
00062     if (*max_element - *min_element > 1) {
00063       TRISYCL_DUMP_T(std::dec « "compute(" « x « ',' « y
00064                               « ") vector clock min = " « *min_element
00065                               « ", max = " « *max_element);
00066       std::terminate();
00067     }
00068   }
```

References TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >::vector_clock, trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >::vector_clock_alloc, x, and trisycl::y.

9.3.2.1.2 Member Data Documentation

9.3.2.1.2.1 vector_clock template<typename Geography >

```
std::mdspan< std::atomic< int >, std::extents< std::size_t, Geography::x_size, Geography↩
::y_size > > trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >::vector_clock {
&vector_clock_alloc[0] }  [private]
```

Definition at line 46 of file debug.hpp.

Referenced by trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >::check().

9.3.2.1.2.2 vector_clock_alloc `template<typename Geography >`

`std::array< std::atomic< int >, Geography::size > trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >::vector_clock_alloc [private]`

A vector clock to compare differerent tiles together.

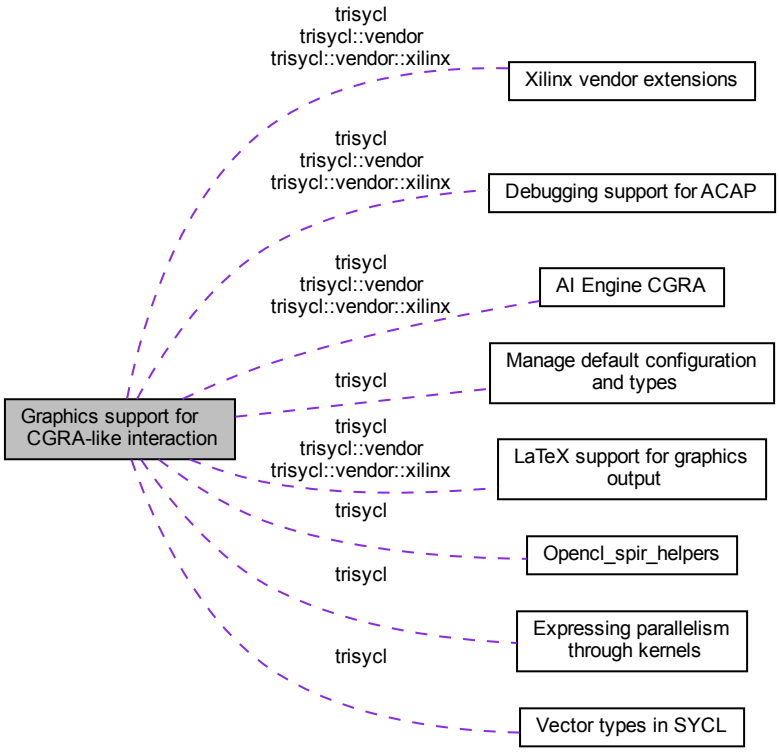
Definition at line 43 of file debug.hpp.

Referenced by trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >::check().

9.4 Graphics support for CGRA-like interaction

This can create a graphics application and typically update some sub-images on a frame.

Collaboration diagram for Graphics support for CGRA-like interaction:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::graphics

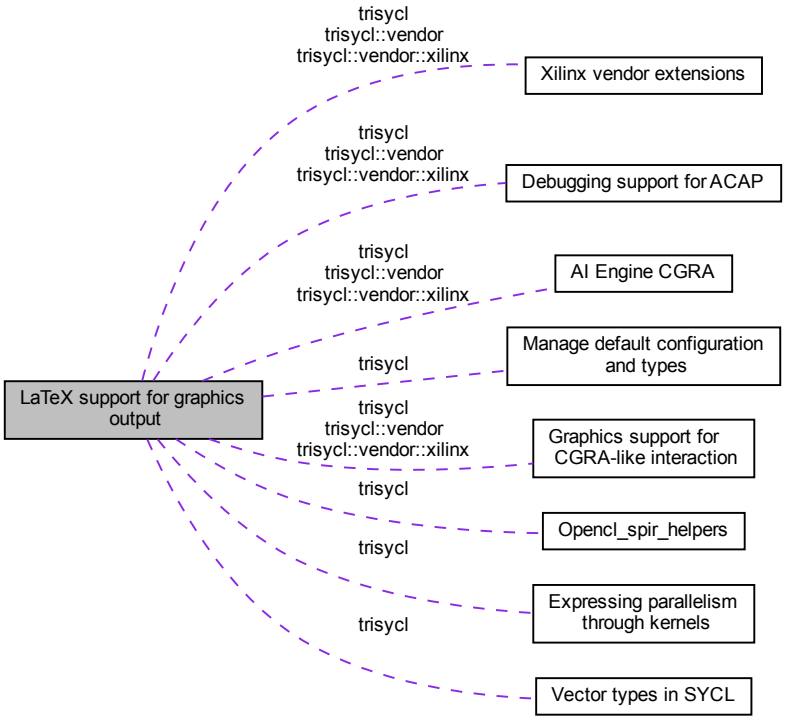
9.4.1 Detailed Description

This can create a graphics application and typically update some sub-images on a frame.

9.5 LaTeX support for graphics output

This can generate LaTeX output used to display internal configuration for debug or teaching purpose.

Collaboration diagram for LaTeX support for graphics output:



Namespaces

- namespace trisycl

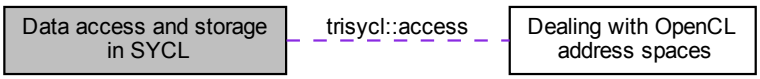
  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::latex

9.5.1 Detailed Description

This can generate LaTeX output used to display internal configuration for debug or teaching purpose.

9.6 Data access and storage in SYCL

Collaboration diagram for Data access and storage in SYCL:



Namespaces

- namespace trisycl::access

  Describe the type of access by kernels.

Classes

- class trisycl::vendor::trisycl::pipe::cout

  triSYCL extension for SYCL pipe object sending data to std::cout More...
- class trisycl::accessor< DataType, Dimensions, AccessMode, Target >

  The accessor abstracts the way buffer or pipe data are accessed inside a kernel in a multidimensional variable length array way. More...
- class trisycl::detail::accessor_base

  The buffer accessor base is here mainly to be accessed from the task without dependency on the buffer data types. More...
- class trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >

  The local accessor specialization abstracts the way local memory is allocated to a kernel to be shared between work-items of the same work-group. More...
- class trisycl::buffer< T, Dimensions, Allocator >

  A SYCL buffer is a multidimensional variable length array (à la C99 VLA or even Fortran before) that is used to store data to work on. More...
- class trisycl::detail::accessor< T, Dimensions, Mode, Target >

  The buffer accessor abstracts the way buffer data are accessed inside a kernel in a multidimensional variable length array way. More...
- class trisycl::detail::buffer< T, Dimensions >

  A SYCL buffer is a multidimensional variable length array (à la C99 VLA or even Fortran before) that is used to store data to work on. More...
- struct trisycl::detail::buffer_base

  Factorize some template independent buffer aspects in a base class. More...
- class trisycl::detail::buffer_waiter< T, Dimensions, Allocator >

  A helper class to wait for the final buffer destruction if the conditions for blocking are met. More...
- struct trisycl::image< Dimensions >
- class trisycl::pipe< Name, T, MinCapacity, SizeT >

  SYCL pipe object. More...

Typedefs

- template<typename T >
  using trisycl::buffer_allocator = std::allocator< T >

  The allocator objects give the programmer some control on how the memory is allocated inside SYCL.
- template<typename T >
  using trisycl::image_allocator = std::allocator< T >

  The allocator used for the `image` inside SYCL.
- template<typename T >
  using trisycl::map_allocator = std::allocator< T >

  The allocator used to map the memory at the same place.

Functions

- template<std::forward_iterator FIB, std::forward_iterator FIE>
  trisycl::buffer (FIB begin, FIE end) -> buffer< typename std::iterator_traits< FIB >::value_type, 1 >

  A deduction guide to infer the buffer type from the read-write forward iterators into a copy-in on construction and copy-back on destruction.
- trisycl::buffer (auto &host_data) -> buffer< std::ranges::range_value_t< decltype(host_data)>, 1 >

  A deduction guide to infer the buffer type from the read-write range.
- template<typename BufferDetail >
  static std::shared_ptr< detail::task > trisycl::detail::buffer_add_to_task (BufferDetail buf, handler *command_group_handler, bool is_write_mode)

  Proxy function to avoid some circular type recursion.
- static std::shared_ptr< detail::task > trisycl::detail::add_buffer_to_task (handler *command_group_handler, std::shared_ptr< detail::buffer_base > b, bool is_write_mode)
- template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
  auto trisycl::detail::waiter (detail::buffer< T, Dimensions > *b)

  Helper function to create a new buffer_waiter.

9.6.1 Detailed Description

9.6.2 Class Documentation

9.6.2.1 class trisycl::vendor::trisycl::pipe::cout

triSYCL extension for SYCL pipe object sending data to std::cout

Experiment with a new pipe extension to send data to std::cout with a simplified pipe extension like with:

```
sycl::vendor::trisycl::pipe::cout::write("salut !\n");
sycl::vendor::trisycl::pipe::cout::stream() « "hello " « 42 « std::endl;
```

Definition at line 28 of file cout.hpp.

Static Public Member Functions

- static auto & stream ()

  Provide the usual stream interface to std::cout.
- template<typename T >
  static void write (const T &value)

  Send some value to std::cout.
- static auto & stream ()

  Provide the usual stream interface to std::cout.
- template<typename T >
  static void write (const T &value)

  Send some value to std::cout.

9.6.2.1.1 Member Function Documentation

9.6.2.1.1.1 stream() [1/2] static auto & trisycl::vendor::trisycl::pipe::cout::stream ( ) [inline], [static]

Provide the usual stream interface to std::cout.

Definition at line 33 of file cout.hpp.

```
00033                           {
00034     return std::cout;
00035   }
```

Referenced by write().

Here is the caller graph for this function:



9.6.2.1.1.2 stream() [2/2] static auto & trisycl::vendor::trisycl::pipe::cout::stream ( ) [inline], [static]

Provide the usual stream interface to std::cout.

Definition at line 33 of file cout.hpp.

```
00033                           {
00034     return std::cout;
00035   }
```

9.6.2.1.1.3 write() [1/2] template<typename T >

```
static void trisycl::vendor::trisycl::pipe::cout::write (
            const T & value ) [inline], [static]
```

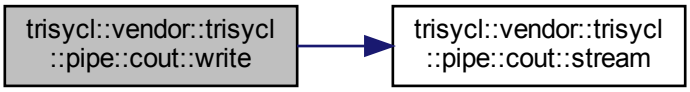
Send some value to std::cout.

Definition at line 39 of file cout.hpp.

```
00039                                   {
00040     stream() « value;
00041   }
```

References stream().

Here is the call graph for this function:



9.6.2.1.1.4 write() [2/2] template<typename T >

```
static void trisycl::vendor::trisycl::pipe::cout::write (
              const T & value )  [inline], [static]
```

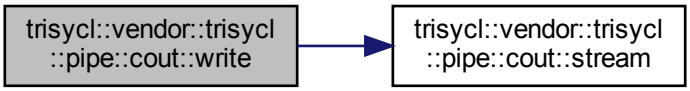
Send some value to std::cout.

Definition at line 39 of file cout.hpp.

```
00039                                   {
00040     stream() « value;
00041   }
```

References stream().

Here is the call graph for this function:



9.6.2.2 class trisycl::accessor

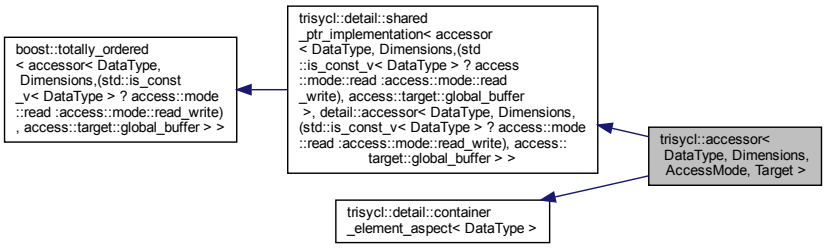
template<typename DataType, int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer>
class trisycl::accessor< DataType, Dimensions, AccessMode, Target >

The accessor abstracts the way buffer or pipe data are accessed inside a kernel in a multidimensional variable length array way.

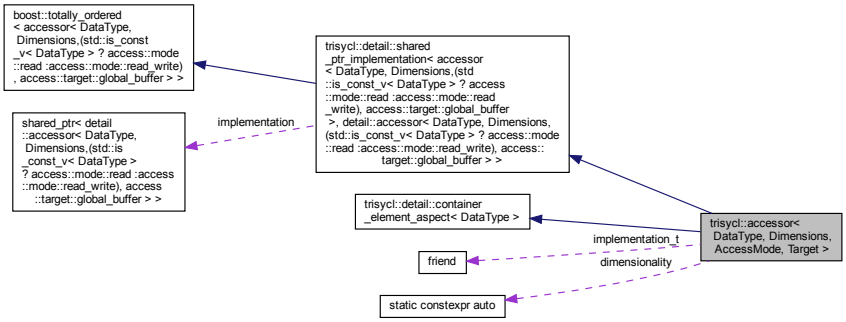
Todo Implement it for images according so section 3.3.4.5

Definition at line 54 of file accessor.hpp.

Inheritance diagram for trisycl::accessor< DataType, Dimensions, AccessMode, Target >:



Collaboration diagram for trisycl::accessor< DataType, Dimensions, AccessMode, Target >:



Public Member Functions

- template<typename Allocator >
  accessor (buffer< DataType, Dimensions, Allocator > &target_buffer, handler &command_group_handler)

  Construct a buffer accessor from a buffer using a command group handler object from the command group scope.
- template<typename Allocator >
  accessor (buffer< DataType, Dimensions, Allocator > &target_buffer)

  Construct a buffer accessor from a buffer.
- template<typename Allocator >
  accessor (buffer< DataType, Dimensions, Allocator > &target_buffer, handler &command_group_handler, const range< Dimensions > &offset, const range< Dimensions > &range)

  Construct a buffer accessor from a buffer given a specific range for access permissions and an offset that provides the starting point for the access range using a command group handler object from the command group scope.
- accessor (const range< Dimensions > &allocation_size, handler &command_group_handler)

  Construct an accessor of dimension Dimensions with elements of type DataType using the passed range to specify the size in each dimension.
- auto get_range () const

  Return a range object representing the size of the buffer in terms of number of elements in each dimension as passed to the constructor.
- auto get_count () const

  Returns the total number of elements behind the accessor.

- auto get_size () const

  Returns the size of the underlying buffer storage in bytes.
- template<std::integral... T>
  decltype(auto) operator[ ] (T... indices)

  Use the accessor with integers à la [i1][i2][i3] or C++23 [i1, i2,...].
- template<std::integral... T>
  decltype(auto) operator[ ] (T... indices) const

  Use the accessor with integers à la [i1][i2][i3] or C++23 [i1, i2,...].
- auto & operator[ ] (const id< dimensionality > &index)

  To use the accessor with [id<>].
- auto & operator[ ] (const id< dimensionality > &index) const

  To use the accessor with [id<>].
- auto & operator[ ] (const item< dimensionality > &index)

  To use an accessor with [item<>].
- auto & operator[ ] (const item< dimensionality > &index) const

  To use an accessor with [item<>].
- auto & operator[ ] (const nd_item< dimensionality > &index)

  To use an accessor with an [nd_item<>].
- auto & operator[ ] (const nd_item< dimensionality > &index) const

  To use an accessor with an [nd_item<>].
- accessor_detail::reference operator* ()

  Get the first element of the accessor.
- accessor_detail::reference operator* () const

  Get the first element of the accessor.
- auto get_pointer () const

  Get the pointer to the start of the data.
- accessor_detail::iterator begin () const

  Forward all the iterator functions to the implementation.
- accessor_detail::iterator end () const
- accessor_detail::const_iterator cbegin () const
- accessor_detail::const_iterator cend () const
- accessor_detail::reverse_iterator rbegin () const
- accessor_detail::reverse_iterator rend () const
- accessor_detail::const_reverse_iterator crbegin () const
- accessor_detail::const_reverse_iterator crend () const

Static Public Member Functions

- static auto constexpr access_mode ()

  Introspect the `access_mode`.
- static auto constexpr target ()

  Introspect the `target`.

Static Public Attributes

- static constexpr auto dimensionality = Dimensions

Private Types

- using accessor_detail = typename detail::accessor< DataType, Dimensions, AccessMode, Target >
- using implementation_t = typename accessor::shared_ptr_implementation

Private Attributes

- friend implementation_t

Additional Inherited Members

9.6.2.2.1 Member Typedef Documentation

9.6.2.2.1.1 accessor_detail

```
template<typename DataType , int Dimensions, access::mode Access↩
Mode = (std::is_const_v<DataType> ?  access::mode::read :  access::mode::read_write), access::target
Target = access::target::global_buffer>
using trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor_detail = typename
detail::accessor<DataType, Dimensions, AccessMode, Target>  [private]
```

Definition at line 67 of file accessor.hpp.

9.6.2.2.1.2 implementation_t

```
template<typename DataType , int Dimensions, access::mode Access↩
Mode = (std::is_const_v<DataType> ?  access::mode::read :  access::mode::read_write), access::target
Target = access::target::global_buffer>
using trisycl::accessor< DataType, Dimensions, AccessMode, Target >::implementation_t = typename
accessor::shared_ptr_implementation  [private]
```

Definition at line 73 of file accessor.hpp.

9.6.2.2.2 Constructor & Destructor Documentation

9.6.2.2.2.1 accessor() [1/4]

```
template<typename DataType , int Dimensions, access::mode Access↩
Mode = (std::is_const_v<DataType> ?  access::mode::read :  access::mode::read_write), access::target
Target = access::target::global_buffer>
template<typename Allocator >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            buffer< DataType, Dimensions, Allocator > & target_buffer,
            handler & command_group_handler )  [inline]
```

Construct a buffer accessor from a buffer using a command group handler object from the command group scope.

Constructor only available for global_buffer or constant_buffer target.

access_target defines the form of access being obtained.

Todo Add template allocator type in all the accessor constructors in the specification or just use a more opaque Buffer type?

Todo fix specification where access mode should be target instead

Definition at line 104 of file accessor.hpp.

```
00105                                         :  implementation_t {
00106     new detail::accessor<DataType, Dimensions, AccessMode, Target> {
00107       target_buffer.implementation->implementation, command_group_handler }
00108   } {
00109     static_assert(Target == access::target::global_buffer
00110                   || Target == access::target::constant_buffer,
00111                   "access target should be global_buffer or constant_buffer "
00112                   "when a handler is used");
00113     // Now the implementation is created, register it
00114     implementation->register_accessor();
00115   }
```

9.6.2.2.2 accessor() [2/4]

```
template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer>
template<typename Allocator >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            buffer< DataType, Dimensions, Allocator > & target_buffer )  [inline]
```

Construct a buffer accessor from a buffer.

Constructor only available for host_buffer target.

access_target defines the form of access being obtained.

Definition at line 125 of file accessor.hpp.

```
00126     :  implementation_t {
00127     new detail::accessor<DataType, Dimensions, AccessMode, Target> {
00128       target_buffer.implementation->implementation }
00129   } {
00130     static_assert(Target == access::target::host_buffer,
00131                   "without a handler, access target should be host_buffer");
00132   }
```

9.6.2.2.3 accessor() [3/4]

```
template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer>
template<typename Allocator >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            buffer< DataType, Dimensions, Allocator > & target_buffer,
            handler & command_group_handler,
            const range< Dimensions > & offset,
            const range< Dimensions > & range )  [inline]
```

Construct a buffer accessor from a buffer given a specific range for access permissions and an offset that provides the starting point for the access range using a command group handler object from the command group scope.

This accessor limits the processing of the buffer to the [offset, offset+range[for every dimension. Any other parts of the buffer will be unaffected.

Constructor only available for access modes global_buffer, and constant_buffer (see Table "Buffer accessor constructors"). access_target defines the form of access being obtained.

This accessor is recommended for discard-write and discard read write access modes, when the unaffected parts of the processing should be retained.

Definition at line 153 of file accessor.hpp.

```
00156                                       {
00157     TRISYCL_UNIMPL;
00158   }
```

References TRISYCL_UNIMPL.

9.6.2.2.4 accessor() [4/4]

```
template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer>
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            const range< Dimensions > & allocation_size,
            handler & command_group_handler ) [inline]
```

Construct an accessor of dimension Dimensions with elements of type DataType using the passed range to specify the size in each dimension.

It needs as a parameter a command group handler object from the command group scope. Constructor only available if AccessMode is local, see Table 3.25.

Definition at line 169 of file accessor.hpp.

```
00171     :  implementation_t { new detail::accessor<DataType,
00172                                                Dimensions,
00173                                                AccessMode,
00174                                                access::target::local> {
00175       allocation_size, command_group_handler
00176         }
00177   }
00178   {
00179     static_assert(Target == access::target::local,
00180                   "This accessor constructor requires "
00181                   "access target be local");
00182   }
```

9.6.2.2.3 Member Function Documentation

9.6.2.2.3.1 access_mode()

```
template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer>
static auto constexpr trisycl::accessor< DataType, Dimensions, AccessMode, Target >::access_mode ( ) [inline], [static], [constexpr]
```

Introspect the `access_mode`.

Definition at line 80 of file accessor.hpp.

```
00080 { return AccessMode; }
```

9.6.2.2.3.2 begin()

```
template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer>
accessor_detail::iterator trisycl::accessor< DataType, Dimensions, AccessMode, Target >::begin ( ) const [inline]
```

Forward all the iterator functions to the implementation.

Todo Add these functions to the specification

Todo The fact that the lambda capture make a const copy of the accessor is not yet elegantly managed... The issue is that begin()/end() dispatch is made according to the accessor constness and not from the array member constness...

Todo try to solve it by using some enable_if on array constness?

Todo The issue is that the end may not be known if it is implemented by a raw OpenCL cl_mem... So only provide on the device the iterators related to the start? Actually the accessor needs to know a part of the shape to have the multidimentional addressing. So this only require a size_t more...

Todo Factor out these in a template helper

Definition at line 345 of file accessor.hpp.

```
00345                                                {
00346     return implementation->begin();
00347   }
```

9.6.2.2.3.3 cbegin()

`template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer> accessor_detail::const_iterator trisycl::accessor< DataType, Dimensions, AccessMode, Target >::cbegin ( ) const [inline]`

Definition at line 355 of file accessor.hpp.

```
00355                                                 {
00356     return implementation->cbegin();
00357   }
```

9.6.2.2.3.4 cend()

`template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer> accessor_detail::const_iterator trisycl::accessor< DataType, Dimensions, AccessMode, Target >::cend ( ) const [inline]`

Definition at line 360 of file accessor.hpp.

```
00360                                               {
00361     return implementation->cend();
00362   }
```

9.6.2.2.3.5 crbegin()

`template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer> accessor_detail::const_reverse_iterator trisycl::accessor< DataType, Dimensions, AccessMode, Target >::crbegin ( ) const [inline]`

Definition at line 375 of file accessor.hpp.

```
00375                                                       {
00376     return implementation->rbegin();
00377   }
```

9.6.2.2.3.6 crend() template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer> accessor_detail::const_reverse_iterator trisycl::accessor< DataType, Dimensions, AccessMode, Target >::crend ( ) const [inline]

Definition at line 380 of file accessor.hpp.

```
00380                                                                {
00381     return implementation->rend();
00382   }
```

9.6.2.2.3.7 end() template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer> accessor_detail::iterator trisycl::accessor< DataType, Dimensions, AccessMode, Target >::end ( ) const [inline]

Definition at line 351 of file accessor.hpp.

```
00351                                                  {
00352     return implementation->end();
00353   }
```

9.6.2.2.3.8 get_count() template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer> auto trisycl::accessor< DataType, Dimensions, AccessMode, Target >::get_count ( ) const [inline]

Returns the total number of elements behind the accessor.

Equal to get_range()[0] * ... * get_range()[Dimensions-1].

Todo Move on https://cvs.khronos.org/bugzilla/show_bug.cgi?id=15564 and https://cvs.khronos.org/bugzilla/show_bug.cgi?id=14404

Definition at line 213 of file accessor.hpp.

```
00213                            {
00214     return implementation->get_count();
00215   }
```

9.6.2.2.3.9 get_pointer() template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer> auto trisycl::accessor< DataType, Dimensions, AccessMode, Target >::get_pointer ( ) const [inline]

Get the pointer to the start of the data.

Todo Should it be named data() instead?

Definition at line 318 of file accessor.hpp.

```
00318                          {
00319     return implementation->get_pointer();
00320   }
```

9.6.2.2.3.10 get_range()

`template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer>`
`auto trisycl::accessor< DataType, Dimensions, AccessMode, Target >::get_range ( ) const [inline]`

Return a range object representing the size of the buffer in terms of number of elements in each dimension as passed to the constructor.

Todo Move on https://cvs.khronos.org/bugzilla/show_bug.cgi?id=15564 and https://cvs.khronos.org/bugzilla/show_bug.cgi?id=14404

Definition at line 193 of file accessor.hpp.

```
00193                           {
00194     /* Interpret the shape which is a pointer to the first element as an
00195 array of Dimensions elements so that the range<Dimensions>
00196 constructor is happy with this collection
00197
00198 \todo Add also a constructor in range<> to accept a const
00199 std::size_t *?
00200 */
00201     return implementation->get_range();
00202   }
```

9.6.2.2.3.11 get_size()

`template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer>`
`auto trisycl::accessor< DataType, Dimensions, AccessMode, Target >::get_size ( ) const [inline]`

Returns the size of the underlying buffer storage in bytes.

Todo It is incompatible with buffer get_size() in the spec

Todo Move on https://cvs.khronos.org/bugzilla/show_bug.cgi?id=15564 and https://cvs.khronos.org/bugzilla/show_bug.cgi?id=14404

Definition at line 226 of file accessor.hpp.

```
00226                           {
00227     return implementation->get_size();
00228   }
```

9.6.2.2.3.12 operator*() [1/2]

`template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer>`
`accessor_detail::reference trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator* ( ) [inline]`

Get the first element of the accessor.

Useful with an accessor on a scalar for example.

Todo Add in the specification

Definition at line 294 of file accessor.hpp.

```
00294                                   {
00295     return **implementation;
00296   }
```

9.6.2.2.3.13 operator*() [2/2]

```
template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer>
accessor_detail::reference trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator* ( ) const [inline]
```

Get the first element of the accessor.

Useful with an accessor on a scalar for example.

Todo Add in the specification?

Todo Add the concept of 0-dim buffer and accessor for scalar and use an implicit conversion to value_type reference to access the value with the accessor?

Definition at line 309 of file accessor.hpp.

```
00309                                                 {
00310     return **implementation;
00311   }
```

9.6.2.2.3.14 operator[]() [1/8]

```
template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer>
auto & trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator[] (
            const id< dimensionality > & index ) [inline]
```

To use the accessor with [id<>].

Definition at line 249 of file accessor.hpp.

```
00249                                                {
00250     return (*implementation)[index];
00251   }
```

9.6.2.2.3.15 operator[]() [2/8]

```
template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer>
auto & trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator[] (
            const id< dimensionality > & index ) const [inline]
```

To use the accessor with [id<>].

Definition at line 254 of file accessor.hpp.

```
00254                                                      {
00255     return (*implementation)[index];
00256   }
```

9.6.2.2.3.16 operator[]() [3/8]

```
template<typename DataType , int Dimensions, access::mode Access↩
Mode = (std::is_const_v<DataType> ?  access::mode::read :  access::mode::read_write), access::target
Target = access::target::global_buffer>
auto & trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator[] (
            const item< dimensionality > & index )  [inline]
```

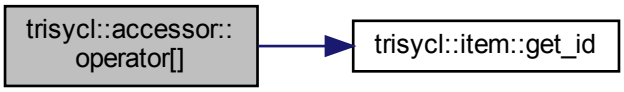
To use an accessor with [item<>].

Definition at line 260 of file accessor.hpp.

```
00260                                                {
00261     return (*this)[index.get_id()];
00262   }
```

References trisycl::item< Dimensions, with_offset >::get_id().

Here is the call graph for this function:



9.6.2.2.3.17 operator[]() [4/8]

```
template<typename DataType , int Dimensions, access::mode Access↩
Mode = (std::is_const_v<DataType> ?  access::mode::read :  access::mode::read_write), access::target
Target = access::target::global_buffer>
auto & trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator[] (
            const item< dimensionality > & index ) const  [inline]
```

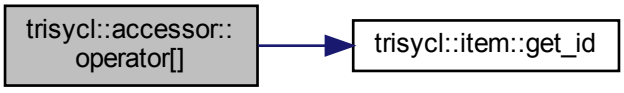
To use an accessor with [item<>].

Definition at line 266 of file accessor.hpp.

```
00266                                                      {
00267     return (*this)[index.get_id()];
00268   }
```

References trisycl::item< Dimensions, with_offset >::get_id().

Here is the call graph for this function:



9.6.2.2.3.18 operator[]() [5/8]

```
template<typename DataType , int Dimensions, access::mode Access↩
Mode = (std::is_const_v<DataType> ?  access::mode::read :  access::mode::read_write), access::target
Target = access::target::global_buffer>
auto & trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator[] (
            const nd_item< dimensionality > & index )  [inline]
```

To use an accessor with an [nd_item<>].

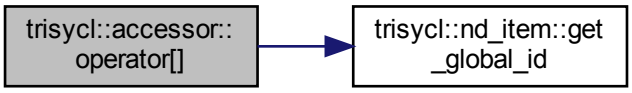
Todo Add in the specification because used by HPC-GPU slide 22

Definition at line 275 of file accessor.hpp.

```
00275                                                    {
00276     return (*this)[index.get_global_id()];
00277   }
```

References trisycl::nd_item< Dimensions >::get_global_id().

Here is the call graph for this function:



9.6.2.2.3.19 operator[]() [6/8]

```
template<typename DataType , int Dimensions, access::mode Access↩
Mode = (std::is_const_v<DataType> ?  access::mode::read :  access::mode::read_write), access::target
Target = access::target::global_buffer>
auto & trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator[] (
            const nd_item< dimensionality > & index ) const  [inline]
```

To use an accessor with an [nd_item<>].

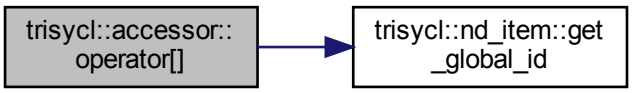
Todo Add in the specification because used by HPC-GPU slide 22

Definition at line 283 of file accessor.hpp.

```
00283                                                          {
00284     return (*this)[index.get_global_id()];
00285   }
```

References trisycl::nd_item< Dimensions >::get_global_id().

Here is the call graph for this function:



9.6.2.2.3.20 operator[]() [7/8]

```
template<typename DataType , int Dimensions, access::mode Access↩
Mode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target
Target = access::target::global_buffer>
template<std::integral... T>
decltype(auto) trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator[] (
            T... indices ) [inline]
```

Use the accessor with integers à la [i1][i2][i3] or C++23 [i1, i2,...].

Returns

decltype(auto) to return either a reference to the final element when the indexing has been fully resolved or a proxy object to handle the remaining []

Definition at line 235 of file accessor.hpp.

```
00235                                                                    {
00236     return (*implementation)[indices...];
00237   }
```

9.6.2.2.3.21 operator[]() [8/8]

```
template<typename DataType , int Dimensions, access::mode Access↩
Mode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target
Target = access::target::global_buffer>
template<std::integral... T>
decltype(auto) trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator[] (
            T... indices ) const [inline]
```

Use the accessor with integers à la [i1][i2][i3] or C++23 [i1, i2,...].

Returns

decltype(auto) to return either a reference to the final element when the indexing has been fully resolved or a proxy object to handle the remaining []

Definition at line 244 of file accessor.hpp.

```
00244                                                                    {
00245     return (*implementation)[indices...];
00246   }
```

9.6.2.2.3.22 rbegin()

```
template<typename DataType , int Dimensions, access::mode AccessMode =
(std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target
Target = access::target::global_buffer>
accessor_detail::reverse_iterator trisycl::accessor< DataType, Dimensions, AccessMode, Target
>::rbegin ( ) const [inline]
```

Definition at line 365 of file accessor.hpp.

```
00365                                                  {
00366     return implementation->rbegin();
00367   };
```

9.6.2.2.3.23 rend() template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer> accessor_detail::reverse_iterator trisycl::accessor< DataType, Dimensions, AccessMode, Target >::rend ( ) const [inline]

Definition at line 370 of file accessor.hpp.

```
00370                                                                    {
00371     return implementation->rend();
00372   }
```

9.6.2.2.3.24 target() template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer> static auto constexpr trisycl::accessor< DataType, Dimensions, AccessMode, Target >::target ( ) [inline], [static], [constexpr]

Introspect the `target`.

Definition at line 83 of file accessor.hpp.

```
00083 { return Target; }
```

9.6.2.2.4 Member Data Documentation

9.6.2.2.4.1 dimensionality template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer> constexpr auto trisycl::accessor< DataType, Dimensions, AccessMode, Target >::dimensionality = Dimensions [static], [constexpr]

Todo in the specification: store the dimension for user request

Definition at line 63 of file accessor.hpp.

9.6.2.2.4.2 implementation_t template<typename DataType , int Dimensions, access::mode AccessMode = (std::is_const_v<DataType> ? access::mode::read : access::mode::read_write), access::target Target = access::target::global_buffer> friend trisycl::accessor< DataType, Dimensions, AccessMode, Target >::implementation_t [private]

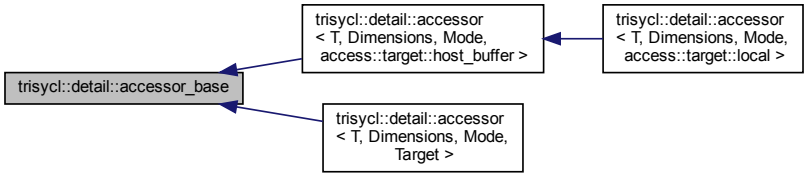
Definition at line 76 of file accessor.hpp.

9.6.2.3 class trisycl::detail::accessor_base

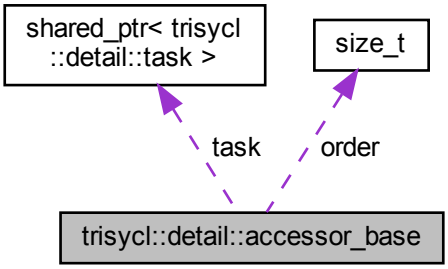
The buffer accessor base is here mainly to be accessed from the task without dependency on the buffer data types.

Definition at line 37 of file accessor_base.hpp.

Inheritance diagram for trisycl::detail::accessor_base:



Collaboration diagram for trisycl::detail::accessor_base:



Public Member Functions

- void set_order (std::size_t o)

  Set the order of the accessor.
- auto get_order ()

  Get the order of the accessor.
- virtual boost::compute::buffer get_cl_buffer () const =0

  Get the boost::compute::buffer or throw if unset.
- virtual ~accessor_base ()

Protected Attributes

- std::shared_ptr< detail::task > task

  The task where the accessor is used in.
- std::size_t order

  The declaration order of the accessor.

9.6.2.3.1 Constructor & Destructor Documentation

9.6.2.3.1.1 ~accessor_base() `virtual trisycl::detail::accessor_base::~accessor_base ( ) [inline], [virtual]`

Definition at line 66 of file accessor_base.hpp.

```
00066 {}
```

9.6.2.3.2 Member Function Documentation

9.6.2.3.2.1 get_cl_buffer() `virtual boost::compute::buffer trisycl::detail::accessor_base::get_↩ cl_buffer ( ) const [pure virtual]`

Get the boost::compute::buffer or throw if unset.

Implemented in trisycl::detail::accessor< T, Dimensions, Mode, Target >, and trisycl::detail::accessor< T, Dimensions, Mode, access

9.6.2.3.2.2 get_order() `auto trisycl::detail::accessor_base::get_order ( ) [inline]`

Get the order of the accessor.

Definition at line 56 of file accessor_base.hpp.

```
00056                     {
00057     return order;
00058   }
```

References order.

9.6.2.3.2.3 set_order() `void trisycl::detail::accessor_base::set_order ( std::size_t o ) [inline]`

Set the order of the accessor.

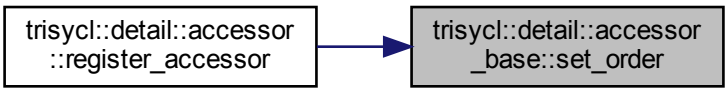
Definition at line 50 of file accessor_base.hpp.

```
00050                                {
00051     order = o;
00052   }
```

References order.

Referenced by trisycl::detail::accessor< T, Dimensions, Mode, Target >::register_accessor().

Here is the caller graph for this function:



9.6.2.3.3 Member Data Documentation

9.6.2.3.3.1 order `std::size_t trisycl::detail::accessor_base::order [protected]`

The declaration order of the accessor.

Definition at line 45 of file accessor_base.hpp.

Referenced by get_order(), and set_order().

9.6.2.3.3.2 task `std::shared_ptr<detail::task> trisycl::detail::accessor_base::task [protected]`

The task where the accessor is used in.

Definition at line 42 of file accessor_base.hpp.

9.6.2.4 class trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >

template<typename T, int Dimensions, access::mode Mode>
class trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >

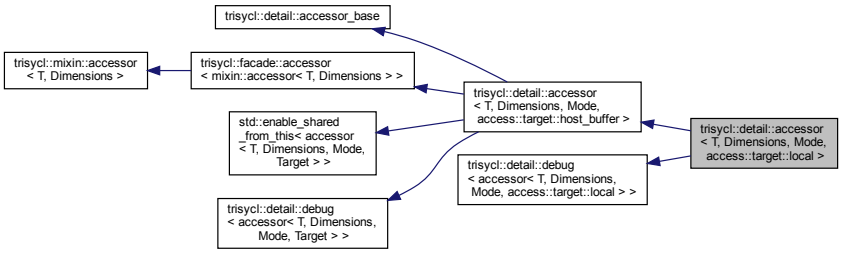
The local accessor specialization abstracts the way local memory is allocated to a kernel to be shared between work-items of the same work-group.

Since this a pure library implementation, implement it as a host_accessor using its own local buffer.

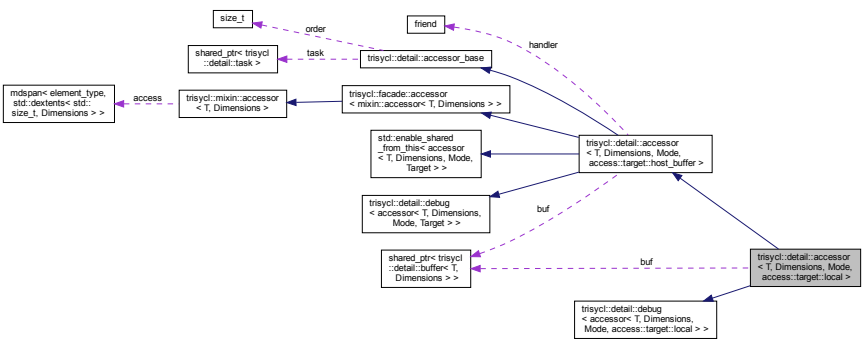
Todo Implement a real local accessor handling several work-group running in parallel

Definition at line 46 of file local_accessor.hpp.

Inheritance diagram for trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >:



Collaboration diagram for trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >:



Public Member Functions

- accessor (const range< Dimensions > &allocation_size, handler &command_group_handler)
  Construct a local accessor of the right size.

Private Types

- using hb = accessor< T, Dimensions, Mode, access::target::host_buffer >

Private Attributes

- std::shared_ptr< buffer< T, Dimensions > > buf

Additional Inherited Members

9.6.2.4.1 Member Typedef Documentation

9.6.2.4.1.1 hb `template<typename T , int Dimensions, access::mode Mode>`

```
using trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >::hb = accessor<T,
Dimensions, Mode, access::target::host_buffer>  [private]
```

Definition at line 50 of file local_accessor.hpp.

9.6.2.4.2 Constructor & Destructor Documentation

9.6.2.4.2.1 accessor() `template<typename T , int Dimensions, access::mode Mode>`

```
trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >::accessor (
            const range< Dimensions > & allocation_size,
            handler & command_group_handler )  [inline]
```

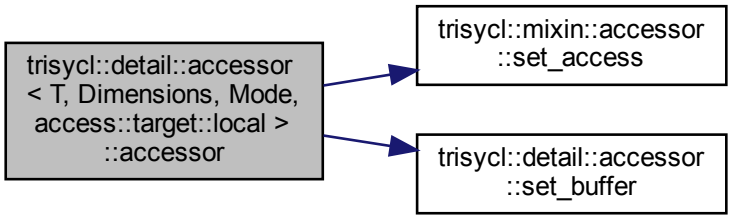
Construct a local accessor of the right size.

Definition at line 57 of file local_accessor.hpp.

```
00059       :  buf { std::make_shared<buffer<T, Dimensions»(allocation_size) } {
00060     this->set_buffer(buf);
00061     this->set_access(buf->access);
00062   }
```

References trisycl::mixin::accessor< T, Dimensions >::set_access(), and trisycl::detail::accessor< T, Dimensions, Mode, Target >::s

Here is the call graph for this function:



9.6.2.4.3 Member Data Documentation

9.6.2.4.3.1 buf `template<typename T , int Dimensions, access::mode Mode> std::shared_ptr<buffer<T, Dimensions> > trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >::buf [private]`

Definition at line 53 of file local_accessor.hpp.

9.6.2.5 class trisycl::buffer

template<typename T, int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>> class trisycl::buffer< T, Dimensions, Allocator >

A SYCL buffer is a multidimensional variable length array (à la C99 VLA or even Fortran before) that is used to store data to work on.

Todo There is a naming inconsistency in the specification between buffer and accessor on T versus datatype

Todo Finish allocator implementation

Todo Think about the need of an allocator when constructing a buffer from other buffers

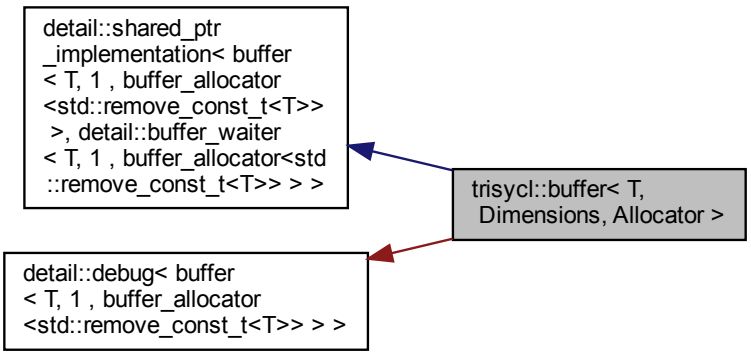
Todo Update the specification to have a non-const allocator for const buffer? Or do we rely on rebind_alloc<T>. But does this work with astate-full allocator?

Todo Add constructors from arrays so that in C++17 the range and type can be infered from the constructor

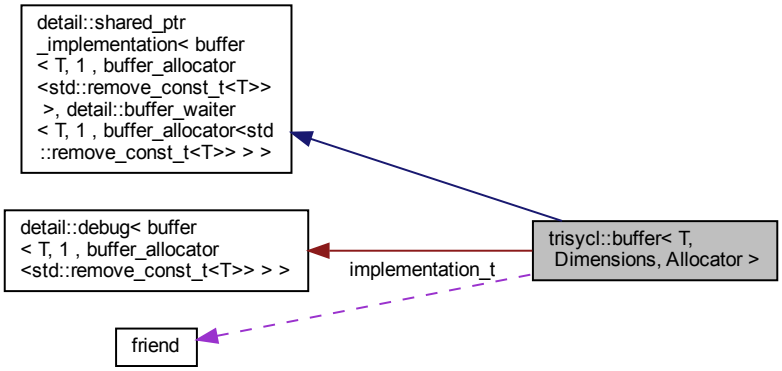
Todo Add constructors from array_ref

Definition at line 63 of file buffer.hpp.

Inheritance diagram for trisycl::buffer< T, Dimensions, Allocator >:



Collaboration diagram for trisycl::buffer< T, Dimensions, Allocator >:



Public Types

- using value_type = T

  The STL-like types.
- using reference = value_type &
- using const_reference = const value_type &
- using allocator_type = Allocator

Public Member Functions

- buffer ()=default

  Use default constructors so that we can create a new buffer copy from another one, with either a l-value or an r-value (for std::move() for example).
- buffer (const range< Dimensions > &r, Allocator allocator={})

  Create a new buffer of the given size with storage managed by the SYCL runtime.
- template<typename Dependent = T, typename = std::enable_if_t<!std::is_const<Dependent>::value>>
  buffer (const T *host_data, const range< Dimensions > &r, Allocator allocator={})

  Create a new buffer with associated host memory.
- buffer (T *host_data, const range< Dimensions > &r, Allocator allocator={})

  Create a new buffer with associated host memory.
- template<typename Range >
  requires requires { std::begin(std::declval<Range>()); std::end(std::declval<Range>()); } && (!detail::is_range_v<Range>)
  buffer (Range &host_data, Allocator allocator={})

  Create a new buffer with associated host memory from a range.
- buffer (shared_ptr_class< T > &host_data, const range< Dimensions > &buffer_range, trisycl::mutex_class &m, Allocator allocator={})

  Create a new buffer with associated memory, using the data in host_data.
- buffer (shared_ptr_class< T > host_data, const range< Dimensions > &buffer_range, Allocator allocator={})

  Create a new buffer with associated memory, using the data in host_data.
- template<typename InputIterator , typename ValueType = typename std::iterator_traits<InputIterator>::value_type>
  buffer (InputIterator start_iterator, InputIterator end_iterator, Allocator allocator={})

  Create a new allocated 1D buffer initialized from the given elements ranging from first up to one before last.

- buffer (buffer< T, Dimensions, Allocator > &b, const id< Dimensions > &base_index, const range< Dimensions > &sub_range, Allocator allocator={})

  Create a new sub-buffer without allocation to have separate accessors later.
- buffer (cl_mem mem_object, queue from_queue, event available_event={}, Allocator allocator={})

  Create a buffer from an existing OpenCL memory object associated with a context after waiting for an event signaling the availability of the OpenCL data.
- template<access::mode Mode, access::target Target = access::target::global_buffer>
  accessor< T, Dimensions, Mode, Target > get_access (handler &command_group_handler)

  Get an accessor to the buffer with the required mode.
- void mark_as_written ()

  Force the buffer to behave like if we had created an accessor in write mode.
- template<access::mode Mode>
  accessor< T, Dimensions, Mode, access::target::host_buffer > get_access ()

  Get a host accessor to the buffer with the required mode.
- auto get_range () const

  Return a range object representing the size of the buffer in terms of number of elements in each dimension as passed to the constructor.
- auto get_count () const

  Returns the total number of elements in the buffer.
- size_t get_size () const

  Returns the size of the buffer storage in bytes.
- auto use_count () const

  Returns the number of buffers that are shared/referenced.
- bool constexpr is_read_only () const

  Ask for read-only status of the buffer.
- void set_final_data (weak_ptr_class< T > finalData)

  Set destination of buffer data on destruction.
- void set_final_data (std::nullptr_t)

  Disable write-back on buffer destruction.
- bool is_cached (trisycl::context &ctx)

  Check if the buffer is already cached in a certain context.
- bool is_data_up_to_date (trisycl::context &ctx)

  Check if the data stored in the buffer is up-to-date in a certain context.
- template<typename Iterator , typename ValueType = typename std::iterator_traits< std::remove_reference_t<Iterator>>::value_type>
  void set_final_data (Iterator &&finalData)

  Set destination of buffer data on destruction.

Static Public Member Functions

- static auto constexpr rank ()

  Get the number of dimensions of the buffer.

Private Types

- using implementation_t = typename buffer::shared_ptr_implementation

Private Attributes

- friend implementation_t

9.6.2.5.1 Member Typedef Documentation

9.6.2.5.1.1 allocator_type

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
using trisycl::buffer< T, Dimensions, Allocator >::allocator_type = Allocator
```

Definition at line 76 of file buffer.hpp.

9.6.2.5.1.2 const_reference

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
using trisycl::buffer< T, Dimensions, Allocator >::const_reference = const value_type&
```

Definition at line 75 of file buffer.hpp.

9.6.2.5.1.3 implementation_t

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
using trisycl::buffer< T, Dimensions, Allocator >::implementation_t = typename buffer::shared_ptr_implementation  [private]
```

Definition at line 87 of file buffer.hpp.

9.6.2.5.1.4 reference

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
using trisycl::buffer< T, Dimensions, Allocator >::reference = value_type&
```

Definition at line 74 of file buffer.hpp.

9.6.2.5.1.5 value_type

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
using trisycl::buffer< T, Dimensions, Allocator >::value_type = T
```

The STL-like types.

Definition at line 73 of file buffer.hpp.

9.6.2.5.2 Constructor & Destructor Documentation

9.6.2.5.2.1 buffer() [1/10]

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
trisycl::buffer< T, Dimensions, Allocator >::buffer ( )  [default]
```

Use default constructors so that we can create a new buffer copy from another one, with either a l-value or an r-value (for std::move() for example).

Since we just copy the shared_ptr<> from the shared_ptr_implementation above, this is where/how the sharing magic is happening with reference counting in this case.

9.6.2.5.2.2 buffer() [2/10]

```
template<typename T , int Dimensions = 1, typename Allocator = buffer↩
_allocator<std::remove_const_t<T>>>
trisycl::buffer< T, Dimensions, Allocator >::buffer (
            const range< Dimensions > & r,
            Allocator allocator = {} )  [inline]
```

Create a new buffer of the given size with storage managed by the SYCL runtime.

The default behavior is to use the default host buffer allocator, in order to allow for host accesses. If the type of the buffer, has the const qualifier, then the default allocator will remove the qualifier to allow host access to the data.

Parameters

in	*r*	defines the size
in	*allocator*	is to be used by the SYCL runtime

Definition at line 120 of file buffer.hpp.

```
00120                                                                {})
00121     :  implementation_t { detail::waiter<T, Dimensions, Allocator>(
00122                           new detail::buffer<T, Dimensions> { r }) }
00123       {}
```

9.6.2.5.2.3 buffer() [3/10]

```
template<typename T , int Dimensions = 1, typename Allocator = buffer↩
_allocator<std::remove_const_t<T>>>
template<typename Dependent = T, typename = std::enable_if_t<!std::is_const<Dependent>↩
::value>>
trisycl::buffer< T, Dimensions, Allocator >::buffer (
            const T * host_data,
            const range< Dimensions > & r,
            Allocator allocator = {} )  [inline]
```

Create a new buffer with associated host memory.

Parameters

in	*host_data*	points to the storage and values used by the buffer
in	*r*	defines the size
in	*allocator*	is to be used by the SYCL runtime, of type `trisycl::buffer_allocator<T>` by default

The host address is
`const T*`

, so the host memory is read-only.

However, the typename T is not const so the device accesses can be both read and write accesses. Since, the host_data is const, this buffer is only initialized with this memory and there is no write after its destruction, unless there is another final data address given after construction of the buffer.

Only enable this constructor if it is not the same as the one with
`const T *host_data`

, which is when `T` is already a constant type.

Todo Actually this is redundant.

Definition at line 153 of file buffer.hpp.

```
00155                                 {})
00156     :  implementation_t { detail::waiter(new detail::buffer<T, Dimensions>
00157                           { host_data, r }) }
00158   {}
```

9.6.2.5.2.4 buffer() [4/10] template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>

```
trisycl::buffer< T, Dimensions, Allocator >::buffer (
             T * host_data,
             const range< Dimensions > & r,
             Allocator allocator = {} )  [inline]
```

Create a new buffer with associated host memory.

Parameters

in,out	*host_data*	points to the storage and values used by the buffer
in	*r*	defines the size
in	*allocator*	is to be used by the SYCL runtime, of type trisycl::buffer_allocator<T> by default

The memory is owned by the runtime during the lifetime of the object. Data is copied back to the host unless the user overrides the behavior using the set_final_data method. host_data points to the storage and values used by the buffer and range<Dimensions> defines the size.

Definition at line 177 of file buffer.hpp.

```
00179                                 {})
00180     :  implementation_t { detail::waiter(new detail::buffer<T, Dimensions>
00181                           { host_data, r }) }
00182   {}
```

9.6.2.5.2.5 buffer() [5/10] template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>

```
template<typename Range >
requires requires { std::begin(std::declval<Range>()); std::end(std::declval<Range>()); } &&
(!detail::is_range_v<Range>)
trisycl::buffer< T, Dimensions, Allocator >::buffer (
             Range & host_data,
             Allocator allocator = {} )  [inline]
```

Create a new buffer with associated host memory from a range.

Parameters

in,out	*host_data*	points to the storage and values used by the buffer
in	*allocator*	is to be used by the SYCL runtime, of type trisycl::buffer_allocator<T> by default

The memory is owned by the runtime during the lifetime of the object. Data is copied back to the host unless the

user overrides the behavior using the set_final_data method.

Definition at line 207 of file buffer.hpp.

```
00208                                {})
00209       :  buffer { host_data.begin(),
00210                   range { std::ranges::distance(host_data) } } {}
```

9.6.2.5.2.6 buffer() [6/10]

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
trisycl::buffer< T, Dimensions, Allocator >::buffer (
              shared_ptr_class< T > & host_data,
              const range< Dimensions > & buffer_range,
              trisycl::mutex_class & m,
              Allocator allocator = {} )  [inline]
```

Create a new buffer with associated memory, using the data in host_data.

Parameters

in,out	*host_data*	points to the storage and values used by the buffer
in	*r*	defines the size
in	*allocator*	is to be used by the SYCL runtime, of type trisycl::buffer_allocator<T> by default

The ownership of the host_data is shared between the runtime and the user. In order to enable both the user application and the SYCL runtime to use the same pointer, a trisycl::mutex_class is used. The mutex m is locked by the runtime whenever the data is in use and unlocked otherwise. Data is synchronized with host_data, when the mutex is unlocked by the runtime.

Todo update the specification to replace the pointer by a reference and provide the constructor with and without a mutex

Definition at line 233 of file buffer.hpp.

```
00236                                {}) {
00237     TRISYCL_UNIMPL;
00238   }
```

9.6.2.5.2.7 buffer() [7/10]

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
trisycl::buffer< T, Dimensions, Allocator >::buffer (
              shared_ptr_class< T > host_data,
              const range< Dimensions > & buffer_range,
              Allocator allocator = {} )  [inline]
```

Create a new buffer with associated memory, using the data in host_data.

Parameters

in,out	*host_data*	points to the storage and values used by the buffer
in	*r*	defines the size
in	*allocator*	is to be used by the SYCL runtime, of type trisycl::buffer_allocator<T> by default

The ownership of the host_data is shared between the runtime and the user. In order to enable both the user application and the SYCL runtime to use the same pointer, a trisycl::mutex_class is used.

Todo add this mutex-less constructor to the specification

Definition at line 259 of file buffer.hpp.

```
00261                                 {})
00262     :  implementation_t { detail::waiter(new detail::buffer<T, Dimensions>
00263                          { host_data, buffer_range }) }
00264   {}
```

9.6.2.5.2.8 buffer() [8/10]

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
template<typename InputIterator , typename ValueType = typename std::iterator_traits<InputIterator>::value_type>
trisycl::buffer< T, Dimensions, Allocator >::buffer (
            InputIterator start_iterator,
            InputIterator end_iterator,
            Allocator allocator = {} )  [inline]
```

Create a new allocated 1D buffer initialized from the given elements ranging from first up to one before last.

The data is copied to an intermediate memory position by the runtime. Data is written back to the same iterator set if the iterator is not a const iterator.

Parameters

in,out	*start_iterator*	points to the first element to copy
in	*end_iterator*	points to just after the last element to copy
in	*allocator*	is to be used by the SYCL runtime, of type trisycl::buffer_allocator<T> by default

Todo Implement the copy back at buffer destruction

Todo Generalize this for n-D and provide column-major and row-major initialization

Todo a reason to have this nD is that set_final_data(weak_ptr_class<T> & finalData) is actually doing this linearization anyway

Todo Allow read-only buffer construction too

Todo update the specification to deal with forward iterators instead and rewrite back only when it is non const and output iterator at least

Todo Allow initialization from ranges and collections à la STL

Definition at line 304 of file buffer.hpp.

```
00306                                 {}) :
00307     implementation_t { detail::waiter(new detail::buffer<T, Dimensions>
00308                        { start_iterator, end_iterator }) }
00309   {}
```

9.6.2.5.2.9 buffer() [9/10]

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
trisycl::buffer< T, Dimensions, Allocator >::buffer (
            buffer< T, Dimensions, Allocator > & b,
            const id< Dimensions > & base_index,
            const range< Dimensions > & sub_range,
            Allocator allocator = {} )  [inline]
```

Create a new sub-buffer without allocation to have separate accessors later.

Parameters

in,out	*b*	is the buffer with the real data
in	*base_index*	specifies the origin of the sub-buffer inside the buffer b
in	*sub_range*	specifies the size of the sub-buffer

Todo To be implemented

Todo Update the specification to replace index by id

Definition at line 326 of file buffer.hpp.

```
00329                                {}) { TRISYCL_UNIMPL; }
```

9.6.2.5.2.10 buffer() [10/10]

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
trisycl::buffer< T, Dimensions, Allocator >::buffer (
            cl_mem mem_object,
            queue from_queue,
            event available_event = {},
            Allocator allocator = {} )  [inline]
```

Create a buffer from an existing OpenCL memory object associated with a context after waiting for an event signaling the availability of the OpenCL data.

Parameters

in,out	*mem_object*	is the OpenCL memory object to use
in,out	*from_queue*	is the queue associated to the memory object
in	*available_event*	specifies the event to wait for if non null

Note that a buffer created from a cl_mem object will only have one underlying cl_mem for the lifetime of the buffer and use on an incompatible queue constitues an error.

Todo To be implemented

Todo Improve the specification to allow CLHPP objects too

Definition at line 353 of file buffer.hpp.

```
00355                                {},
00356          Allocator allocator = {}) { TRISYCL_UNIMPL;  }
```

9.6.2.5.3 Member Function Documentation

9.6.2.5.3.1 get_access() [1/2]

```
template<typename T , int Dimensions = 1, typename Allocator =
buffer_allocator<std::remove_const_t<T>>>
template<access::mode Mode>
accessor< T, Dimensions, Mode, access::target::host_buffer > trisycl::buffer< T, Dimensions,
Allocator >::get_access ( )  [inline]
```

Get a host accessor to the buffer with the required mode.

Parameters

Mode	is the requested access mode

Todo Implement the modes

Todo More elegant solution

Definition at line 408 of file buffer.hpp.

```
00408                {
00409     implementation->implementation->template track_access_mode<Mode, access::target::host_buffer>();
00410     return { *this };
00411   }
```

9.6.2.5.3.2 get_access() [2/2]

```
template<typename T , int Dimensions = 1, typename Allocator =
buffer_allocator<std::remove_const_t<T>>>
template<access::mode Mode, access::target Target = access::target::global_buffer>
accessor< T, Dimensions, Mode, Target > trisycl::buffer< T, Dimensions, Allocator >::get_↩
access (
             handler & command_group_handler )  [inline]
```

Get an accessor to the buffer with the required mode.

Parameters

	Mode	is the requested access mode
	Target	is the type of object to be accessed
in	*command_group_handler*	is the command group handler in which the kernel is to be executed

Todo Do we need for an accessor to increase the reference count of a buffer object? It does make more sense for a host-side accessor.

Todo Implement the modes and targets

Definition at line 379 of file buffer.hpp.

```
00379                                          {
00380     static_assert(Target == access::target::global_buffer
00381                   || Target == access::target::constant_buffer,
```

```
00382                   "get_access(handler) can only deal with access::global_buffer"
00383                   " or access::constant_buffer (for host_buffer accessor"
00384                   " do not use a command group handler");
00385     implementation->implementation->template track_access_mode<Mode, Target>();
00386     return { *this, command_group_handler };
00387   }
```

References trisycl::access::constant_buffer, and trisycl::access::global_buffer.

9.6.2.5.3.3 get_count() template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>

auto trisycl::buffer< T, Dimensions, Allocator >::get_count ( ) const [inline]

Returns the total number of elements in the buffer.

Equal to get_range()[0] ∗ ... ∗ get_range()[Dimensions-1].

Definition at line 435 of file buffer.hpp.

```
00435                           {
00436     return implementation->implementation->get_count();
00437   }
```

9.6.2.5.3.4 get_range() template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>

auto trisycl::buffer< T, Dimensions, Allocator >::get_range ( ) const [inline]

Return a range object representing the size of the buffer in terms of number of elements in each dimension as passed to the constructor.

Todo rename to the equivalent from array_ref proposals? Such as size() in http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0009r2.html

Definition at line 422 of file buffer.hpp.

```
00422                           {
00423     /* Interpret the shape which is a pointer to the first element as an
00424 array of Dimensions elements so that the range<Dimensions>
00425 constructor is happy with this collection
00426 */
00427     return implementation->implementation->get_range();
00428   }
```

9.6.2.5.3.5 get_size() template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>

size_t trisycl::buffer< T, Dimensions, Allocator >::get_size ( ) const [inline]

Returns the size of the buffer storage in bytes.

Equal to get_count()∗sizeof(T).

Todo rename to something else. In http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/p0122r0.pdf it is named bytes() for example

Definition at line 448 of file buffer.hpp.

```
00448                           {
00449     return implementation->implementation->get_size();
00450   }
```

9.6.2.5.3.6 is_cached()

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
bool trisycl::buffer< T, Dimensions, Allocator >::is_cached (
            trisycl::context & ctx )  [inline]
```

Check if the buffer is already cached in a certain context.

Definition at line 513 of file buffer.hpp.

```
00513                                    {
00514     return implementation->implementation->is_cached(ctx);
00515   }
```

9.6.2.5.3.7 is_data_up_to_date()

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
bool trisycl::buffer< T, Dimensions, Allocator >::is_data_up_to_date (
            trisycl::context & ctx )  [inline]
```

Check if the data stored in the buffer is up-to-date in a certain context.

Definition at line 520 of file buffer.hpp.

```
00520                                              {
00521     return implementation->implementation->is_data_up_to_date(ctx);
00522   }
```

9.6.2.5.3.8 is_read_only()

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
bool constexpr trisycl::buffer< T, Dimensions, Allocator >::is_read_only ( ) const  [inline], [constexpr]
```

Ask for read-only status of the buffer.

Todo Add to specification

Definition at line 475 of file buffer.hpp.

```
00475                                    {
00476     return std::is_const<T>::value;
00477   }
```

9.6.2.5.3.9 mark_as_written()

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
void trisycl::buffer< T, Dimensions, Allocator >::mark_as_written ( )  [inline]
```

Force the buffer to behave like if we had created an accessor in write mode.

Definition at line 393 of file buffer.hpp.

```
00393                              {
00394     return implementation->implementation->mark_as_written();
00395   }
```

9.6.2.5.3.10 rank() `template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>`

`static auto constexpr trisycl::buffer< T, Dimensions, Allocator >::rank ( ) [inline], [static], [constexpr]`

Get the number of dimensions of the buffer.

Name inspired from ISO C++ P0009 mdspan papers

Definition at line 82 of file buffer.hpp.

```
00082 { return Dimensions; }
```

9.6.2.5.3.11 set_final_data() [1/3] `template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>`

`template<typename Iterator , typename ValueType = typename std::iterator_traits< std::remove_reference_t<Iterator>>::value_type>`

`void trisycl::buffer< T, Dimensions, Allocator >::set_final_data ( Iterator && finalData ) [inline]`

Set destination of buffer data on destruction.

WARNING: the user has to ensure that the object referred to by the iterator will be alive after buffer destruction, otherwise the behavior is undefined.

Definition at line 535 of file buffer.hpp.

```
00535                                                {
00536     implementation->implementation->set_final_data(
00537       std::forward<Iterator>(finalData));
00538   }
```

9.6.2.5.3.12 set_final_data() [2/3] `template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>`

`void trisycl::buffer< T, Dimensions, Allocator >::set_final_data ( std::nullptr_t ) [inline]`

Disable write-back on buffer destruction.

Definition at line 505 of file buffer.hpp.

```
00505                                      {
00506     implementation->implementation->set_final_data(nullptr);
00507   }
```

9.6.2.5.3.13 set_final_data() [3/3]

```
template<typename T , int Dimensions = 1, typename Allocator
= buffer_allocator<std::remove_const_t<T>>>
void trisycl::buffer< T, Dimensions, Allocator >::set_final_data (
            weak_ptr_class< T > finalData )  [inline]
```

Set destination of buffer data on destruction.

The finalData points to the host memory to which, the outcome of all the buffer processing is going to be copied to.

This is the final pointer, which is going to be accessible after the destruction of the buffer and in the case where this is a valid pointer, the data are going to be copied to this host address.

finalData is different from the original host address, if the buffer was created associated with one. This is mainly to be used when a shared_ptr is given in the constructor and the output data will reside in a different location from the initialization data.

It is defined as a weak_ptr referring to a shared_ptr that is not associated with the trisycl::buffer, and so the trisycl::buffer will have no ownership of finalData.

Definition at line 498 of file buffer.hpp.

```
00498                                                {
00499     implementation->implementation->set_final_data(std::move(finalData));
00500   }
```

9.6.2.5.3.14 use_count()

```
template<typename T , int Dimensions = 1, typename Allocator = buffer↩
_allocator<std::remove_const_t<T>>>
auto trisycl::buffer< T, Dimensions, Allocator >::use_count ( ) const  [inline]
```

Returns the number of buffers that are shared/referenced.

For example

```
trisycl::buffer<int> b { 1000 };
// Here b.use_count() should return 1
trisycl::buffer<int> c { b };
// Here b.use_count() and b.use_count() should return 2
```

Todo Add to the specification, useful for validation

Definition at line 465 of file buffer.hpp.

```
00465                                {
00466     // Rely on the shared_ptr<> use_count()
00467     return implementation.use_count();
00468   }
```

9.6.2.5.4 Member Data Documentation

9.6.2.5.4.1 implementation_t

```
template<typename T , int Dimensions = 1, typename Allocator =
buffer_allocator<std::remove_const_t<T>>>
friend trisycl::buffer< T, Dimensions, Allocator >::implementation_t  [private]
```

Definition at line 90 of file buffer.hpp.

9.6.2.6 class trisycl::detail::accessor

template<typename T, int Dimensions, access::mode Mode, access::target Target>
class trisycl::detail::accessor< T, Dimensions, Mode, Target >

The buffer accessor abstracts the way buffer data are accessed inside a kernel in a multidimensional variable length array way.

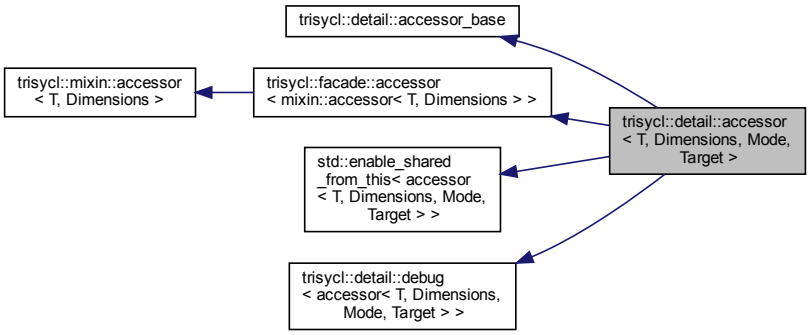
This implementation relies on std::mdspan to provide this nice syntax and behavior.

Right now the aim of this class is just to access to the buffer in a read-write mode, even if capturing the accessor from a lambda make it const (since in examples we have lambda with [=] without mutable lambda).

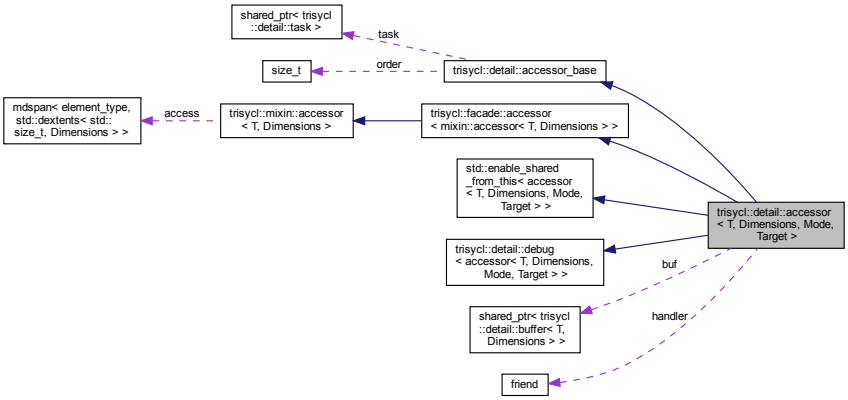
Todo Use the access::mode

Definition at line 55 of file accessor.hpp.

Inheritance diagram for trisycl::detail::accessor< T, Dimensions, Mode, Target >:



Collaboration diagram for trisycl::detail::accessor< T, Dimensions, Mode, Target >:



Public Member Functions

- accessor ()=default

  Used by the local accessor hack on top of host accessor.
- accessor (std::shared_ptr< detail::buffer< T, Dimensions > > target_buffer)

  Construct a host accessor from an existing buffer.
- accessor (std::shared_ptr< detail::buffer< T, Dimensions > > target_buffer, handler &command_group_↩ handler)

  Construct a device accessor from an existing buffer.
- void register_accessor ()

  Register the accessor once a `std::shared_ptr` is created on it.
- detail::buffer< T, Dimensions > & get_buffer ()

  Get the buffer used to create the accessor.
- constexpr bool is_read_access () const

  Test if the accessor has a read access right.
- constexpr bool is_write_access () const

  Test if the accessor has a write access right.

Protected Member Functions

- void set_buffer (std::shared_ptr< detail::buffer< T, Dimensions > > b)

  Set later the current buffer associated to this accessor.

Private Types

- using facade = facade::accessor< mixin::accessor< T, Dimensions > >

  Where most of the user-facing interface dwells.

Private Member Functions

- boost::compute::buffer get_cl_buffer () const override

  Get the boost::compute::buffer or throw if unset.
- void copy_in_cl_buffer ()

  Lazily associate a CL buffer to the SYCL buffer and copy data in it if required, updates the state of the data in the buffer across contexts.
- void copy_back_cl_buffer ()

  Does nothing.

Private Attributes

- std::shared_ptr< detail::buffer< T, Dimensions > > buf

  Keep a reference to the accessed buffer.
- friend handler

Additional Inherited Members

9.6.2.6.1 Member Typedef Documentation

9.6.2.6.1.1 facade

```
template<typename T , int Dimensions, access::mode Mode, access::target
Target>
using trisycl::detail::accessor< T, Dimensions, Mode, Target >::facade = facade::accessor<mixin::accessor<T,
Dimensions> >  [private]
```

Where most of the user-facing interface dwells.

Definition at line 69 of file accessor.hpp.

9.6.2.6.2 Constructor & Destructor Documentation

9.6.2.6.2.1 accessor() [1/3]

```
template<typename T , int Dimensions, access::mode Mode, access::target
Target>
trisycl::detail::accessor< T, Dimensions, Mode, Target >::accessor ( )  [default]
```

Used by the local accessor hack on top of host accessor.

9.6.2.6.2.2 accessor() [2/3]

```
template<typename T , int Dimensions, access::mode Mode, access::target
Target>
trisycl::detail::accessor< T, Dimensions, Mode, Target >::accessor (
            std::shared_ptr< detail::buffer< T, Dimensions > > target_buffer )  [inline]
```

Construct a host accessor from an existing buffer.

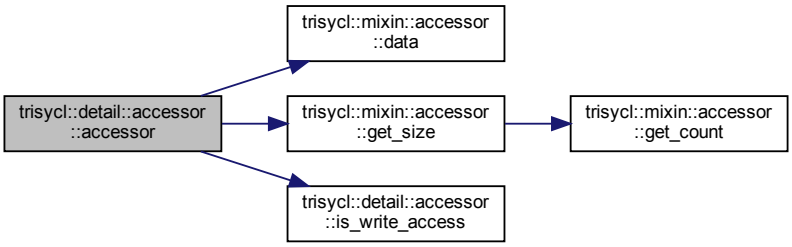
Todo fix the specification to rename target that shadows template parm

Definition at line 91 of file accessor.hpp.

```
00092       :  facade { target_buffer->access }
00093       , buf { target_buffer } {
00094     target_buffer->template track_access_mode<Mode>();
00095     TRISYCL_DUMP_T("Create a host accessor write = " « is_write_access());
00096     static_assert(Target == access::target::host_buffer,
00097                   "without a handler, access target should be host_buffer");
00098     /* The host needs to wait for all the producers of the buffer to
00099 have finished */
00100     buf->wait();
00101
00102 #ifdef TRISYCL_OPENCL
00103     /* For the host context, we are obligated to update the buffer state
00104 during the accessors creation, otherwise we have no way of knowing
00105 if a buffer was modified on the host.  This is only true because
00106 host accessors are blocking
00107 */
00108     trisycl::context ctx;
00109     buf->update_buffer_state(ctx, Mode, facade::get_size(), facade::data());
00110 #endif
00111   }
```

References trisycl::detail::accessor< T, Dimensions, Mode, Target >::buf, trisycl::mixin::accessor< T, Dimensions >::data(), trisycl::mixin::accessor< T, Dimensions >::get_size(), trisycl::access::host_buffer, trisycl::detail::accessor< T, Dimensions, Mode, Ta and TRISYCL_DUMP_T.

Here is the call graph for this function:



9.6.2.6.2.3 accessor() [3/3]

```
template<typename T , int Dimensions, access::mode Mode, access::target
Target>
trisycl::detail::accessor< T, Dimensions, Mode, Target >::accessor (
            std::shared_ptr< detail::buffer< T, Dimensions > > target_buffer,
            handler & command_group_handler )  [inline]
```

Construct a device accessor from an existing buffer.

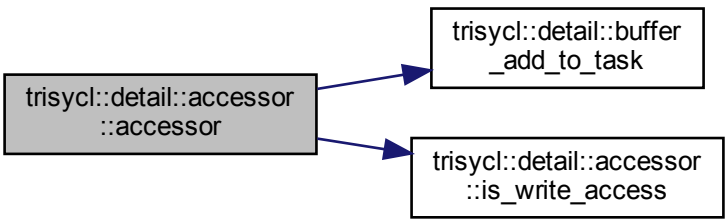
Todo fix the specification to rename target that shadows template parm

Definition at line 118 of file accessor.hpp.

```
00120       :  facade { target_buffer->access }
00121       , buf { target_buffer } {
00122     target_buffer->template track_access_mode<Mode>();
00123     TRISYCL_DUMP_T("Create a kernel accessor write = " « is_write_access());
00124     static_assert(Target == access::target::global_buffer ||
00125                   Target == access::target::constant_buffer,
00126                   "access target should be global_buffer or constant_buffer "
00127                   "when a handler is used");
00128     // Register the buffer to the task dependencies
00129     task = buffer_add_to_task(buf, &command_group_handler, is_write_access());
00130   }
```

References trisycl::detail::accessor< T, Dimensions, Mode, Target >::buf, trisycl::detail::buffer_add_to_task(), trisycl::access::constant_buffer, trisycl::access::global_buffer, trisycl::detail::accessor< T, Dimensions, Mode, Target >::is_write_acce and TRISYCL_DUMP_T.

Here is the call graph for this function:



9.6.2.6.3 Member Function Documentation

9.6.2.6.3.1 copy_back_cl_buffer() template<typename T , int Dimensions, access::mode Mode, access::target Target>

void trisycl::detail::accessor< T, Dimensions, Mode, Target >::copy_back_cl_buffer ( ) [inline], [private]

Does nothing.

Definition at line 217 of file accessor.hpp.

```
00217                                {
00218     /* The copy back is handled by the host accessor and the buffer destructor.
00219 We don't need to systematically transfer the data after the
00220 kernel execution
00221
00222 \todo Figure out what to do with this function
00223 */
00224   }
```

9.6.2.6.3.2 copy_in_cl_buffer() template<typename T , int Dimensions, access::mode Mode, access::target Target>

void trisycl::detail::accessor< T, Dimensions, Mode, Target >::copy_in_cl_buffer ( ) [inline], [private]

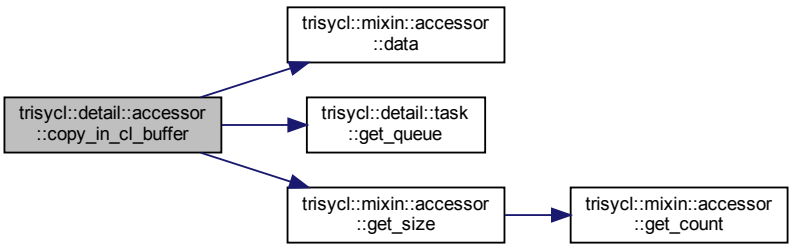
Lazily associate a CL buffer to the SYCL buffer and copy data in it if required, updates the state of the data in the buffer across contexts.

Definition at line 208 of file accessor.hpp.

```
00208                               {
00209     /* Create the OpenCL buffer and copy in it the data from the host if
00210 the buffer doesn't already exists or if the data is not up to date
00211 */
00212     auto ctx = task->get_queue()->get_context();
00213     buf->update_buffer_state(ctx, Mode, facade::get_size(), facade::data());
00214   }
```

References trisycl::detail::accessor< T, Dimensions, Mode, Target >::buf, trisycl::mixin::accessor< T, Dimensions >::data(), trisycl::detail::task::get_queue(), and trisycl::mixin::accessor< T, Dimensions >::get_size().

Here is the call graph for this function:



9.6.2.6.3.3 get_buffer()

`template<typename T , int Dimensions, access::mode Mode, access::target Target>`
`detail::buffer< T, Dimensions > & trisycl::detail::accessor< T, Dimensions, Mode, Target >::get_buffer ( ) [inline]`

Get the buffer used to create the accessor.

Definition at line 164 of file accessor.hpp.

```
00164 { return *buf; }
```

References trisycl::detail::accessor< T, Dimensions, Mode, Target >::buf.

9.6.2.6.3.4 get_cl_buffer()

`template<typename T , int Dimensions, access::mode Mode, access::target Target>`
`boost::compute::buffer trisycl::detail::accessor< T, Dimensions, Mode, Target >::get_cl_buffer ( ) const [inline], [override], [private], [virtual]`

Get the boost::compute::buffer or throw if unset.

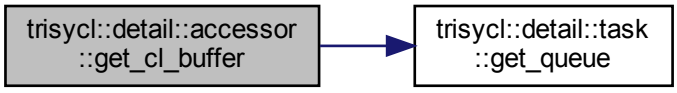
Implements trisycl::detail::accessor_base.

Definition at line 199 of file accessor.hpp.

```
00199                                                    {
00200     // This throws if not set
00201     auto ctx = task->get_queue()->get_context();
00202     return buf->get_cl_buffer(ctx);
00203   }
```

References trisycl::detail::accessor< T, Dimensions, Mode, Target >::buf, and trisycl::detail::task::get_queue().

Here is the call graph for this function:



9.6.2.6.3.5 is_read_access()

`template<typename T , int Dimensions, access::mode Mode, access::target Target>`
`constexpr bool trisycl::detail::accessor< T, Dimensions, Mode, Target >::is_read_access ( ) const [inline], [constexpr]`

Test if the accessor has a read access right.

Todo Strangely, it is not really constexpr because it is not a static method...

Todo to move in the access::mode enum class and add to the specification ?

Definition at line 174 of file accessor.hpp.

```
00174                                        {
00175     return Mode == access::mode::read || Mode == access::mode::read_write ||
00176            Mode == access::mode::discard_read_write;
00177   }
```

References trisycl::access::discard_read_write, trisycl::access::read, and trisycl::access::read_write.

9.6.2.6.3.6 is_write_access() template<typename T , int Dimensions, access::mode Mode, access::target Target>

constexpr bool trisycl::detail::accessor< T, Dimensions, Mode, Target >::is_write_access ( ) const [inline], [constexpr]

Test if the accessor has a write access right.

Todo Strangely, it is not really constexpr because it is not a static method...

Todo to move in the access::mode enum class and add to the specification ?

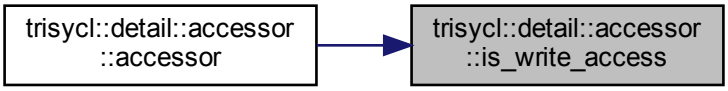
Definition at line 187 of file accessor.hpp.

```
00187                                       {
00188     return Mode == access::mode::write || Mode == access::mode::read_write ||
00189            Mode == access::mode::discard_write ||
00190            Mode == access::mode::discard_read_write;
00191   }
```

References trisycl::access::discard_read_write, trisycl::access::discard_write, trisycl::access::read_write, and trisycl::access::write.

Referenced by trisycl::detail::accessor< T, Dimensions, Mode, Target >::accessor().

Here is the caller graph for this function:



9.6.2.6.3.7 register_accessor() template<typename T , int Dimensions, access::mode Mode, access::target Target>

void trisycl::detail::accessor< T, Dimensions, Mode, Target >::register_accessor ( ) [inline]

Register the accessor once a `std::shared_ptr` is created on it.

This is to be called from outside once the object is created. It has been tried directly inside the contructor, but calling `shared_from_this()` from the constructor dead-lock with libstdc++6

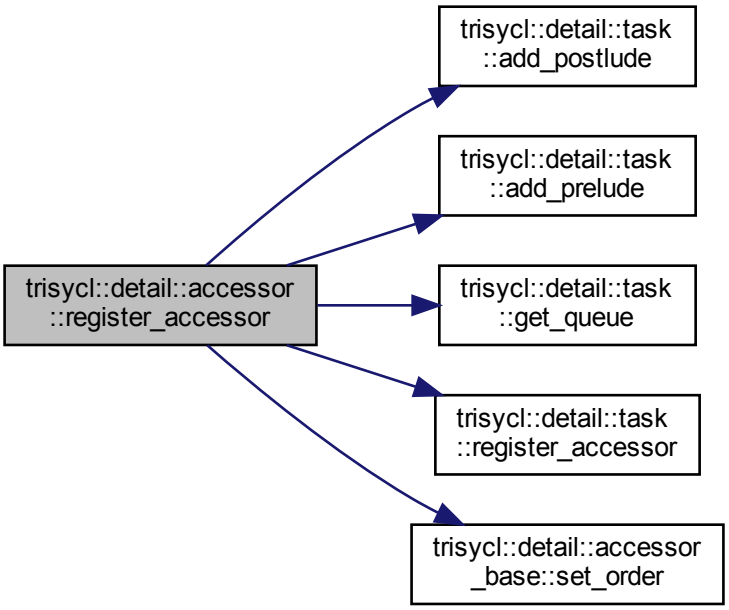
Todo Double-check with the C++ committee on this issue.

Definition at line 141 of file accessor.hpp.

```
00141                           {
00142     if (!task->get_queue()->is_host()) {
00143       // To keep alive this accessor in the following lambdas
00144       auto acc = this->shared_from_this();
00145       // Attach the accessor to the task and get its order
00146       set_order(task->register_accessor(acc));
00147 #ifdef TRISYCL_OPENCL
00148       /* Before running the kernel, make sure the cl_mem behind this
00149 accessor is up-to-date on the device if needed and pass it to
00150 the kernel */
00151       task->add_prelude([=] { acc->copy_in_cl_buffer(); });
00152       // After running the kernel, deal with some copy-back if needed
00153       task->add_postlude([=] {
00154         /* Even if this function does nothing, it is required to
00155 have the capture of acc to keep the accessor alive across
00156 the kernel execution up to the execution postlude */
00157         acc->copy_back_cl_buffer();
00158       });
00159 #endif
00160     }
00161   }
```

References trisycl::detail::task::add_postlude(), trisycl::detail::task::add_prelude(), trisycl::detail::task::get_queue(), trisycl::detail::task::register_accessor(), and trisycl::detail::accessor_base::set_order().

Here is the call graph for this function:



9.6.2.6.3.8 set_buffer()

```
template<typename T , int Dimensions, access::mode Mode, access::target
Target>
void trisycl::detail::accessor< T, Dimensions, Mode, Target >::set_buffer (
            std::shared_ptr< detail::buffer< T, Dimensions > > b )  [inline], [protected]
```

Set later the current buffer associated to this accessor.

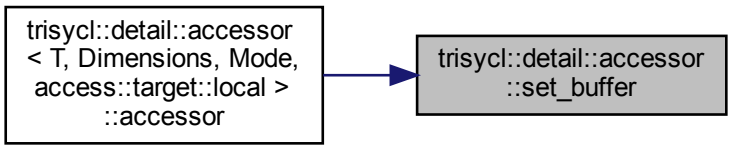
Definition at line 229 of file accessor.hpp.

```
00229                                                                {
00230     buf = b;
00231   }
```

References b, and trisycl::detail::accessor< T, Dimensions, Mode, Target >::buf.

Referenced by trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >::accessor().

Here is the caller graph for this function:



9.6.2.6.4 Member Data Documentation

9.6.2.6.4.1 buf `template<typename T , int Dimensions, access::mode Mode, access::target Target> std::shared_ptr<detail::buffer<T, Dimensions> > trisycl::detail::accessor< T, Dimensions, Mode, Target >::buf [private]`

Keep a reference to the accessed buffer.

Beware that it owns the buffer, which means that the accessor has to be destroyed to release the buffer and potentially unblock a kernel at the end of its execution

Definition at line 66 of file accessor.hpp.

Referenced by trisycl::detail::accessor< T, Dimensions, Mode, Target >::accessor(), trisycl::detail::accessor< T, Dimensions, Mode, T
trisycl::detail::accessor< T, Dimensions, Mode, Target >::get_buffer(), trisycl::detail::accessor< T, Dimensions, Mode, Target >::get_
and trisycl::detail::accessor< T, Dimensions, Mode, Target >::set_buffer().

9.6.2.6.4.2 handler `template<typename T , int Dimensions, access::mode Mode, access::target Target> friend trisycl::detail::accessor< T, Dimensions, Mode, Target >::handler [private]`

Definition at line 196 of file accessor.hpp.

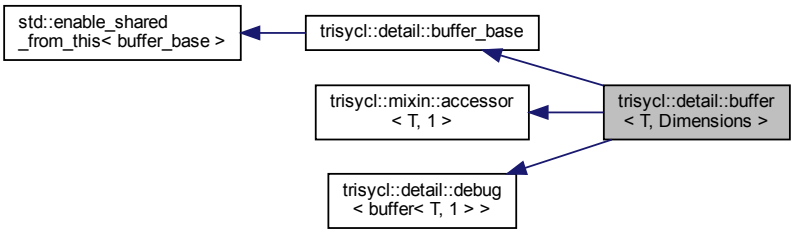
9.6.2.7 class trisycl::detail::buffer

template<typename T, int Dimensions = 1>
class trisycl::detail::buffer< T, Dimensions >

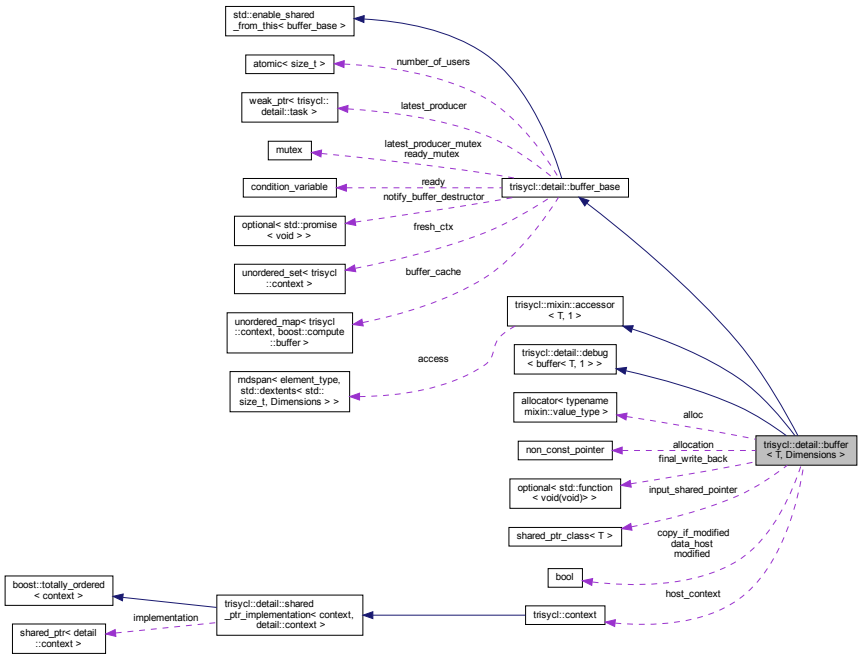
A SYCL buffer is a multidimensional variable length array (à la C99 VLA or even Fortran before) that is used to store data to work on.

Definition at line 37 of file buffer.hpp.

Inheritance diagram for trisycl::detail::buffer< T, Dimensions >:



Collaboration diagram for trisycl::detail::buffer< T, Dimensions >:



Public Member Functions

- buffer (const range< Dimensions > &r)

  Create a new read-write buffer of size.
- buffer (T *host_data, const range< Dimensions > &r)

  Create a new read-write buffer from.
- template<typename Dependent = T, typename = std::enable_if_t<!std::is_const<Dependent>::value>>
  buffer (const T *host_data, const range< Dimensions > &r)

  Create a new read-only buffer from.
- buffer (shared_ptr_class< T > &host_data, const range< Dimensions > &r)

  Create a new buffer with associated memory, using the data in host_data.
- template<typename Iterator >
  buffer (Iterator start_iterator, Iterator end_iterator)

  Create a new allocated 1D buffer from the given elements.
- ~buffer ()

  Create a new sub-buffer without allocation to have separate accessors later.
- void mark_as_written ()

  Enforce the buffer to be considered as being modified.
- template<access::mode Mode, access::target Target = access::target::host_buffer>
  void track_access_mode ()

  This method is to be called whenever an accessor is created.
- void set_final_data (std::weak_ptr< T > final_data)

  Set the weak pointer as destination for write-back on buffer destruction.
- void set_final_data (std::nullptr_t)

  Disable write-back on buffer destruction as an iterator.
- template<typename Iterator , typename ValueType = typename std::iterator_traits<Iterator>::value_type>
  void set_final_data (Iterator final_data)

  Provide destination for write-back on buffer destruction as an iterator.
- boost::optional< std::future< void > > get_destructor_future ()

  Get a `future` to wait from inside the `trisycl::buffer` in case there is something to copy back to the host.

Private Types

- using mixin = mixin::accessor< T, Dimensions >

Private Member Functions

- auto allocate_buffer (const range< Dimensions > &r)

  Allocate uninitialized buffer memory.
- void deallocate_buffer ()

  Deallocate buffer memory if required.
- template<typename StartIter , typename EndIter >
  void assign (StartIter start_iterator, EndIter end_iterator)

  Assign the 1-D storage behind the accessor.
- template<typename BaseType = T, typename DataType >
  void call_update_buffer_state (trisycl::context ctx, access::mode mode, size_t size, DataType *data, std::enable_if_t<!std::is_const< BaseType >::value > *=0)

  Function pair to work around the fact that T might be a `const` type.
- template<typename BaseType = T, typename DataType >
  void call_update_buffer_state (trisycl::context ctx, access::mode mode, size_t size, DataType *data, std::enable_if_t< std::is_const< BaseType >::value > *=0)

  Version of `call_update_buffer_state` that does nothing.

Private Attributes

- std::allocator< typename mixin::value_type > alloc

  The allocator to be used when some memory is needed.
- mixin::non_const_pointer allocation = nullptr

  If some allocation is requested on the host for the buffer memory, this is where the memory is attached to.
- boost::optional< std::function< void(void)> > final_write_back

  How to copy back data on buffer destruction, can be modified with set_final_data(...)
- shared_ptr_class< T > input_shared_pointer

  Keep the shared pointer used to create the buffer.
- bool data_host = false

  Track if the buffer memory is backed by user-provided host memory.
- bool copy_if_modified = false

  Track if data should be copied into buffer-owned memory if a modification occurs.
- bool modified = false
- trisycl::context host_context { trisycl::device {} }

  Track the host context.

Friends

- template<typename U , int D, access::mode Mode, access::target Target>
  class detail::accessor

Additional Inherited Members

9.6.2.7.1 Member Typedef Documentation

9.6.2.7.1.1 mixin `template<typename T , int Dimensions = 1>`

```
using trisycl::detail::buffer< T, Dimensions >::mixin = mixin::accessor<T, Dimensions>  [private]
```

Definition at line 43 of file buffer.hpp.

9.6.2.7.2 Constructor & Destructor Documentation

9.6.2.7.2.1 buffer() `[1/5]` `template<typename T , int Dimensions = 1>`

```
trisycl::detail::buffer< T, Dimensions >::buffer (
            const range< Dimensions > & r )  [inline]
```

Create a new read-write buffer of size.

Parameters

r	

Todo Lazily allocate memory since it might not be used on host

Definition at line 88 of file buffer.hpp.

```
00090       :  mixin { allocate_buffer(r), r } {}
```

9.6.2.7.2.2 buffer() [2/5] template<typename T , int Dimensions = 1>

```
trisycl::detail::buffer< T, Dimensions >::buffer (
              T * host_data,
              const range< Dimensions > & r )  [inline]
```

Create a new read-write buffer from.

Parameters

host_data	of size
r	without further allocation

Definition at line 94 of file buffer.hpp.

```
00095       :  mixin { host_data, r }
00096       , data_host { true } {}
```

9.6.2.7.2.3 buffer() [3/5] template<typename T , int Dimensions = 1>

```
template<typename Dependent = T, typename = std::enable_if_t<!std::is_const<Dependent>↩
::value>>
trisycl::detail::buffer< T, Dimensions >::buffer (
              const T * host_data,
              const range< Dimensions > & r )  [inline]
```

Create a new read-only buffer from.

Parameters

host_data	of size
r	without further allocation

If the buffer is non const, use a copy-on-write mechanism with internal writable memory.

Todo Clarify the semantics in the spec. What happens if the host change the host_data after buffer creation?

Only enable this constructor if the value type is not constant, because if it is constant, the buffer is constant too.

Definition at line 112 of file buffer.hpp.

```
00113       :  /* The buffer is read-only, even if the internal multidimensional
00114 wrapper is not.  If a write accessor is requested, there should
00115 be a copy on write.  So this pointer should not be written and
00116 this const_cast should be acceptable.  */
00117       mixin { const_cast<T*>(host_data), r }
00118       , data_host { true }
00119       ,
00120       /* Set copy_if_modified to true, so that if an accessor with write
00121 access is created, data are copied before to be modified.  */
00122       copy_if_modified { true } {}
```

9.6.2.7.2.4 buffer() [4/5] template<typename T , int Dimensions = 1>

```
trisycl::detail::buffer< T, Dimensions >::buffer (
            shared_ptr_class< T > & host_data,
            const range< Dimensions > & r )  [inline]
```

Create a new buffer with associated memory, using the data in host_data.

The ownership of the host_data is shared between the runtime and the user. In order to enable both the user application and the SYCL runtime to use the same pointer, a trisycl::mutex_class is used.

Definition at line 132 of file buffer.hpp.

```
00133       :  mixin { host_data.get(), r }
00134       , input_shared_pointer { host_data }
00135       , data_host { true } {}
```

9.6.2.7.2.5 buffer() [5/5] template<typename T , int Dimensions = 1>

```
template<typename Iterator >
trisycl::detail::buffer< T, Dimensions >::buffer (
            Iterator start_iterator,
            Iterator end_iterator )  [inline]
```

Create a new allocated 1D buffer from the given elements.

Definition at line 139 of file buffer.hpp.

```
00140       :  mixin { allocate_buffer(std::distance(start_iterator, end_iterator)),
00141                  range<1> { static_cast<std::size_t>(
00142                      std::distance(start_iterator, end_iterator)) } } {
00143     assign(start_iterator, end_iterator);
00144   }
```

9.6.2.7.2.6 ~buffer() template<typename T , int Dimensions = 1>

```
trisycl::detail::buffer< T, Dimensions >::~buffer ( )  [inline]
```

Create a new sub-buffer without allocation to have separate accessors later.

Todo To implement and deal with reference counting buffer(buffer<T, Dimensions> b, index<Dimensions> base↩ _index, range<Dimensions> sub_range)

Todo Allow CLHPP objects too?

The buffer content may be copied back on destruction to some final location

Definition at line 165 of file buffer.hpp.

```
00165             {
00166 #ifdef TRISYCL_OPENCL
00167     /* We ensure that the host has the most up-to-date version of the data
00168 before the buffer is destroyed.  This is necessary because we do not
00169 systematically transfer the data back from a device with
00170 \c copy_back_cl_buffer any more.
00171 \todo Optimize for the case the buffer is not based on host memory
00172 */
00173     call_update_buffer_state(host_context, access::mode::read,
00174                              mixin::get_size(), mixin::data());
00175
00176 #endif
00177     if (modified && final_write_back)
00178       (*final_write_back)();
00179     // Allocate explicitly allocated memory if required
00180     deallocate_buffer();
00181   }
```

References trisycl::access::read.

9.6.2.7.3 Member Function Documentation

9.6.2.7.3.1 allocate_buffer() template<typename T , int Dimensions = 1>

```
auto trisycl::detail::buffer< T, Dimensions >::allocate_buffer (
            const range< Dimensions > & r )  [inline], [private]
```

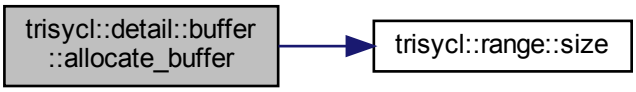
Allocate uninitialized buffer memory.

Definition at line 259 of file buffer.hpp.

```
00259                                                {
00260     auto count = r.size();
00261     // Allocate uninitialized memory
00262     allocation = alloc.allocate(count);
00263     return allocation;
00264   }
```

References trisycl::range< Dimensions >::size().

Here is the call graph for this function:



9.6.2.7.3.2 assign() template<typename T , int Dimensions = 1>

```
template<typename StartIter , typename EndIter >
void trisycl::detail::buffer< T, Dimensions >::assign (
            StartIter start_iterator,
            EndIter end_iterator )  [inline], [private]
```

Assign the 1-D storage behind the accessor.

Use 2 different iterator types since in C++20 ranges it is now the case.

Definition at line 278 of file buffer.hpp.

```
00278                                                    {
00279     std::copy(start_iterator, end_iterator, mixin::data());
00280   }
```

9.6.2.7.3.3 call_update_buffer_state() [1/2] template<typename T , int Dimensions = 1>

```
template<typename BaseType = T, typename DataType >
void trisycl::detail::buffer< T, Dimensions >::call_update_buffer_state (
            trisycl::context ctx,
            access::mode mode,
            size_t size,
            DataType * data,
            std::enable_if_t< std::is_const< BaseType >::value > *  = 0 )  [inline], [private]
```

Version of `call_update_buffer_state` that does nothing.

It is called if the type of the data in the buffer is `const`

Definition at line 300 of file buffer.hpp.

```
00302                                                {}
```

9.6.2.7.3.4 call_update_buffer_state() [2/2] template<typename T , int Dimensions = 1>

```
template<typename BaseType = T, typename DataType >
void trisycl::detail::buffer< T, Dimensions >::call_update_buffer_state (
            trisycl::context ctx,
            access::mode mode,
            size_t size,
            DataType * data,
            std::enable_if_t<!std::is_const< BaseType >::value > *  = 0 )  [inline], [private]
```

Function pair to work around the fact that T might be a `const` type.

We call update_buffer_state only if T is not `const`, we have to use `enable_if` otherwise the compiler will try to cast `const void*` to `void*` if we create a buffer with a `const` type

Todo Use `if constexpr` when it is available with C++17

Definition at line 290 of file buffer.hpp.

```
00292                                                {
00293     update_buffer_state(ctx, mode, size, data);
00294   }
```

9.6.2.7.3.5 deallocate_buffer() template<typename T , int Dimensions = 1>

```
void trisycl::detail::buffer< T, Dimensions >::deallocate_buffer ( )  [inline], [private]
```

Deallocate buffer memory if required.

Definition at line 267 of file buffer.hpp.

```
00267                          {
00268     if (allocation)
00269       alloc.deallocate(allocation, mixin::get_count());
00270   }
```

9.6.2.7.3.6 get_destructor_future() template<typename T , int Dimensions = 1> boost::optional< std::future< void > > trisycl::detail::buffer< T, Dimensions >::get_destructor_future ( ) [inline]

Get a `future` to wait from inside the `trisycl::buffer` in case there is something to copy back to the host.

Returns

A `future` in the `optional` if there is something to wait for, otherwise an empty `optional`

Todo Make the function private again

Definition at line 312 of file buffer.hpp.

```
00312                                                   {
00313     /* If there is only 1 shared_ptr user of the buffer, this is the
00314 caller of this function, the \c buffer_waiter, so there is no
00315 need to get a \ future otherwise there will be a dead-lock if
00316 there is only 1 thread waiting for itself.
00317
00318 Since \c use_count() is applied to a \c shared_ptr just created
00319 for this purpose, it actually increase locally the count by 1,
00320 so check for 1 + 1 use count instead...
00321 */
00322     // If the buffer's destruction triggers a write-back, wait
00323     if ((shared_from_this().use_count() > 2) && modified &&
00324         (final_write_back || data_host)) {
00325       // Create a promise to wait for
00326       notify_buffer_destructor = std::promise<void> {};
00327       // And return the future to wait for it
00328       return notify_buffer_destructor->get_future();
00329     }
00330     return boost::none;
00331   }
```

9.6.2.7.3.7 mark_as_written() template<typename T , int Dimensions = 1> void trisycl::detail::buffer< T, Dimensions >::mark_as_written ( ) [inline]

Enforce the buffer to be considered as being modified.

Same as creating an accessor with write access.

Definition at line 186 of file buffer.hpp.

```
00186 { modified = true; }
```

9.6.2.7.3.8 set_final_data() [1/3] template<typename T , int Dimensions = 1> template<typename Iterator , typename ValueType = typename std::iterator_traits<Iterator>::value_type> void trisycl::detail::buffer< T, Dimensions >::set_final_data (Iterator *final_data*) [inline]

Provide destination for write-back on buffer destruction as an iterator.

Definition at line 246 of file buffer.hpp.

```
00246                                        {
00247     /*   using type_ = typename iterator_value_type<Iterator>::value_type;
00248 static_assert(std::is_same<type_, T>::value, "buffer type mismatch");
00249 static_assert(!(std::is_const<type_>::value),
00250 "const iterator is not allowed");*/
00251     // Capture this by reference is enough since the buffer will still exist
00252     final_write_back = [this, final_data = std::move(final_data)] {
00253       std::copy_n(mixin::data(), mixin::get_count(), final_data);
00254     };
00255   }
```

9.6.2.7.3.9 set_final_data() [2/3] template<typename T , int Dimensions = 1>

```
void trisycl::detail::buffer< T, Dimensions >::set_final_data (
            std::nullptr_t  )  [inline]
```

Disable write-back on buffer destruction as an iterator.

Definition at line 238 of file buffer.hpp.

```
00238 { final_write_back = boost::none; }
```

9.6.2.7.3.10 set_final_data() [3/3] template<typename T , int Dimensions = 1>

```
void trisycl::detail::buffer< T, Dimensions >::set_final_data (
            std::weak_ptr< T > final_data )  [inline]
```

Set the weak pointer as destination for write-back on buffer destruction.

Definition at line 227 of file buffer.hpp.

```
00227                                                {
00228     // Capture this by reference is enough since the buffer will still exist
00229     final_write_back = [this, final_data = std::move(final_data)] {
00230       if (auto sptr = final_data.lock()) {
00231         std::copy_n(mixin::data(), mixin::get_count(), sptr.get());
00232       }
00233     };
00234   }
```

9.6.2.7.3.11 track_access_mode() template<typename T , int Dimensions = 1>

```
template<access::mode Mode, access::target Target = access::target::host_buffer>
void trisycl::detail::buffer< T, Dimensions >::track_access_mode ( )  [inline]
```

This method is to be called whenever an accessor is created.

Its current purpose is to track if an accessor with write access is created and acting accordingly.

Definition at line 195 of file buffer.hpp.

```
00195                              {
00196     // test if write access is required
00197     if (Mode == access::mode::write || Mode == access::mode::read_write ||
00198         Mode == access::mode::discard_write ||
00199         Mode == access::mode::discard_read_write ||
00200         Mode == access::mode::atomic) {
00201       modified = true;
00202       if (copy_if_modified) {
00203         // Implement the allocate & copy-on-write optimization
00204         copy_if_modified = false;
00205         // Since the mixin store the geometry, keep a copy before updating
00206         auto current_access = mixin::access;
00207         /* The range is actually computed from \c access itself, so
00208 save it */
00209         auto current_range = mixin::get_range();
00210         allocate_buffer(current_range);
00211         /* Update the mixin accessor to point to the new allocated
00212 memory instead */
00213         mixin::update(allocation, current_range);
00214         // Then copy the read-only data to the new allocated place
00215         std::uninitialized_copy_n(current_access.data_handle(), mixin::get_count(),
00216                                   mixin::data());
00217         /* Now the data of the buffer is no longer backed-up by host
00218 user provided memory */
00219         data_host = false;
00220       }
00221     }
00222   }
```

References trisycl::access::atomic, trisycl::access::discard_read_write, trisycl::access::discard_write, trisycl::access::read_write, and trisycl::access::write.

9.6.2.7.4 Friends And Related Function Documentation

9.6.2.7.4.1 detail::accessor

```
template<typename T , int Dimensions = 1>
template<typename U , int D, access::mode Mode, access::target Target>
friend class detail::accessor  [friend]
```

Definition at line 49 of file buffer.hpp.

9.6.2.7.5 Member Data Documentation

9.6.2.7.5.1 alloc

```
template<typename T , int Dimensions = 1>
std::allocator<typename mixin::value_type> trisycl::detail::buffer< T, Dimensions >::alloc
[private]
```

The allocator to be used when some memory is needed.

Todo Implement user-provided allocator

Definition at line 55 of file buffer.hpp.

9.6.2.7.5.2 allocation

```
template<typename T , int Dimensions = 1>
mixin::non_const_pointer trisycl::detail::buffer< T, Dimensions >::allocation = nullptr  [private]
```

If some allocation is requested on the host for the buffer memory, this is where the memory is attached to.

Note that this is uninitialized memory, as stated in SYCL specification.

Definition at line 63 of file buffer.hpp.

9.6.2.7.5.3 copy_if_modified

```
template<typename T , int Dimensions = 1>
bool trisycl::detail::buffer< T, Dimensions >::copy_if_modified = false  [private]
```

Track if data should be copied into buffer-owned memory if a modification occurs.

Definition at line 78 of file buffer.hpp.

9.6.2.7.5.4 data_host

```
template<typename T , int Dimensions = 1>
bool trisycl::detail::buffer< T, Dimensions >::data_host = false  [private]
```

Track if the buffer memory is backed by user-provided host memory.

Definition at line 74 of file buffer.hpp.

9.6.2.7.5.5 final_write_back

```
template<typename T , int Dimensions = 1>
boost::optional<std::function<void(void)> > trisycl::detail::buffer< T, Dimensions >::final_write_back [private]
```

How to copy back data on buffer destruction, can be modified with set_final_data(...)

Definition at line 68 of file buffer.hpp.

9.6.2.7.5.6 host_context

```
template<typename T , int Dimensions = 1>
trisycl::context trisycl::detail::buffer< T, Dimensions >::host_context { trisycl::device {} } [private]
```

Track the host context.

Definition at line 84 of file buffer.hpp.

9.6.2.7.5.7 input_shared_pointer

```
template<typename T , int Dimensions = 1>
shared_ptr_class<T> trisycl::detail::buffer< T, Dimensions >::input_shared_pointer [private]
```

Keep the shared pointer used to create the buffer.

Definition at line 71 of file buffer.hpp.

9.6.2.7.5.8 modified

```
template<typename T , int Dimensions = 1>
bool trisycl::detail::buffer< T, Dimensions >::modified = false [private]
```

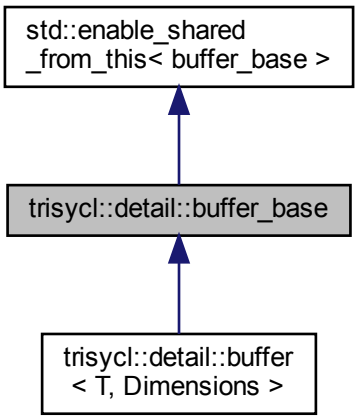
Definition at line 81 of file buffer.hpp.

9.6.2.8 struct trisycl::detail::buffer_base

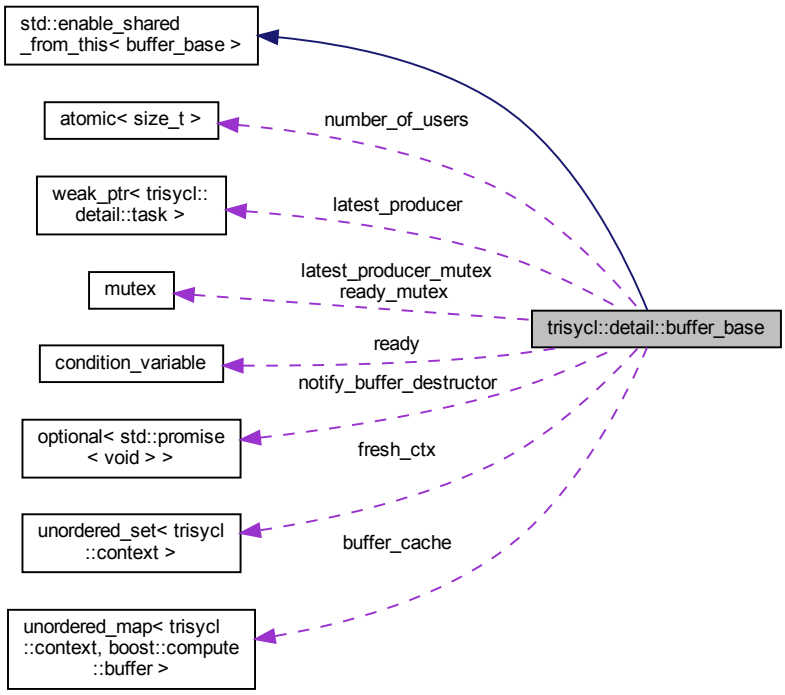
Factorize some template independent buffer aspects in a base class.

Definition at line 49 of file buffer_base.hpp.

Inheritance diagram for trisycl::detail::buffer_base:



Collaboration diagram for trisycl::detail::buffer_base:



Public Member Functions

- buffer_base ()

  Create a buffer base and marks the host context as the context that holds the most recent version of the data.
- ~buffer_base ()

  The destructor waits for not being used anymore.
- void wait ()

  Wait for this buffer to be ready, which is no longer in use.
- void use ()

  Mark this buffer in use by a task.
- void release ()

  A task has released the buffer.
- std::shared_ptr< detail::task > get_latest_producer ()

  Return the latest producer for the buffer.
- std::shared_ptr< detail::task > set_latest_producer (std::weak_ptr< detail::task > newer_latest_producer)

  Return the latest producer for the buffer and set another future producer.
- std::shared_ptr< detail::task > add_to_task (handler *command_group_handler, bool is_write_mode)

  Add a buffer to the task running the command group.
- bool is_data_up_to_date (const trisycl::context &ctx)

  Check if the data of this buffer is up-to-date in a certain context.
- bool is_cached (const trisycl::context &ctx)

  Check if the buffer is already cached for a certain context.
- void create_in_cache (const trisycl::context &ctx, size_t size, cl_mem_flags flags, void *data)

Create a `boost::compute::buffer` for this `trisycl::buffer` in the cache and associate it with a given context.

- void sync_with_host (std::size_t size, void *data)

  Transfer the most up-to-date version of the data to the host if the host version is not already up-to-date.

- void update_buffer_state (const trisycl::context &target_ctx, access::mode mode, std::size_t size, void *data)

  When a transfer is requested this function is called, it will update the state of the buffer according to the context in which the accessor is created and the access mode.

- boost::compute::buffer get_cl_buffer (const trisycl::context &context)

  Returns the cl_buffer for a given context.

Public Attributes

- std::atomic< size_t > number_of_users
- std::weak_ptr< detail::task > latest_producer

  Track the latest task to produce this buffer.

- std::mutex latest_producer_mutex

  To protect the access to latest_producer.

- std::condition_variable ready

  To signal when this buffer ready.

- std::mutex ready_mutex

  To protect the access to the condition variable.

- boost::optional< std::promise< void > > notify_buffer_destructor

  If the SYCL user buffer destructor is blocking, use this to block until this buffer implementation is destroyed.

- std::unordered_set< trisycl::context > fresh_ctx

  To track contexts in which the data is up-to-date.

- std::unordered_map< trisycl::context, boost::compute::buffer > buffer_cache

  Buffer-side cache that keeps the `boost::compute::buffer` (and the underlying `cl_buffer`) so that if the buffer already exists inside the same context it is not recreated.

9.6.2.8.1 Constructor & Destructor Documentation

9.6.2.8.1.1 buffer_base() `trisycl::detail::buffer_base::buffer_base ( ) [inline]`

Create a buffer base and marks the host context as the context that holds the most recent version of the data.

Todo Use lazy allocation for the context tracking set

Definition at line 86 of file buffer_base.hpp.

```
00086                 :  number_of_users { 0 },
00087                   fresh_ctx { trisycl::context {} } {}
```

9.6.2.8.1.2 ~buffer_base() trisycl::detail::buffer_base::~buffer_base ( ) [inline]

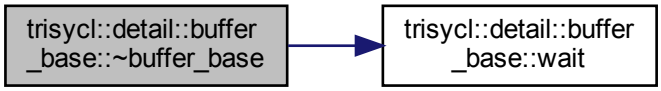
The destructor waits for not being used anymore.

Definition at line 91 of file buffer_base.hpp.

```
00091                 {
00092     wait();
00093     // If there is the last SYCL user buffer waiting, notify it
00094     if (notify_buffer_destructor)
00095       notify_buffer_destructor->set_value();
00096   }
```

References notify_buffer_destructor, and wait().

Here is the call graph for this function:



9.6.2.8.2 Member Function Documentation

9.6.2.8.2.1 add_to_task() std::shared_ptr< detail::task > trisycl::detail::buffer_base::add_↩ to_task (handler * command_group_handler, bool is_write_mode) [inline]

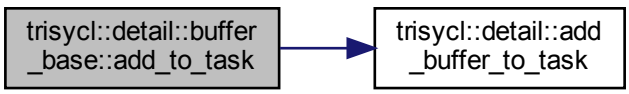
Add a buffer to the task running the command group.

Definition at line 153 of file buffer_base.hpp.

```
00153                                                                    {
00154     return add_buffer_to_task(command_group_handler,
00155                               shared_from_this(),
00156                               is_write_mode);
00157   }
```

References trisycl::detail::add_buffer_to_task().

Here is the call graph for this function:



9.6.2.8.2.2 create_in_cache()

```
void trisycl::detail::buffer_base::create_in_cache (
    const trisycl::context & ctx,
    size_t size,
    cl_mem_flags flags,
    void * data )  [inline]
```

Create a `boost::compute::buffer` for this `trisycl::buffer` in the cache and associate it with a given context.

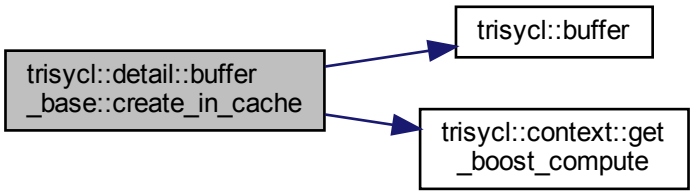
Definition at line 176 of file buffer_base.hpp.

```
00177                                                    {
00178     buffer_cache[ctx] = boost::compute::buffer
00179       { ctx.get_boost_compute(),
00180         size,
00181         flags,
00182         data
00183       };
00184   }
```

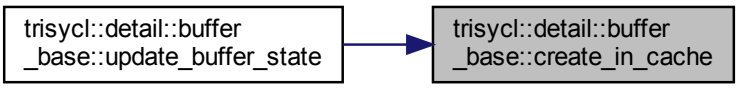
References trisycl::buffer(), buffer_cache, and trisycl::context::get_boost_compute().

Referenced by update_buffer_state().

Here is the call graph for this function:



Here is the caller graph for this function:



9.6.2.8.2.3 get_cl_buffer() boost::compute::buffer trisycl::detail::buffer_base::get_cl_buffer (const trisycl::context & context) [inline]

Returns the cl_buffer for a given context.

Definition at line 312 of file buffer_base.hpp.

```
00312                                                                {
00313     return buffer_cache[context];
00314   }
```

References buffer_cache.

9.6.2.8.2.4 get_latest_producer() std::shared_ptr< detail::task > trisycl::detail::buffer_base::get_latest_producer ( ) [inline]

Return the latest producer for the buffer.

Definition at line 130 of file buffer_base.hpp.

```
00130                                                  {
00131     std::lock_guard<std::mutex> lg { latest_producer_mutex };
00132     // Return the valid shared_ptr to the task, if any
00133     return latest_producer.lock();
00134   }
```

References latest_producer, and latest_producer_mutex.

9.6.2.8.2.5 is_cached() bool trisycl::detail::buffer_base::is_cached (const trisycl::context & ctx) [inline]

Check if the buffer is already cached for a certain context.

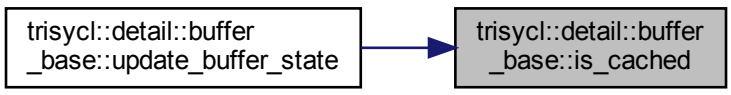
Definition at line 168 of file buffer_base.hpp.

```
00168                                                 {
00169     return buffer_cache.count(ctx);
00170   }
```

References buffer_cache.

Referenced by update_buffer_state().

Here is the caller graph for this function:



9.6.2.8.2.6 is_data_up_to_date() bool trisycl::detail::buffer_base::is_data_up_to_date (const trisycl::context & ctx) [inline]

Check if the data of this buffer is up-to-date in a certain context.

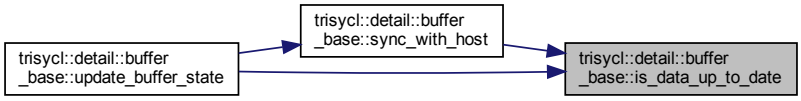
Definition at line 162 of file buffer_base.hpp.

```
00162                                                    {
00163     return fresh_ctx.count(ctx);
00164   }
```

References fresh_ctx.

Referenced by sync_with_host(), and update_buffer_state().

Here is the caller graph for this function:



9.6.2.8.2.7 release() void trisycl::detail::buffer_base::release ( ) [inline]

A task has released the buffer.

Definition at line 117 of file buffer_base.hpp.

```
00117                  {
00118     std::unique_lock<std::mutex> lock { ready_mutex };
00119     if (--number_of_users == 0) {
00120       // Micro-optimization:  unlock before the notification
00121       // https://en.cppreference.com/w/cpp/thread/condition_variable/notify_all
00122       lock.unlock();
00123       // Notify the host consumers or the buffer destructor that it is ready
00124       ready.notify_all();
00125     }
00126   }
```

References number_of_users, ready, and ready_mutex.

9.6.2.8.2.8 set_latest_producer() std::shared_ptr< detail::task > trisycl::detail::buffer_base::set_latest_producer (std::weak_ptr< detail::task > newer_latest_producer) [inline]

Return the latest producer for the buffer and set another future producer.

Definition at line 141 of file buffer_base.hpp.

```
00141                                                                    {
00142     std::lock_guard<std::mutex> lg { latest_producer_mutex };
00143     using std::swap;
00144
00145     swap(newer_latest_producer, latest_producer);
00146     // Return the valid shared_ptr to the previous producing task, if any
00147     return newer_latest_producer.lock();
00148   }
```

References latest_producer, and latest_producer_mutex.

9.6.2.8.2.9 sync_with_host() `void trisycl::detail::buffer_base::sync_with_host ( std::size_t size, void * data ) [inline]`

Transfer the most up-to-date version of the data to the host if the host version is not already up-to-date.

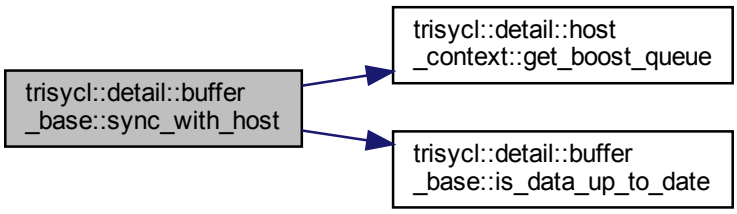
Definition at line 190 of file buffer_base.hpp.

```
00190                                                {
00191     trisycl::context host_context;
00192     if (!is_data_up_to_date(host_context) && !fresh_ctx.empty()) {
00193       /* We know that the context(s) in \c fresh_ctx hold the most recent
00194 version of the buffer
00195 */
00196       auto fresh_context = *(fresh_ctx.begin());
00197       auto fresh_q = fresh_context.get_boost_queue();
00198       fresh_q.enqueue_read_buffer(buffer_cache[fresh_context], 0, size, data);
00199       fresh_ctx.insert(host_context);
00200     }
00201   }
```

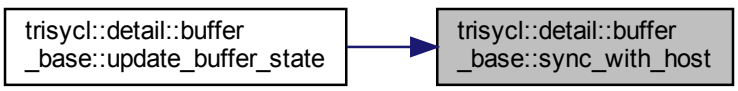
References buffer_cache, fresh_ctx, trisycl::detail::host_context::get_boost_queue(), and is_data_up_to_date().

Referenced by update_buffer_state().

Here is the call graph for this function:



Here is the caller graph for this function:



9.6.2.8.2.10 update_buffer_state()

```
void trisycl::detail::buffer_base::update_buffer_state (
    const trisycl::context & target_ctx,
    access::mode mode,
    std::size_t size,
    void * data )  [inline]
```

When a transfer is requested this function is called, it will update the state of the buffer according to the context in which the accessor is created and the access mode.

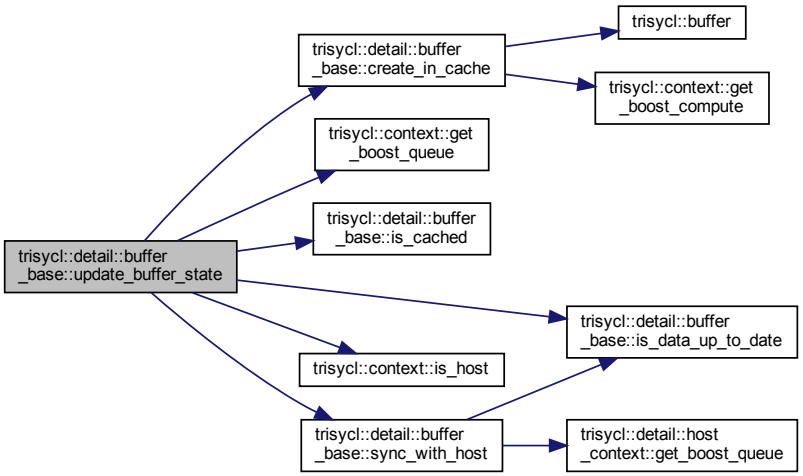
Definition at line 208 of file buffer_base.hpp.

```
00209                                                                    {
00210     /* The \c cl_buffer we put in the cache might get accessed again in the
00211 future, this means that we have to always to create it in read/write
00212 mode to be able to write to it if it is accessed through a
00213 write accessor in the future
00214 */
00215     auto constexpr flag = CL_MEM_READ_WRITE;
00216
00217     /* The buffer is accessed in read mode, we want to transfer the data only if
00218 necessary.  We start a transfer if the data on the target context is not
00219 up to date and then update the fresh context set.
00220 */
00221     if (mode == access::mode::read) {
00222
00223       if (is_data_up_to_date(target_ctx))
00224         // If read mode and the data is up-to-date there is nothing to do
00225         return;
00226
00227       // The data is not up-to-date, we need a transfer
00228       // We also want to be sure that the host holds the most recent data
00229       sync_with_host(size, data);
00230
00231       if (!target_ctx.is_host()) {
00232         // If the target context is a device context
00233         if (!is_cached(target_ctx)) {
00234           /* If not cached, we create the buffer and copy the data
00235 at the same time
00236 */
00237           create_in_cache(target_ctx, size,
00238                           (flag | CL_MEM_COPY_HOST_PTR), data);
00239           fresh_ctx.insert(target_ctx);
00240           return;
00241         }
00242
00243         /* Else we transfer the data to the existing buffer associated
00244 with the target context buffer
00245 */
00246         auto q = target_ctx.get_boost_queue();
00247         q.enqueue_write_buffer(buffer_cache[target_ctx], 0, size, data);
00248         fresh_ctx.insert(target_ctx);
00249       }
00250       return;
00251     }
00252
00253     /* The buffer might be written to, this means that we have to consider
00254 every version of the data obsolete except in the target context
00255
00256 We go through the same process as in read mode but in addition
00257 we empty the fresh context set and just add the target context
00258
00259 If the data is up to date on the target we just have to update
00260 the context set and nothing else
00261 */
00262     if (!is_data_up_to_date(target_ctx)) {
00263
00264       if (   mode == access::mode::read_write
00265           || mode == access::mode::write
00266           || mode == access::mode::atomic) {
00267         // If the data is not up-to-date in the target context
00268         // We want to host to be up-to-date
00269         sync_with_host(size, data);
00270
00271         if (!target_ctx.is_host()) {
00272           // If the target context is a device context
00273           if (!is_cached(target_ctx)) {
00274             create_in_cache(target_ctx, size,
00275                             (flag | CL_MEM_COPY_HOST_PTR), data);
00276           }
00277           else {
00278             // We update the buffer associated with the target context
00279             auto q = target_ctx.get_boost_queue();
00280             q.enqueue_write_buffer(buffer_cache[target_ctx], 0, size, data);
00281           }
```

```
00282         }
00283       }
00284
00285       /* When in discard mode we don't need to transfer any data, we just create
00286 the \c cl_buffer if it doesn't exist in the cache
00287 */
00288       if (   mode == access::mode::discard_write
00289           || mode == access::mode::discard_read_write) {
00290         /* We only need to create the buffer if it doesn't exist
00291 but without copying any data because of the discard mode
00292 */
00293         if (!target_ctx.is_host() && !is_cached(target_ctx)) {
00294           // If the context doesn't exist we create it.
00295           /* We don't want to transfer any data so we don't
00296 add \c CL_MEM_COPY_HOST_PTR
00297 */
00298           create_in_cache(target_ctx, size, flag, 0);
00299         }
00300       }
00301     }
00302     /* Here we are sure that we are in some kind of write mode,
00303 we indicate that all contexts except the target context
00304 are not up-to-date anymore
00305 */
00306     fresh_ctx.clear();
00307     fresh_ctx.insert(target_ctx);
00308   }
```

References trisycl::access::atomic, buffer_cache, create_in_cache(), trisycl::access::discard_read_write, trisycl::access::discard_write, fresh_ctx, trisycl::context::get_boost_queue(), is_cached(), is_data_up_to_date(), trisycl::context::is_host(), trisycl::access::read, trisycl::access::read_write, sync_with_host(), and trisycl::access::write.

Here is the call graph for this function:



9.6.2.8.2.11 use() `void trisycl::detail::buffer_base::use ( ) [inline]`

Mark this buffer in use by a task.

Definition at line 110 of file buffer_base.hpp.

```
00110             {
00111     // Increment the use count
00112     ++number_of_users;
00113   }
```

References number_of_users.

9.6.2.8.2.12 wait() `void trisycl::detail::buffer_base::wait ( ) [inline]`

Wait for this buffer to be ready, which is no longer in use.

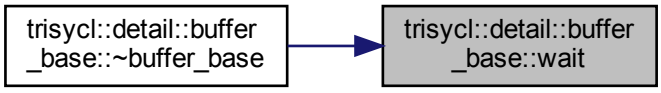
Definition at line 100 of file buffer_base.hpp.

```
00100             {
00101     std::unique_lock<std::mutex> ul { ready_mutex };
00102     ready.wait(ul, [&] {
00103         // When there is no producer for this buffer, we are ready to use it
00104         return number_of_users == 0;
00105       });
00106   }
```

References number_of_users, ready, and ready_mutex.

Referenced by ~buffer_base().

Here is the caller graph for this function:



9.6.2.8.3 Member Data Documentation

9.6.2.8.3.1 buffer_cache `std::unordered_map<trisycl::context, boost::compute::buffer> trisycl↩ ::detail::buffer_base::buffer_cache`

Buffer-side cache that keeps the `boost::compute::buffer` (and the underlying `cl_buffer`) so that if the buffer already exists inside the same context it is not recreated.

Definition at line 79 of file buffer_base.hpp.

Referenced by create_in_cache(), get_cl_buffer(), is_cached(), sync_with_host(), and update_buffer_state().

9.6.2.8.3.2 fresh_ctx `std::unordered_set<trisycl::context> trisycl::detail::buffer_base::fresh↩ _ctx`

To track contexts in which the data is up-to-date.

Definition at line 72 of file buffer_base.hpp.

Referenced by is_data_up_to_date(), sync_with_host(), and update_buffer_state().

9.6.2.8.3.3 latest_producer `std::weak_ptr<detail::task> trisycl::detail::buffer_base::latest_producer`

Track the latest task to produce this buffer.

Definition at line 55 of file buffer_base.hpp.

Referenced by get_latest_producer(), and set_latest_producer().

9.6.2.8.3.4 latest_producer_mutex `std::mutex trisycl::detail::buffer_base::latest_producer_mutex`

To protect the access to latest_producer.

Definition at line 57 of file buffer_base.hpp.

Referenced by get_latest_producer(), and set_latest_producer().

9.6.2.8.3.5 notify_buffer_destructor `boost::optional<std::promise<void> > trisycl::detail::buffer_base::notify_buffer_destructor`

If the SYCL user buffer destructor is blocking, use this to block until this buffer implementation is destroyed.

Use a void promise since there is no value to send, only waiting

Definition at line 69 of file buffer_base.hpp.

Referenced by ~buffer_base().

9.6.2.8.3.6 number_of_users `std::atomic<size_t> trisycl::detail::buffer_base::number_of_users`

Definition at line 52 of file buffer_base.hpp.

Referenced by release(), use(), and wait().

9.6.2.8.3.7 ready `std::condition_variable trisycl::detail::buffer_base::ready`

To signal when this buffer ready.

Definition at line 60 of file buffer_base.hpp.

Referenced by release(), and wait().

9.6.2.8.3.8 ready_mutex `std::mutex trisycl::detail::buffer_base::ready_mutex`

To protect the access to the condition variable.

Definition at line 62 of file buffer_base.hpp.

Referenced by release(), and wait().

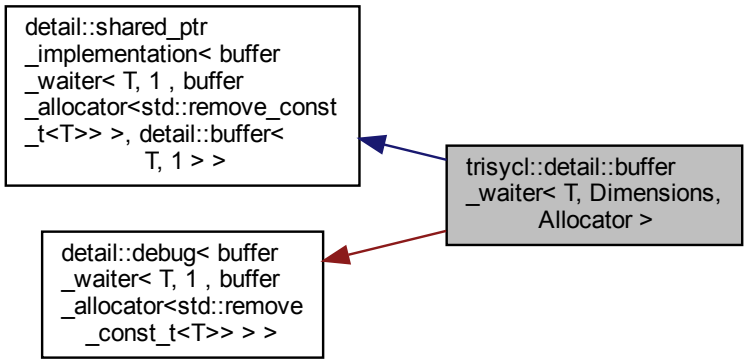
9.6.2.9 class trisycl::detail::buffer_waiter

template<typename T, int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
class trisycl::detail::buffer_waiter< T, Dimensions, Allocator >

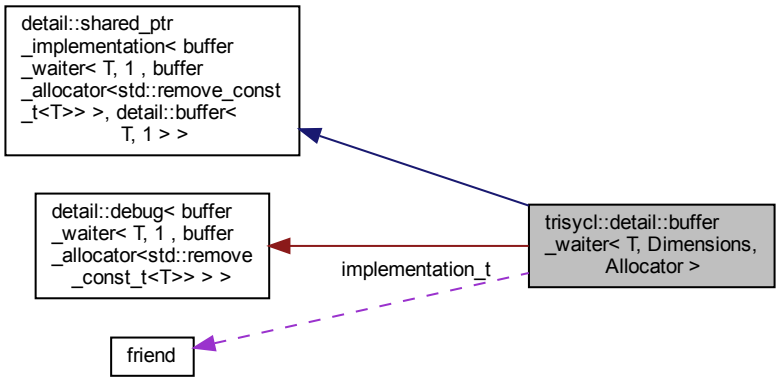
A helper class to wait for the final buffer destruction if the conditions for blocking are met.

Definition at line 31 of file buffer_waiter.hpp.

Inheritance diagram for trisycl::detail::buffer_waiter< T, Dimensions, Allocator >:



Collaboration diagram for trisycl::detail::buffer_waiter< T, Dimensions, Allocator >:



Public Member Functions

- buffer_waiter (detail::buffer< T, Dimensions > *b)

  Create a new buffer_waiter on top of a detail::buffer.
- ~buffer_waiter ()

  The buffer_waiter destructor waits for any data to be written back to the host, if any.

Private Types

- using implementation_t = typename buffer_waiter::shared_ptr_implementation

Private Attributes

- friend implementation_t

9.6.2.9.1 Member Typedef Documentation

9.6.2.9.1.1 implementation_t `template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>`

`using trisycl::detail::buffer_waiter< T, Dimensions, Allocator >::implementation_t = typename buffer_waiter::shared_ptr_implementation [private]`

Definition at line 39 of file buffer_waiter.hpp.

9.6.2.9.2 Constructor & Destructor Documentation

9.6.2.9.2.1 buffer_waiter() `template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>`

`trisycl::detail::buffer_waiter< T, Dimensions, Allocator >::buffer_waiter ( detail::buffer< T, Dimensions > * b ) [inline]`

Create a new buffer_waiter on top of a detail::buffer.

Definition at line 50 of file buffer_waiter.hpp.

```
00050 : implementation_t { b } {}
```

9.6.2.9.2.2 ~buffer_waiter() `template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>`

`trisycl::detail::buffer_waiter< T, Dimensions, Allocator >::~buffer_waiter ( ) [inline]`

The buffer_waiter destructor waits for any data to be written back to the host, if any.

Definition at line 56 of file buffer_waiter.hpp.

```
00056                   {
00057     /* Get a future from the implementation if we have to wait for its
00058 destruction */
00059     auto f = implementation->get_destructor_future();
00060     if (f) {
00061       /* No longer carry for the implementation buffer which is free to
00062 live its life up to its destruction */
00063       implementation.reset();
00064       TRISYCL_DUMP_T("~buffer_waiter() is waiting");
00065       // Then wait for its end in some other thread
00066       f->wait();
00067       TRISYCL_DUMP_T("~buffer_waiter() is done");
00068     }
00069   }
```

References TRISYCL_DUMP_T.

9.6.2.9.3 Member Data Documentation

9.6.2.9.3.1 implementation_t `template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>`

`friend trisycl::detail::buffer_waiter< T, Dimensions, Allocator >::implementation_t [private]`

Definition at line 42 of file buffer_waiter.hpp.

9.6.2.10 struct trisycl::image

template<int Dimensions>
struct trisycl::image< Dimensions >

Todo implement image

Definition at line 22 of file image.hpp.

9.6.2.11 class trisycl::pipe

template<typename Name, typename T, std::size_t MinCapacity = 0, typename SizeT = std::size_t>
class trisycl::pipe< Name, T, MinCapacity, SizeT >

SYCL pipe object.

The member functions can throw for now to help debugging or implementation on CPU.

This is mainly a CPU implementation of the Intel extension described in

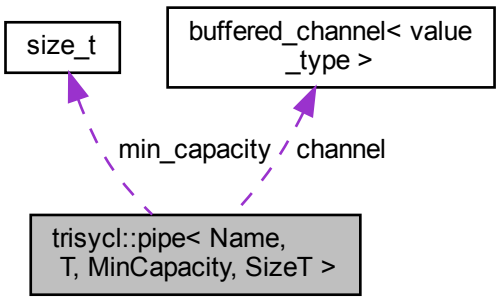
- https://github.com/intel/llvm/pull/292
- https://github.com/intel/llvm/pull/635
- https://github.com/intel/llvm/issues/832

Todo Bike-shed the naming & API according to current C++ proposals or libraries about static sized FIFO

This could be implemented on top of https://xilinx.github.io/XRT/master/html/streaming_data_support.html and/or expose some features from XRT

Definition at line 47 of file pipe.hpp.

Collaboration diagram for trisycl::pipe< Name, T, MinCapacity, SizeT >:



Public Types

- using value_type = T
- using size_type = SizeT

Static Public Member Functions

- static value_type read ()

  Blocking read from the pipe.
- static void write (const value_type &value)

  Blocking write to the pipe.
- static value_type read (bool &success_code)

  Non-blocking read from the pipe.
- static void write (const value_type &data, bool &success_code)

  Non-blocking write to the pipe.

Static Public Attributes

- static constexpr SizeT min_capacity = bit_ceil_at_least_2(MinCapacity)

  The minimum capacity in value_type of this pipe.

Static Private Member Functions

- static constexpr std::size_t bit_ceil_at_least_2 (std::size_t value)

  Finds the smallest integral power of two not less than the given value but at least 2 to fit the requirement from boost::fibers::buffered_channel.

Static Private Attributes

- static boost::fibers::buffered_channel< value_type > channel { min_capacity }

  A FIFO used to implement the pipe.

9.6.2.11.1 Member Typedef Documentation

9.6.2.11.1.1 size_type

```
template<typename Name , typename T , std::size_t MinCapacity = 0, typename
SizeT = std::size_t>
using trisycl::pipe< Name, T, MinCapacity, SizeT >::size_type = SizeT
```

Definition at line 73 of file pipe.hpp.

9.6.2.11.1.2 value_type

```
template<typename Name , typename T , std::size_t MinCapacity = 0,
typename SizeT = std::size_t>
using trisycl::pipe< Name, T, MinCapacity, SizeT >::value_type = T
```

Definition at line 65 of file pipe.hpp.

9.6.2.11.2 Member Function Documentation

9.6.2.11.2.1 bit_ceil_at_least_2() `template<typename Name , typename T , std::size_t MinCapacity = 0, typename SizeT = std::size_t>`

```
static constexpr std::size_t trisycl::pipe< Name, T, MinCapacity, SizeT >::bit_ceil_at_least_2 (
              std::size_t value )  [inline], [static], [constexpr], [private]
```

Finds the smallest integral power of two not less than the given value but at least 2 to fit the requirement from boost::fibers::buffered_channel.

Todo At some point use C++20 std::bit_ceil instead of this DIY function

Definition at line 55 of file pipe.hpp.

```
00055                                                                    {
00056     std::size_t result = 2;
00057     while (value > result)
00058       result *= 2;
00059     return result;
00060   }
```

9.6.2.11.2.2 read() [1/2] `template<typename Name , typename T , std::size_t MinCapacity = 0, typename SizeT = std::size_t>`

```
static value_type trisycl::pipe< Name, T, MinCapacity, SizeT >::read ( )  [inline], [static]
```

Blocking read from the pipe.

Returns

the value

Definition at line 87 of file pipe.hpp.

```
00087                           {
00088     return channel.value_pop();
00089   }
```

References trisycl::pipe< Name, T, MinCapacity, SizeT >::channel.

9.6.2.11.2.3 read() [2/2] `template<typename Name , typename T , std::size_t MinCapacity = 0, typename SizeT = std::size_t>`

```
static value_type trisycl::pipe< Name, T, MinCapacity, SizeT >::read (
              bool & success_code )  [inline], [static]
```

Non-blocking read from the pipe.

Parameters

out	*success_code*	is true iff the read was successful

Returns

the value read from the pipe or the default one

Definition at line 107 of file pipe.hpp.

```
00107                                                {
00108     value_type value;
00109     success_code =
00110       channel.try_pop(value) == boost::fibers::channel_op_status::success;
00111     /* The current API might be inefficient if we copy always value back.
00112 But we should have a copy elision...  */
00113     return value;
00114   }
```

References trisycl::pipe< Name, T, MinCapacity, SizeT >::channel.

9.6.2.11.2.4 write() [1/2] template<typename Name , typename T , std::size_t MinCapacity = 0, typename SizeT = std::size_t>

```
static void trisycl::pipe< Name, T, MinCapacity, SizeT >::write (
              const value_type & data,
              bool & success_code )  [inline], [static]
```

Non-blocking write to the pipe.

Parameters

out	*success_code*	is true iff the read was successful

Returns

the value read from the pipe or the default one

Definition at line 123 of file pipe.hpp.

```
00123                                                                {
00124       success_code =
00125         channel.try_push(data) != boost::fibers::channel_op_status::full;
00126   }
```

References trisycl::pipe< Name, T, MinCapacity, SizeT >::channel.

9.6.2.11.2.5 write() [2/2] template<typename Name , typename T , std::size_t MinCapacity = 0, typename SizeT = std::size_t>

```
static void trisycl::pipe< Name, T, MinCapacity, SizeT >::write (
              const value_type & value )  [inline], [static]
```

Blocking write to the pipe.

Parameters

in	*value*	is the data to send to the pipe

Definition at line 96 of file pipe.hpp.

```
00096                                           {
00097     channel.push(value);
00098   }
```

References trisycl::pipe< Name, T, MinCapacity, SizeT >::channel.

9.6.2.11.3 Member Data Documentation

9.6.2.11.3.1 channel `template<typename Name , typename T , std::size_t MinCapacity = 0, typename SizeT = std::size_t>`
`boost::fibers::buffered_channel<value_type> trisycl::pipe< Name, T, MinCapacity, SizeT >::channel { min_capacity }  [inline], [static], [private]`

A FIFO used to implement the pipe.

Definition at line 79 of file pipe.hpp.

Referenced by trisycl::pipe< Name, T, MinCapacity, SizeT >::read(), and trisycl::pipe< Name, T, MinCapacity, SizeT >::write().

9.6.2.11.3.2 min_capacity `template<typename Name , typename T , std::size_t MinCapacity = 0, typename SizeT = std::size_t>`
`constexpr SizeT trisycl::pipe< Name, T, MinCapacity, SizeT >::min_capacity = bit_ceil_at_least_2(MinCapacity)  [static], [constexpr]`

The minimum capacity in value_type of this pipe.

The underlying implementation requires a power of 2

Definition at line 70 of file pipe.hpp.

9.6.3 Typedef Documentation

9.6.3.1 buffer_allocator

```
template<typename T >
using trisycl::buffer_allocator = typedef std::allocator<T>
```

`#include <include/triSYCL/allocator.hpp>`

The allocator objects give the programmer some control on how the memory is allocated inside SYCL.

The default buffer allocator used by the runtime, when no allocator is defined by the user.

The allocator used for the `buffer` inside SYCL

Just use the default allocator for now.

Reuse the C++ default allocator.

Definition at line 29 of file allocator.hpp.

9.6.3.2 image_allocator

```
template<typename T >
using trisycl::image_allocator = typedef std::allocator<T>
```

`#include <include/triSYCL/allocator.hpp>`

The allocator used for the `image` inside SYCL.

Just use the default allocator for now.

Definition at line 37 of file allocator.hpp.

9.6.3.3 map_allocator

```
template<typename T >
using trisycl::map_allocator = typedef std::allocator<T>
```

`#include <include/triSYCL/allocator.hpp>`

The allocator used to map the memory at the same place.

Just use the default allocator for now.

Todo : implement and clarify the specification. It looks like it is not really an allocator according the current spec

Definition at line 48 of file allocator.hpp.

9.6.4 Function Documentation

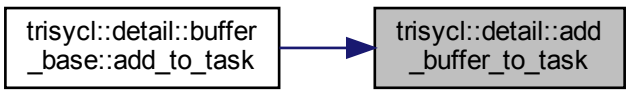
9.6.4.1 add_buffer_to_task()

```
static std::shared_ptr< detail::task > trisycl::detail::add_buffer_to_task (
            handler * command_group_handler,
            std::shared_ptr< detail::buffer_base > b,
            bool is_write_mode )  [inline], [static]
```

`#include <include/triSYCL/buffer/detail/buffer_base.hpp>`

Referenced by trisycl::detail::buffer_base::add_to_task().

Here is the caller graph for this function:



9.6.4.2 buffer() [1/2]

```
trisycl::buffer (
            auto & host_data ) -> buffer< std::ranges::range_value_t< decltype(host_data)>,
1 >
```

`#include <include/triSYCL/buffer.hpp>`

A deduction guide to infer the buffer type from the read-write range.

Todo Use some standard concepts

Todo Add this to SYCL Next

9.6.4.3 buffer() [2/2]

```
template<std::forward_iterator FIB, std::forward_iterator FIE>
trisycl::buffer (
            FIB begin,
            FIE end ) -> buffer< typename std::iterator_traits< FIB >::value_type, 1 >
```

`#include <include/triSYCL/buffer.hpp>`

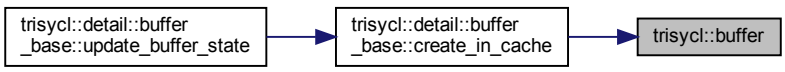
A deduction guide to infer the buffer type from the read-write forward iterators into a copy-in on construction and copy-back on destruction.

Todo Use some standard concepts

Todo Add this to SYCL Next

Referenced by trisycl::detail::buffer_base::create_in_cache().

Here is the caller graph for this function:



9.6.4.4 buffer_add_to_task()

```
template<typename BufferDetail >
static std::shared_ptr< detail::task > trisycl::detail::buffer_add_to_task (
            BufferDetail buf,
            handler * command_group_handler,
            bool is_write_mode )  [static]
```

`#include <include/triSYCL/buffer/detail/buffer.hpp>`

Proxy function to avoid some circular type recursion.

Returns

a shared_ptr<task>

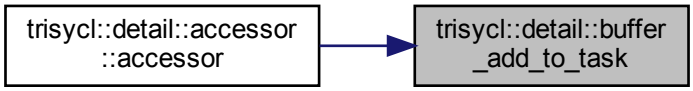
Todo To remove with some refactoring

Definition at line 352 of file buffer.hpp.

```
00353                                        {
00354   return buf->add_to_task(command_group_handler, is_write_mode);
00355 }
```

Referenced by trisycl::detail::accessor< T, Dimensions, Mode, Target >::accessor().

Here is the caller graph for this function:



9.6.4.5 waiter()

```
template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_↩
const_t<T>>>
auto trisycl::detail::waiter (
            detail::buffer< T, Dimensions > * b )  [inline]
```

`#include <include/triSYCL/buffer/detail/buffer_waiter.hpp>`

Helper function to create a new buffer_waiter.

Definition at line 77 of file buffer_waiter.hpp.

```
00077                                                {
00078   return new buffer_waiter<T, Dimensions, Allocator> { b };
00079 }
```

References b.

9.7 triSYCL extension for storage scopes

Classes

- struct trisycl::vendor::trisycl::scope::empty_device_scope

  An empty device scope to make compilation errors clearer. More...
- class trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >

  A conceptual device implementing some device-scoped storage and based on a platform with platform-scoped storage. More...
- class trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >

  A conceptual device implementing some device-scoped storage and based on a platform with platform-scoped storage. More...
- struct trisycl::vendor::trisycl::scope::empty_platform_scope

  An empty platform scope to make compilation errors clearer. More...
- class trisycl::vendor::trisycl::scope::platform< PlatformStorage >

  A conceptual platform implementing some platform-scoped storage. More...
- class trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >

  A conceptual platform implementing some platform-scoped storage. More...
- struct trisycl::vendor::trisycl::scope::empty_queue_scope

  An empty queue scope to make compilation errors clearer. More...
- class trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >

  A conceptual queue implementing some queue-scoped storage constructed from a device with some device-scope storage. More...
- class trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >

  A conceptual queue implementing some queue-scoped storage constructed from a device with some device-scope storage. More...

9.7.1 Detailed Description

9.7.2 Class Documentation

9.7.2.1 struct trisycl::vendor::trisycl::scope::empty_device_scope

An empty device scope to make compilation errors clearer.

Definition at line 38 of file device.hpp.

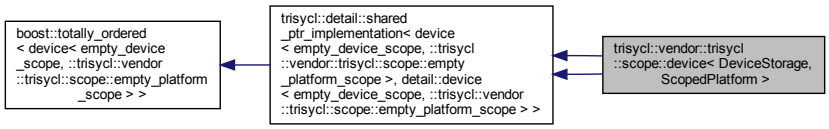
9.7.2.2 class trisycl::vendor::trisycl::scope::device

template<typename DeviceStorage = empty_device_scope, typename ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
class trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >

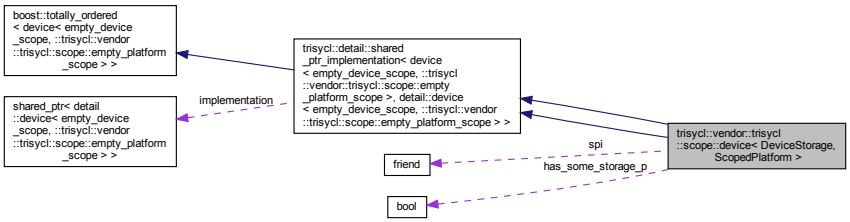
A conceptual device implementing some device-scoped storage and based on a platform with platform-scoped storage.

Definition at line 49 of file device.hpp.

Inheritance diagram for trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >:



Collaboration diagram for trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >:



Public Types

- using storage_type = DeviceStorage

  The type of the scoped storage in the device.
- using device_scope_type = storage_type

  The device scope storage type.
- using platform_type = ScopedPlatform

  The platform type targeted by this queue.
- using platform_scope_type = detail::storage_type_trait_t< platform_type >

  The platform scope storage type.
- using storage_type = DeviceStorage

  The type of the scoped storage in the device.
- using device_scope_type = storage_type

  The device scope storage type.
- using platform_type = ScopedPlatform

  The platform type targeted by this queue.
- using platform_scope_type = detail::storage_type_trait_t< platform_type >

  The platform scope storage type.

Public Member Functions

- device (const ::trisycl::device &d, const ScopedPlatform &p={})

  Construct the device with some device-scoped storage on top of a SYCL device.
- device ()

  By default use the default (host) device with no platform storage.
- device (const device &)=default

  It is still copyable.

- device (device &&)=default

  It is still movable.
- auto & get_storage () const

  Access to the device-scoped storage.
- auto & get_platform () const

  Access to the underlying scoped platform.
- operator::trisycl::device & () const

  Add a conversion to `trisycl::device&` so it can be used as a normal device.
- device (const ::trisycl::device &d, const ScopedPlatform &p={})

  Construct the device with some device-scoped storage on top of a SYCL device.
- device ()

  By default use the default (host) device with no platform storage.
- device (const device &)=default

  It is still copyable.
- device (device &&)=default

  It is still movable.
- auto & get_storage () const

  Access to the device-scoped storage.
- auto & get_platform () const

  Access to the underlying scoped platform.
- operator::trisycl::device & () const

  Add a conversion to `trisycl::device&` so it can be used as a normal device.

Public Attributes

- std::shared_ptr< Implementation > implementation

  The implementation forward everything to this... implementation.
- std::shared_ptr< Implementation > implementation

  The implementation forward everything to this... implementation.

Static Public Attributes

- static constexpr bool has_some_storage_p

  True if there is some storage in the device.

Private Types

- using spi = ::trisycl::detail::shared_ptr_implementation< device< DeviceStorage, ScopedPlatform >, detail::device< DeviceStorage, ScopedPlatform > >
- using spi = ::trisycl::detail::shared_ptr_implementation< device< DeviceStorage, ScopedPlatform >, detail::device< DeviceStorage, ScopedPlatform > >

Private Attributes

- friend spi

9.7.2.2.1 Member Typedef Documentation

9.7.2.2.1.1 device_scope_type [1/2]

```
template<typename DeviceStorage = empty_device_scope, typename
ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
using trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::device_scope↩
_type = storage_type
```

The device scope storage type.

Definition at line 77 of file device.hpp.

9.7.2.2.1.2 device_scope_type [2/2]

```
template<typename DeviceStorage = empty_device_scope, typename
ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
using trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::device_scope↩
_type = storage_type
```

The device scope storage type.

Definition at line 77 of file device.hpp.

9.7.2.2.1.3 platform_scope_type [1/2]

```
template<typename DeviceStorage = empty_device_scope,
typename ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
using trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::platform_↩
scope_type = detail::storage_type_trait_t<platform_type>
```

The platform scope storage type.

Definition at line 83 of file device.hpp.

9.7.2.2.1.4 platform_scope_type [2/2]

```
template<typename DeviceStorage = empty_device_scope,
typename ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
using trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::platform_↩
scope_type = detail::storage_type_trait_t<platform_type>
```

The platform scope storage type.

Definition at line 83 of file device.hpp.

9.7.2.2.1.5 platform_type [1/2]

```
template<typename DeviceStorage = empty_device_scope, typename
ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
using trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::platform_type
= ScopedPlatform
```

The platform type targeted by this queue.

Definition at line 80 of file device.hpp.

9.7.2.2.1.6 platform_type [2/2]

```
template<typename DeviceStorage = empty_device_scope, typename
ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
using trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::platform_type
= ScopedPlatform
```

The platform type targeted by this queue.

Definition at line 80 of file device.hpp.

9.7.2.2.1.7 spi [1/2]

```
template<typename DeviceStorage = empty_device_scope, typename Scoped↩
Platform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
using trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::spi = ::trisycl::detail::shar
Storage, ScopedPlatform>, detail::device <DeviceStorage, ScopedPlatform> >  [private]
```

Definition at line 58 of file device.hpp.

9.7.2.2.1.8 spi [2/2]

```
template<typename DeviceStorage = empty_device_scope, typename Scoped↩
Platform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
using trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::spi = ::trisycl::detail::shar
Storage, ScopedPlatform>, detail::device <DeviceStorage, ScopedPlatform> >  [private]
```

Definition at line 58 of file device.hpp.

9.7.2.2.1.9 storage_type [1/2]

```
template<typename DeviceStorage = empty_device_scope, typename
ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
using trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::storage_type =
DeviceStorage
```

The type of the scoped storage in the device.

Definition at line 74 of file device.hpp.

9.7.2.2.1.10 storage_type [2/2]

```
template<typename DeviceStorage = empty_device_scope, typename
ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
using trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::storage_type =
DeviceStorage
```

The type of the scoped storage in the device.

Definition at line 74 of file device.hpp.

9.7.2.2.2 Constructor & Destructor Documentation

9.7.2.2.2.1 device() [1/8]

```
template<typename DeviceStorage = empty_device_scope, typename Scoped↩
Platform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::device (
            const ::trisycl::device< DeviceStorage, ScopedPlatform > & d,
            const ScopedPlatform & p = {} )  [inline]
```

Construct the device with some device-scoped storage on top of a SYCL device.

Parameters

in	*d*	is the real device to use

Definition at line 95 of file device.hpp.

```
00096                                       {}) :
00097     spi {
00098       new detail::device<DeviceStorage, ScopedPlatform> { d, p }
00099     } {}
```

9.7.2.2.2 device() [2/8]

```
template<typename DeviceStorage = empty_device_scope, typename Scoped↩
Platform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::device ( )  [inline]
```

By default use the default (host) device with no platform storage.

Definition at line 103 of file device.hpp.

```
00103 :  device(::trisycl::device {}) {}
```

9.7.2.2.3 device() [3/8]

```
template<typename DeviceStorage = empty_device_scope, typename Scoped↩
Platform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::device (
            const device< DeviceStorage, ScopedPlatform > &  )  [default]
```

It is still copyable.

9.7.2.2.4 device() [4/8]

```
template<typename DeviceStorage = empty_device_scope, typename Scoped↩
Platform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::device (
            device< DeviceStorage, ScopedPlatform > &&  )  [default]
```

It is still movable.

9.7.2.2.5 device() [5/8]

```
template<typename DeviceStorage = empty_device_scope, typename Scoped↩
Platform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::device (
            const ::trisycl::device< DeviceStorage, ScopedPlatform > & d,
            const ScopedPlatform & p = {} )  [inline]
```

Construct the device with some device-scoped storage on top of a SYCL device.

Parameters

in	*d*	is the real device to use

Definition at line 95 of file device.hpp.

```
00096                                     {}) :
00097     spi {
00098       new detail::device<DeviceStorage, ScopedPlatform> { d, p }
00099     } {}
```

9.7.2.2.6 device() [6/8]

```
template<typename DeviceStorage = empty_device_scope, typename Scoped↩
Platform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::device ( )  [inline]
```

By default use the default (host) device with no platform storage.

Definition at line 103 of file device.hpp.

```
00103 :  device(::trisycl::device {}) {}
```

9.7.2.2.7 device() [7/8]

```
template<typename DeviceStorage = empty_device_scope, typename Scoped↩
Platform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::device (
            const device< DeviceStorage, ScopedPlatform > &  )  [default]
```

It is still copyable.

9.7.2.2.8 device() [8/8]

```
template<typename DeviceStorage = empty_device_scope, typename Scoped↩
Platform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::device (
            device< DeviceStorage, ScopedPlatform > &&  )  [default]
```

It is still movable.

9.7.2.2.3 Member Function Documentation

9.7.2.2.3.1 get_platform() [1/2]

```
template<typename DeviceStorage = empty_device_scope, typename
ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
auto & trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::get_platform
( ) const  [inline]
```

Access to the underlying scoped platform.

Definition at line 121 of file device.hpp.

```
00121                                 {
00122     return implementation->get_platform();
00123   }
```

References trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::implementation.

9.7.2.2.3.2 get_platform() [2/2]

```
template<typename DeviceStorage = empty_device_scope, typename
ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
auto & trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::get_platform
( ) const  [inline]
```

Access to the underlying scoped platform.

Definition at line 121 of file device.hpp.

```
00121                                {
00122     return implementation->get_platform();
00123   }
```

References trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::implementation.

9.7.2.2.3.3 get_storage() [1/2]

```
template<typename DeviceStorage = empty_device_scope, typename
ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
auto & trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::get_storage (
) const  [inline]
```

Access to the device-scoped storage.

Definition at line 115 of file device.hpp.

```
00115                               {
00116     return implementation->get_storage();
00117   }
```

References trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::implementation.

9.7.2.2.3.4 get_storage() [2/2]

```
template<typename DeviceStorage = empty_device_scope, typename
ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
auto & trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::get_storage (
) const  [inline]
```

Access to the device-scoped storage.

Definition at line 115 of file device.hpp.

```
00115                               {
00116     return implementation->get_storage();
00117   }
```

References trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::implementation.

9.7.2.2.3.5 operator::trisycl::device &() [1/2]

```
template<typename DeviceStorage = empty_device_↩
scope, typename ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::operator::trisycl↩
::device & ( ) const  [inline]
```

Add a conversion to `trisycl::device&` so it can be used as a normal device.

Definition at line 128 of file device.hpp.

```
00128                                    {
00129       return implementation->get_underlying_device();
00130     }
```

References trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::implementation.

9.7.2.2.3.6 operator::trisycl::device &() [2/2]

```
template<typename DeviceStorage = empty_device_scope, typename ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::operator::trisycl::device & ( ) const  [inline]
```

Add a conversion to `trisycl::device&` so it can be used as a normal device.

Definition at line 128 of file device.hpp.

```
00128                                    {
00129       return implementation->get_underlying_device();
00130     }
```

References trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::implementation.

9.7.2.2.4 Member Data Documentation

9.7.2.2.4.1 has_some_storage_p

```
template<typename DeviceStorage = empty_device_scope, typename ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
constexpr bool trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::has_some_storage_p  [static], [constexpr]
```

Initial value:

```
=
    not std::is_same_v<DeviceStorage, empty_device_scope>
```

True if there is some storage in the device.

Definition at line 86 of file device.hpp.

9.7.2.2.4.2 implementation [1/2]

```
template<typename DeviceStorage = empty_device_scope, typename ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
std::shared_ptr<Implementation> trisycl::detail::shared_ptr_implementation< Parent, Implementation >::implementation
```

The implementation forward everything to this... implementation.

Definition at line 41 of file shared_ptr_implementation.hpp.

Referenced by trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::get_platform(), trisycl::vendor::trisycl::scope and trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::operator::trisycl::device &().

9.7.2.2.4.3 implementation [2/2]

```
template<typename DeviceStorage = empty_device_scope, typename ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>
std::shared_ptr<Implementation> trisycl::detail::shared_ptr_implementation< Parent, Implementation >::implementation
```

The implementation forward everything to this... implementation.

Definition at line 41 of file shared_ptr_implementation.hpp.

9.7.2.2.4.4 spi `template<typename DeviceStorage = empty_device_scope, typename ScopedPlatform = ::trisycl::vendor::trisycl::scope::empty_platform_scope>`

`friend trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >::spi [private]`

Definition at line 66 of file device.hpp.

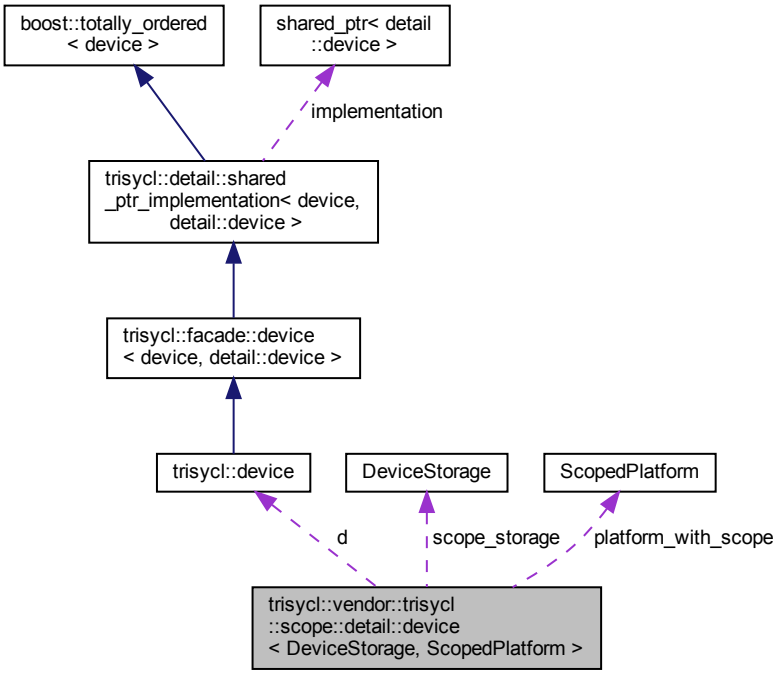
9.7.2.3 class trisycl::vendor::trisycl::scope::detail::device

template<typename DeviceStorage, typename ScopedPlatform>
class trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >

A conceptual device implementing some device-scoped storage and based on a platform with platform-scoped storage.

Definition at line 29 of file device.hpp.

Collaboration diagram for trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >:



Public Member Functions

- device (const ::trisycl::device &d, const ScopedPlatform &p)

  Construct the device with some device-scoped storage on top of a SYCL device.
- device (const ::trisycl::device &d)

  Construct the device with some device-scoped storage on top of a SYCL device.
- auto & get_underlying_device ()

  Get the device behind the curtain.

- auto & get_storage ()

  Access to the device-scoped storage.
- auto & get_platform ()

  Access to the underlying scoped platform.
- device (const ::trisycl::device &d, const ScopedPlatform &p)

  Construct the device with some device-scoped storage on top of a SYCL device.
- device (const ::trisycl::device &d)

  Construct the device with some device-scoped storage on top of a SYCL device.
- auto & get_underlying_device ()

  Get the device behind the curtain.
- auto & get_storage ()

  Access to the device-scoped storage.
- auto & get_platform ()

  Access to the underlying scoped platform.

Private Attributes

- ::trisycl::device d

  The storage-less device behind the scene.
- DeviceStorage scope_storage

  The device-scoped storage default-initialized.
- ScopedPlatform platform_with_scope

  The scoped platform the scoped device is built into.

9.7.2.3.1 Constructor & Destructor Documentation

9.7.2.3.1.1 device() [1/4]

```
template<typename DeviceStorage , typename ScopedPlatform >
trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::device (
            const ::trisycl::device< DeviceStorage, ScopedPlatform > & d,
            const ScopedPlatform & p )  [inline]
```

Construct the device with some device-scoped storage on top of a SYCL device.

Parameters

in	*d*	is the real device to use

Definition at line 52 of file device.hpp.

```
00054     : d { d }, platform_with_scope { p } {}
```

9.7.2.3.1.2 device() [2/4]

```
template<typename DeviceStorage , typename ScopedPlatform >
trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::device (
            const ::trisycl::device< DeviceStorage, ScopedPlatform > & d )  [inline]
```

Construct the device with some device-scoped storage on top of a SYCL device.

Parameters

in	*d*	is the real device to use

Definition at line 62 of file device.hpp.

```
00062 :  d { d } {}
```

9.7.2.3.1.3 device() [3/4]

```
template<typename DeviceStorage , typename ScopedPlatform >
trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::device (
            const ::trisycl::device< DeviceStorage, ScopedPlatform > & d,
            const ScopedPlatform & p )  [inline]
```

Construct the device with some device-scoped storage on top of a SYCL device.

Parameters

in	*d*	is the real device to use

Definition at line 52 of file device.hpp.

```
00054     :  d { d }, platform_with_scope { p } {}
```

9.7.2.3.1.4 device() [4/4]

```
template<typename DeviceStorage , typename ScopedPlatform >
trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::device (
            const ::trisycl::device< DeviceStorage, ScopedPlatform > & d )  [inline]
```

Construct the device with some device-scoped storage on top of a SYCL device.

Parameters

in	*d*	is the real device to use

Definition at line 62 of file device.hpp.

```
00062 :  d { d } {}
```

9.7.2.3.2 Member Function Documentation

9.7.2.3.2.1 get_platform() [1/2]

```
template<typename DeviceStorage , typename ScopedPlatform >
auto & trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::get↩
_platform ( )  [inline]
```

Access to the underlying scoped platform.

Definition at line 77 of file device.hpp.

```
00077                          {
00078     return platform_with_scope;
00079   }
```

References trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::platform_with_scope.

9.7.2.3.2.2 get_platform() [2/2]

```
template<typename DeviceStorage , typename ScopedPlatform >
auto & trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::get_platform ( ) [inline]
```

Access to the underlying scoped platform.

Definition at line 77 of file device.hpp.

```
00077                           {
00078     return platform_with_scope;
00079   }
```

References trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::platform_with_scope.

9.7.2.3.2.3 get_storage() [1/2]

```
template<typename DeviceStorage , typename ScopedPlatform >
auto & trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::get_storage ( ) [inline]
```

Access to the device-scoped storage.

Definition at line 71 of file device.hpp.

```
00071                           {
00072     return scope_storage;
00073   }
```

References trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::scope_storage.

9.7.2.3.2.4 get_storage() [2/2]

```
template<typename DeviceStorage , typename ScopedPlatform >
auto & trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::get_storage ( ) [inline]
```

Access to the device-scoped storage.

Definition at line 71 of file device.hpp.

```
00071                           {
00072     return scope_storage;
00073   }
```

References trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::scope_storage.

9.7.2.3.2.5 get_underlying_device() [1/2]

```
template<typename DeviceStorage , typename ScopedPlatform >
auto & trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::get_underlying_device ( ) [inline]
```

Get the device behind the curtain.

Definition at line 65 of file device.hpp.

```
00065                                 {
00066     return d;
00067   }
```

References trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::d.

9.7.2.3.2.6 get_underlying_device() [2/2]

```
template<typename DeviceStorage , typename ScopedPlatform >
auto & trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::get_underlying_device ( )  [inline]
```

Get the device behind the curtain.

Definition at line 65 of file device.hpp.

```
00065                                {
00066     return d;
00067   }
```

References trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::d.

9.7.2.3.3 Member Data Documentation

9.7.2.3.3.1 d

```
template<typename DeviceStorage , typename ScopedPlatform >
::trisycl::device trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::d  [private]
```

The storage-less device behind the scene.

Definition at line 31 of file device.hpp.

Referenced by trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::get_underlying_device().

9.7.2.3.3.2 platform_with_scope

```
template<typename DeviceStorage , typename ScopedPlatform >
ScopedPlatform trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::platform_with_scope  [private]
```

The scoped platform the scoped device is built into.

Definition at line 42 of file device.hpp.

Referenced by trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::get_platform().

9.7.2.3.3.3 scope_storage

```
template<typename DeviceStorage , typename ScopedPlatform >
DeviceStorage trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::scope_storage  [private]
```

The device-scoped storage default-initialized.

Todo For now it is allocated on the host in this CPU emulation but a device compiler and runtime can create this on a real device

Todo For now it is allocated on the host in this CPU emulation but a device compiler and runtime can create this on a real device

Definition at line 39 of file device.hpp.

Referenced by trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >::get_storage().

9.7.2.4 struct trisycl::vendor::trisycl::scope::empty_platform_scope

An empty platform scope to make compilation errors clearer.

Definition at line 36 of file platform.hpp.

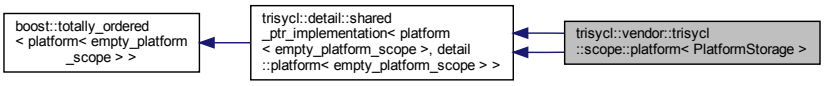
9.7.2.5 class trisycl::vendor::trisycl::scope::platform

template<typename PlatformStorage = empty_platform_scope>
class trisycl::vendor::trisycl::scope::platform< PlatformStorage >

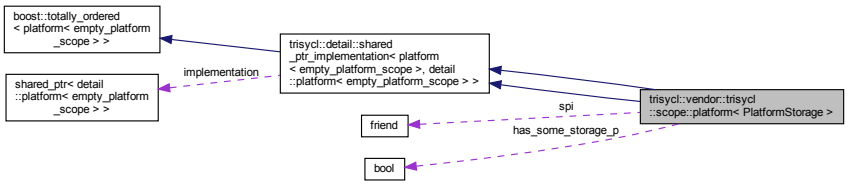
A conceptual platform implementing some platform-scoped storage.

Definition at line 42 of file platform.hpp.

Inheritance diagram for trisycl::vendor::trisycl::scope::platform< PlatformStorage >:



Collaboration diagram for trisycl::vendor::trisycl::scope::platform< PlatformStorage >:



Public Types

- using storage_type = PlatformStorage

  The type of the scoped storage in the platform.
- using platform_scope_type = PlatformStorage

  The platform scope storage type.
- using storage_type = PlatformStorage

  The type of the scoped storage in the platform.
- using platform_scope_type = PlatformStorage

  The platform scope storage type.

Public Member Functions

- platform (const ::trisycl::platform &p)

  Construct the platform with some platform-scoped storage on top of a SYCL platform.
- platform ()

  By default use the default (host) platform.
- platform (const platform &)=default

  It is still copyable.
- platform (platform &&)=default

  It is still movable.
- auto & get_storage () const

  Access to the platform-scoped storage.
- operator::trisycl::platform & () const

  Add a conversion to `trisycl::platform&` so it can be used as a normal platform.
- platform (const ::trisycl::platform &p)

  Construct the platform with some platform-scoped storage on top of a SYCL platform.
- platform ()

  By default use the default (host) platform.
- platform (const platform &)=default

  It is still copyable.
- platform (platform &&)=default

  It is still movable.
- auto & get_storage () const

  Access to the platform-scoped storage.
- operator::trisycl::platform & () const

  Add a conversion to `trisycl::platform&` so it can be used as a normal platform.

Public Attributes

- std::shared_ptr< Implementation > implementation

  The implementation forward everything to this... implementation.
- std::shared_ptr< Implementation > implementation

  The implementation forward everything to this... implementation.

Static Public Attributes

- static constexpr bool has_some_storage_p

  True if there is some storage in the platform.

Private Types

- using spi = ::trisycl::detail::shared_ptr_implementation< platform< PlatformStorage >, detail::platform< PlatformStorage > >
- using spi = ::trisycl::detail::shared_ptr_implementation< platform< PlatformStorage >, detail::platform< PlatformStorage > >

Private Attributes

- friend spi

9.7.2.5.1 Member Typedef Documentation

9.7.2.5.1.1 platform_scope_type [1/2]

```
template<typename PlatformStorage = empty_platform_scope>
using trisycl::vendor::trisycl::scope::platform< PlatformStorage >::platform_scope_type =
PlatformStorage
```

The platform scope storage type.

Definition at line 67 of file platform.hpp.

9.7.2.5.1.2 platform_scope_type [2/2]

```
template<typename PlatformStorage = empty_platform_scope>
using trisycl::vendor::trisycl::scope::platform< PlatformStorage >::platform_scope_type =
PlatformStorage
```

The platform scope storage type.

Definition at line 67 of file platform.hpp.

9.7.2.5.1.3 spi [1/2]

```
template<typename PlatformStorage = empty_platform_scope>
using trisycl::vendor::trisycl::scope::platform< PlatformStorage >::spi = ::trisycl::detail::shared_ptr_imple
Storage>, detail::platform <PlatformStorage> >  [private]
```

Definition at line 50 of file platform.hpp.

9.7.2.5.1.4 spi [2/2]

```
template<typename PlatformStorage = empty_platform_scope>
using trisycl::vendor::trisycl::scope::platform< PlatformStorage >::spi = ::trisycl::detail::shared_ptr_imple
Storage>, detail::platform <PlatformStorage> >  [private]
```

Definition at line 50 of file platform.hpp.

9.7.2.5.1.5 storage_type [1/2]

```
template<typename PlatformStorage = empty_platform_scope>
using trisycl::vendor::trisycl::scope::platform< PlatformStorage >::storage_type = Platform↩
Storage
```

The type of the scoped storage in the platform.

Definition at line 64 of file platform.hpp.

9.7.2.5.1.6 storage_type [2/2]

```
template<typename PlatformStorage = empty_platform_scope>
using trisycl::vendor::trisycl::scope::platform< PlatformStorage >::storage_type = Platform↩
Storage
```

The type of the scoped storage in the platform.

Definition at line 64 of file platform.hpp.

9.7.2.5.2 Constructor & Destructor Documentation

9.7.2.5.2.1 platform() [1/8]

```
template<typename PlatformStorage = empty_platform_scope>
trisycl::vendor::trisycl::scope::platform< PlatformStorage >::platform (
            const ::trisycl::platform< PlatformStorage > & p )  [inline]
```

Construct the platform with some platform-scoped storage on top of a SYCL platform.

Parameters

in	*p*	is the real platform to use

Definition at line 78 of file platform.hpp.

```
00078                                     :
00079     spi {
00080       new detail::platform<PlatformStorage> { p }
00081     } {}
```

References p.

9.7.2.5.2.2 platform() [2/8] template<typename PlatformStorage = empty_platform_scope> trisycl::vendor::trisycl::scope::platform< PlatformStorage >::platform ( ) [inline]

By default use the default (host) platform.

Definition at line 85 of file platform.hpp.

```
00085 : platform(::trisycl::platform {}) {}
```

9.7.2.5.2.3 platform() [3/8] template<typename PlatformStorage = empty_platform_scope> trisycl::vendor::trisycl::scope::platform< PlatformStorage >::platform (const platform< PlatformStorage > &) [default]

It is still copyable.

9.7.2.5.2.4 platform() [4/8] template<typename PlatformStorage = empty_platform_scope> trisycl::vendor::trisycl::scope::platform< PlatformStorage >::platform (platform< PlatformStorage > &&) [default]

It is still movable.

9.7.2.5.2.5 platform() [5/8] template<typename PlatformStorage = empty_platform_scope> trisycl::vendor::trisycl::scope::platform< PlatformStorage >::platform (const ::trisycl::platform< PlatformStorage > & *p*) [inline]

Construct the platform with some platform-scoped storage on top of a SYCL platform.

Parameters

in	*p*	is the real platform to use

Definition at line 78 of file platform.hpp.

```
00078                                     :
```

```
00079     spi {
00080       new detail::platform<PlatformStorage> { p }
00081     } {}
```

References p.

9.7.2.5.2.6 platform() [6/8] template<typename PlatformStorage = empty_platform_scope> trisycl::vendor::trisycl::scope::platform< PlatformStorage >::platform ( ) [inline]

By default use the default (host) platform.

Definition at line 85 of file platform.hpp.

```
00085 : platform(::trisycl::platform {}) {}
```

9.7.2.5.2.7 platform() [7/8] template<typename PlatformStorage = empty_platform_scope> trisycl::vendor::trisycl::scope::platform< PlatformStorage >::platform (const platform< PlatformStorage > &) [default]

It is still copyable.

9.7.2.5.2.8 platform() [8/8] template<typename PlatformStorage = empty_platform_scope> trisycl::vendor::trisycl::scope::platform< PlatformStorage >::platform (platform< PlatformStorage > &&) [default]

It is still movable.

9.7.2.5.3 Member Function Documentation

9.7.2.5.3.1 get_storage() [1/2] template<typename PlatformStorage = empty_platform_scope> auto & trisycl::vendor::trisycl::scope::platform< PlatformStorage >::get_storage ( ) const [inline]

Access to the platform-scoped storage.

Definition at line 97 of file platform.hpp.

```
00097                                {
00098     return implementation->get_storage();
00099   }
```

9.7.2.5.3.2 get_storage() [2/2] template<typename PlatformStorage = empty_platform_scope> auto & trisycl::vendor::trisycl::scope::platform< PlatformStorage >::get_storage ( ) const [inline]

Access to the platform-scoped storage.

Definition at line 97 of file platform.hpp.

```
00097                                {
00098     return implementation->get_storage();
00099   }
```

9.7.2.5.3.3 operator::trisycl::platform &() [1/2]

template<typename PlatformStorage = empty_platform_scope>
trisycl::vendor::trisycl::scope::platform< PlatformStorage >::operator::trisycl::platform & ( ) const [inline]

Add a conversion to `trisycl::platform&` so it can be used as a normal platform.

Definition at line 104 of file platform.hpp.

```
00104                                   {
00105       return implementation->get_underlying_platform();
00106     }
```

9.7.2.5.3.4 operator::trisycl::platform &() [2/2]

template<typename PlatformStorage = empty_platform_scope>
trisycl::vendor::trisycl::scope::platform< PlatformStorage >::operator::trisycl::platform & ( ) const [inline]

Add a conversion to `trisycl::platform&` so it can be used as a normal platform.

Definition at line 104 of file platform.hpp.

```
00104                                   {
00105       return implementation->get_underlying_platform();
00106     }
```

9.7.2.5.4 Member Data Documentation

9.7.2.5.4.1 has_some_storage_p

template<typename PlatformStorage = empty_platform_scope>
constexpr bool trisycl::vendor::trisycl::scope::platform< PlatformStorage >::has_some_storage_p [static], [constexpr]

Initial value:

```
=
    not std::is_same_v<PlatformStorage, empty_platform_scope>
```

True if there is some storage in the platform.

Definition at line 70 of file platform.hpp.

9.7.2.5.4.2 implementation [1/2]

template<typename PlatformStorage = empty_platform_scope>
std::shared_ptr<Implementation> trisycl::detail::shared_ptr_implementation< Parent, Implementation >::implementation

The implementation forward everything to this... implementation.

Definition at line 41 of file shared_ptr_implementation.hpp.

9.7.2.5.4.3 implementation [2/2] template<typename PlatformStorage = empty_platform_scope> std::shared_ptr<Implementation> trisycl::detail::shared_ptr_implementation< Parent, Implementation >::implementation

The implementation forward everything to this... implementation.

Definition at line 41 of file shared_ptr_implementation.hpp.

9.7.2.5.4.4 spi template<typename PlatformStorage = empty_platform_scope> friend trisycl::vendor::trisycl::scope::platform< PlatformStorage >::spi [private]

Definition at line 56 of file platform.hpp.

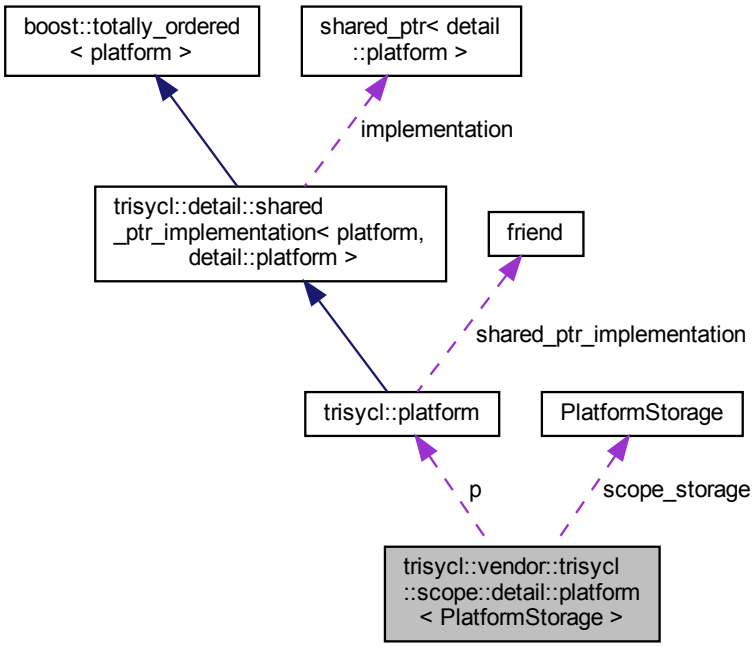
9.7.2.6 class trisycl::vendor::trisycl::scope::detail::platform

template<typename PlatformStorage>
class trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >

A conceptual platform implementing some platform-scoped storage.

Definition at line 26 of file platform.hpp.

Collaboration diagram for trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >:



Public Member Functions

- platform (const ::trisycl::platform &p)

  Construct the platform with some platform-scoped storage on top of a SYCL platform.
- auto get_underlying_platform ()

  Get the platform behind the curtain.
- auto & get_storage ()

  Access to the platform-scoped storage.
- platform (const ::trisycl::platform &p)

  Construct the platform with some platform-scoped storage on top of a SYCL platform.
- auto get_underlying_platform ()

  Get the platform behind the curtain.
- auto & get_storage ()

  Access to the platform-scoped storage.

Private Attributes

- ::trisycl::platform p

  The storage-less platform behind the scene.
- PlatformStorage scope_storage

  The platform-scoped storage default-initialized.

9.7.2.6.1 Constructor & Destructor Documentation

9.7.2.6.1.1 platform() [1/2] template<typename PlatformStorage >

```
trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::platform (
            const ::trisycl::platform< PlatformStorage > & p )  [inline]
```

Construct the platform with some platform-scoped storage on top of a SYCL platform.

Parameters

in	*p*	is the real platform to use

Definition at line 45 of file platform.hpp.

```
00045 :  p { p } {}
```

9.7.2.6.1.2 platform() [2/2] template<typename PlatformStorage >

```
trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::platform (
            const ::trisycl::platform< PlatformStorage > & p )  [inline]
```

Construct the platform with some platform-scoped storage on top of a SYCL platform.

Parameters

in	*p*	is the real platform to use

Definition at line 45 of file platform.hpp.
```
00045 :  p { p } {}
```

9.7.2.6.2 Member Function Documentation

9.7.2.6.2.1 get_storage() [1/2] template<typename PlatformStorage >

```
auto & trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::get_storage ( )
[inline]
```

Access to the platform-scoped storage.

Definition at line 54 of file platform.hpp.
```
00054                          {
00055     return scope_storage;
00056   }
```

References trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::scope_storage.

9.7.2.6.2.2 get_storage() [2/2] template<typename PlatformStorage >

```
auto & trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::get_storage ( )
[inline]
```

Access to the platform-scoped storage.

Definition at line 54 of file platform.hpp.
```
00054                          {
00055     return scope_storage;
00056   }
```

References trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::scope_storage.

9.7.2.6.2.3 get_underlying_platform() [1/2] template<typename PlatformStorage >

```
auto trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::get_underlying_↩
platform ( )  [inline]
```

Get the platform behind the curtain.

Definition at line 48 of file platform.hpp.
```
00048                                  {
00049     return p;
00050   }
```

References trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::p.

9.7.2.6.2.4 get_underlying_platform() [2/2] template<typename PlatformStorage >

```
auto trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::get_underlying_↩
platform ( )  [inline]
```

Get the platform behind the curtain.

Definition at line 48 of file platform.hpp.
```
00048                                  {
00049     return p;
00050   }
```

References trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::p.

9.7.2.6.3 Member Data Documentation

9.7.2.6.3.1 p `template<typename PlatformStorage >`

`::trisycl::platform trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::p`

`[private]`

The storage-less platform behind the scene.

Definition at line 28 of file platform.hpp.

Referenced by trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::get_underlying_platform().

9.7.2.6.3.2 scope_storage `template<typename PlatformStorage >`

`PlatformStorage trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::scope_storage [private]`

The platform-scoped storage default-initialized.

Todo For now it is allocated on the host in this CPU emulation but a device compiler and run time can create this on a real device

Todo For now it is allocated on the host in this CPU emulation but a device compiler and run time can create this on a real device

Definition at line 36 of file platform.hpp.

Referenced by trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >::get_storage().

9.7.2.7 struct trisycl::vendor::trisycl::scope::empty_queue_scope

An empty queue scope to make compilation errors clearer.

Definition at line 36 of file queue.hpp.

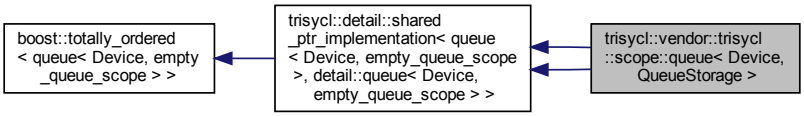
9.7.2.8 class trisycl::vendor::trisycl::scope::queue

template<typename Device, typename QueueStorage = empty_queue_scope>
class trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >

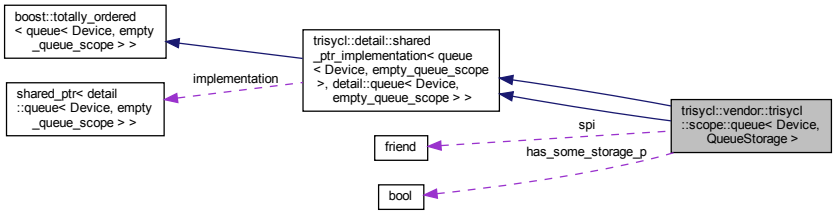
A conceptual queue implementing some queue-scoped storage constructed from a device with some device-scope storage.

Definition at line 45 of file queue.hpp.

Inheritance diagram for trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >:



Collaboration diagram for trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >:



Public Types

- using storage_type = QueueStorage

  The type of the scoped storage in the queue.
- using device_type = Device

  The device type targeted by this queue.
- using platform_type = typename device_type::platform_type

  The platform type targeted by this queue.
- using queue_scope_type = storage_type

  The queue scope storage type.
- using device_scope_type = detail::storage_type_trait_t< device_type >

  The device scope storage type.
- using platform_scope_type = detail::storage_type_trait_t< platform_type >

  The platform scope storage type.
- using storage_type = QueueStorage

  The type of the scoped storage in the queue.
- using device_type = Device

The device type targeted by this queue.

- using platform_type = typename device_type::platform_type

  The platform type targeted by this queue.
- using queue_scope_type = storage_type

  The queue scope storage type.
- using device_scope_type = detail::storage_type_trait_t< device_type >

  The device scope storage type.
- using platform_scope_type = detail::storage_type_trait_t< platform_type >

  The platform scope storage type.

Public Member Functions

- queue (const ::trisycl::queue &q, const Device &d)

  Construct the queue with some queue-scoped storage on top of a SYCL queue.
- queue (const Device &d)

  Construct the queue with some queue-scoped storage on top of a scoped device.
- queue (const queue &)=default

  It is still copyable.
- queue (queue &&)=default

  It is still movable.
- auto & get_storage () const

  Access to the queue-scoped storage.
- auto & get_device () const

  Access to the underlying scoped device.
- auto & get_platform () const

  Access to the underlying scoped platform.
- auto & device_scope ()

  Provide access to the device scope from the host.
- template<typename PS = platform_type, typename E = std::enable_if_t<PS::has_some_storage_p>>
  auto & platform_scope ()

  Provide access to the platform scope from the host, if any.
- operator::trisycl::queue & () const

  Add a conversion to `trisycl::queue&` so it can be used as a normal queue.
- template<typename CG >
  void submit (CG cgh)

  Submit a command-group lambda function.
- void wait ()

  Performs a blocking wait for the completion all enqueued tasks in the queue.
- queue (const ::trisycl::queue &q, const Device &d)

  Construct the queue with some queue-scoped storage on top of a SYCL queue.
- queue (const Device &d)

  Construct the queue with some queue-scoped storage on top of a scoped device.
- queue (const queue &)=default

  It is still copyable.
- queue (queue &&)=default

  It is still movable.
- auto & get_storage () const

  Access to the queue-scoped storage.
- auto & get_device () const

  Access to the underlying scoped device.

- auto & get_platform () const

  Access to the underlying scoped platform.
- auto & device_scope ()

  Provide access to the device scope from the host.
- template<typename PS = platform_type, typename E = std::enable_if_t<PS::has_some_storage_p>>
  auto & platform_scope ()

  Provide access to the platform scope from the host, if any.
- operator::trisycl::queue & () const

  Add a conversion to `trisycl::queue&` so it can be used as a normal queue.
- template<typename CG >
  void submit (CG cgh)

  Submit a command-group lambda function.
- void wait ()

  Performs a blocking wait for the completion all enqueued tasks in the queue.

Public Attributes

- std::shared_ptr< Implementation > implementation

  The implementation forward everything to this... implementation.
- std::shared_ptr< Implementation > implementation

  The implementation forward everything to this... implementation.

Static Public Attributes

- static constexpr bool has_some_storage_p

  True if there is some storage in the queue.

Private Types

- using spi = ::trisycl::detail::shared_ptr_implementation< queue< Device, QueueStorage >, detail::queue< Device, QueueStorage > >
- using spi = ::trisycl::detail::shared_ptr_implementation< queue< Device, QueueStorage >, detail::queue< Device, QueueStorage > >

Private Attributes

- friend spi

9.7.2.8.1 Member Typedef Documentation

9.7.2.8.1.1 device_scope_type [1/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::device_scope_type =
detail::storage_type_trait_t<device_type>
```

The device scope storage type.

Definition at line 83 of file queue.hpp.

9.7.2.8.1.2 device_scope_type [2/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::device_scope_type =
detail::storage_type_trait_t<device_type>
```

The device scope storage type.

Definition at line 83 of file queue.hpp.

9.7.2.8.1.3 device_type [1/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::device_type = Device
```

The device type targeted by this queue.

Definition at line 74 of file queue.hpp.

9.7.2.8.1.4 device_type [2/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::device_type = Device
```

The device type targeted by this queue.

Definition at line 74 of file queue.hpp.

9.7.2.8.1.5 platform_scope_type [1/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::platform_scope_type =
detail::storage_type_trait_t<platform_type>
```

The platform scope storage type.

Definition at line 86 of file queue.hpp.

9.7.2.8.1.6 platform_scope_type [2/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::platform_scope_type =
detail::storage_type_trait_t<platform_type>
```

The platform scope storage type.

Definition at line 86 of file queue.hpp.

9.7.2.8.1.7 platform_type [1/2]

```
template<typename Device , typename QueueStorage = empty_queue↩
_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::platform_type = typename
device_type::platform_type
```

The platform type targeted by this queue.

Definition at line 77 of file queue.hpp.

9.7.2.8.1.8 platform_type [2/2]

```
template<typename Device , typename QueueStorage = empty_queue↩
_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::platform_type = typename
device_type::platform_type
```

The platform type targeted by this queue.

Definition at line 77 of file queue.hpp.

9.7.2.8.1.9 queue_scope_type [1/2]

```
template<typename Device , typename QueueStorage = empty_↩
queue_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::queue_scope_type =
storage_type
```

The queue scope storage type.

Definition at line 80 of file queue.hpp.

9.7.2.8.1.10 queue_scope_type [2/2]

```
template<typename Device , typename QueueStorage = empty↩
_queue_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::queue_scope_type =
storage_type
```

The queue scope storage type.

Definition at line 80 of file queue.hpp.

9.7.2.8.1.11 spi [1/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::spi = ::trisycl::detail::shared_ptr_imp
QueueStorage>, detail::queue <Device, QueueStorage> >  [private]
```

Definition at line 53 of file queue.hpp.

9.7.2.8.1.12 spi [2/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::spi = ::trisycl::detail::shared_ptr_imp
QueueStorage>, detail::queue <Device, QueueStorage> >  [private]
```

Definition at line 53 of file queue.hpp.

9.7.2.8.1.13 storage_type [1/2]

```
template<typename Device , typename QueueStorage = empty_↩
queue_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::storage_type = Queue↩
Storage
```

The type of the scoped storage in the queue.

Definition at line 67 of file queue.hpp.

9.7.2.8.1.14 storage_type [2/2]

```
template<typename Device , typename QueueStorage = empty_↩
queue_scope>
using trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::storage_type = Queue↩
Storage
```

The type of the scoped storage in the queue.

Definition at line 67 of file queue.hpp.

9.7.2.8.2 Constructor & Destructor Documentation

9.7.2.8.2.1 queue() [1/8]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::queue (
            const ::trisycl::queue< Device, QueueStorage > & q,
            const Device & d )  [inline]
```

Construct the queue with some queue-scoped storage on top of a SYCL queue.

Parameters

in	*q*	is the real queue to use
in	*d*	is the a scoped device to use

Definition at line 94 of file queue.hpp.

```
00095     :  spi { new detail::queue<Device, QueueStorage> { q, d } } {}
```

9.7.2.8.2.2 queue() [2/8]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::queue (
            const Device & d )  [inline]
```

Construct the queue with some queue-scoped storage on top of a scoped device.

Parameters

in	*d*	is the a scoped device to use

Definition at line 103 of file queue.hpp.

```
00103 :  queue { ::trisycl::queue { d }, d } {}
```

9.7.2.8.2.3 queue() [3/8] template<typename Device , typename QueueStorage = empty_queue_scope>

```
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::queue (
            const queue< Device, QueueStorage > &  )  [default]
```

It is still copyable.

9.7.2.8.2.4 queue() [4/8] template<typename Device , typename QueueStorage = empty_queue_scope>

```
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::queue (
            queue< Device, QueueStorage > &&  )  [default]
```

It is still movable.

9.7.2.8.2.5 queue() [5/8] template<typename Device , typename QueueStorage = empty_queue_scope>

```
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::queue (
            const ::trisycl::queue< Device, QueueStorage > & q,
            const Device & d )  [inline]
```

Construct the queue with some queue-scoped storage on top of a SYCL queue.

Parameters

in	*q*	is the real queue to use
in	*d*	is the a scoped device to use

Definition at line 94 of file queue.hpp.

```
00095     :  spi { new detail::queue<Device, QueueStorage> { q, d } } {}
```

9.7.2.8.2.6 queue() [6/8] template<typename Device , typename QueueStorage = empty_queue_scope>

```
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::queue (
            const Device & d )  [inline]
```

Construct the queue with some queue-scoped storage on top of a scoped device.

Parameters

in	*d*	is the a scoped device to use

Definition at line 103 of file queue.hpp.
```
00103 :  queue { ::trisycl::queue { d }, d } {}
```

9.7.2.8.2.7 queue() [7/8]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::queue (
            const queue< Device, QueueStorage > &  )  [default]
```

It is still copyable.

9.7.2.8.2.8 queue() [8/8]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::queue (
            queue< Device, QueueStorage > &&  )  [default]
```

It is still movable.

9.7.2.8.3 Member Function Documentation

9.7.2.8.3.1 device_scope() [1/2]

```
template<typename Device , typename QueueStorage = empty_↩
queue_scope>
auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::device_scope ( )  [inline]
```

Provide access to the device scope from the host.

Definition at line 133 of file queue.hpp.
```
00133 { return get_device().get_storage(); }
```

9.7.2.8.3.2 device_scope() [2/2]

```
template<typename Device , typename QueueStorage = empty_↩
queue_scope>
auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::device_scope ( )  [inline]
```

Provide access to the device scope from the host.

Definition at line 133 of file queue.hpp.
```
00133 { return get_device().get_storage(); }
```

9.7.2.8.3.3 get_device() [1/2]

```
template<typename Device , typename QueueStorage = empty_queue_↩
scope>
auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::get_device ( ) const
[inline]
```

Access to the underlying scoped device.

Definition at line 121 of file queue.hpp.
```
00121                          {
00122     return implementation->get_device();
00123   }
```

9.7.2.8.3.4 get_device() [2/2] template<typename Device , typename QueueStorage = empty_queue_↩ scope>

auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::get_device ( ) const [inline]

Access to the underlying scoped device.

Definition at line 121 of file queue.hpp.

```
00121                                {
00122     return implementation->get_device();
00123   }
```

9.7.2.8.3.5 get_platform() [1/2] template<typename Device , typename QueueStorage = empty_queue↩ _scope>

auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::get_platform ( ) const [inline]

Access to the underlying scoped platform.

Definition at line 127 of file queue.hpp.

```
00127                                  {
00128     return get_device().get_platform();
00129   }
```

9.7.2.8.3.6 get_platform() [2/2] template<typename Device , typename QueueStorage = empty_queue↩ _scope>

auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::get_platform ( ) const [inline]

Access to the underlying scoped platform.

Definition at line 127 of file queue.hpp.

```
00127                                  {
00128     return get_device().get_platform();
00129   }
```

9.7.2.8.3.7 get_storage() [1/2] template<typename Device , typename QueueStorage = empty_queue↩ _scope>

auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::get_storage ( ) const [inline]

Access to the queue-scoped storage.

Definition at line 115 of file queue.hpp.

```
00115                                {
00116     return implementation->get_storage();
00117   }
```

9.7.2.8.3.8 get_storage() [2/2]

```
template<typename Device , typename QueueStorage = empty_queue↩
_scope>
auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::get_storage ( ) const
[inline]
```

Access to the queue-scoped storage.

Definition at line 115 of file queue.hpp.

```
00115                                {
00116     return implementation->get_storage();
00117   }
```

9.7.2.8.3.9 operator::trisycl::queue &() [1/2]

```
template<typename Device , typename QueueStorage =
empty_queue_scope>
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::operator::trisycl::queue & ( )
const  [inline]
```

Add a conversion to `trisycl::queue&` so it can be used as a normal queue.

Definition at line 143 of file queue.hpp.

```
00143                                    {
00144     return implementation->get_underlying_queue();
00145   }
```

9.7.2.8.3.10 operator::trisycl::queue &() [2/2]

```
template<typename Device , typename QueueStorage =
empty_queue_scope>
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::operator::trisycl::queue & ( )
const  [inline]
```

Add a conversion to `trisycl::queue&` so it can be used as a normal queue.

Definition at line 143 of file queue.hpp.

```
00143                                    {
00144     return implementation->get_underlying_queue();
00145   }
```

9.7.2.8.3.11 platform_scope() [1/2]

```
template<typename Device , typename QueueStorage = empty_↩
queue_scope>
template<typename PS = platform_type, typename E = std::enable_if_t<PS::has_some_storage_p>>
auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::platform_scope ( )
[inline]
```

Provide access to the platform scope from the host, if any.

Definition at line 138 of file queue.hpp.

```
00138 { return get_platform().get_storage(); }
```

9.7.2.8.3.12 platform_scope() [2/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename PS = platform_type, typename E = std::enable_if_t<PS::has_some_storage_p>>
auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::platform_scope ( )
[inline]
```

Provide access to the platform scope from the host, if any.

Definition at line 138 of file queue.hpp.

```
00138 { return get_platform().get_storage(); }
```

9.7.2.8.3.13 submit() [1/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CG >
void trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::submit (
            CG cgh )  [inline]
```

Submit a command-group lambda function.

Definition at line 200 of file queue.hpp.

```
00200                       {
00201     implementation->get_underlying_queue()
00202       .submit([&] (::trisycl::handler &cgh_generic) {
00203           command_group<CG> cg { cgh_generic, get_device() };
00204           cgh(cg);
00205         });
00206   }
```

9.7.2.8.3.14 submit() [2/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CG >
void trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::submit (
            CG cgh )  [inline]
```

Submit a command-group lambda function.

Definition at line 200 of file queue.hpp.

```
00200                       {
00201     implementation->get_underlying_queue()
00202       .submit([&] (::trisycl::handler &cgh_generic) {
00203           command_group<CG> cg { cgh_generic, get_device() };
00204           cgh(cg);
00205         });
00206   }
```

9.7.2.8.3.15 wait() [1/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
void trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::wait ( )  [inline]
```

Performs a blocking wait for the completion all enqueued tasks in the queue.

Definition at line 211 of file queue.hpp.

```
00211                 {
00212     implementation->get_underlying_queue().wait();
00213   }
```

9.7.2.8.3.16 wait() [2/2] template<typename Device , typename QueueStorage = empty_queue_scope> void trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::wait ( ) [inline]

Performs a blocking wait for the completion all enqueued tasks in the queue.

Definition at line 211 of file queue.hpp.

```
00211               {
00212     implementation->get_underlying_queue().wait();
00213   }
```

9.7.2.8.4 Member Data Documentation

9.7.2.8.4.1 has_some_storage_p template<typename Device , typename QueueStorage = empty_↩ queue_scope> constexpr bool trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::has_some_↩ storage_p [static], [constexpr]

Initial value:

```
=
    not std::is_same_v<QueueStorage, empty_queue_scope>
```

True if there is some storage in the queue.

Definition at line 70 of file queue.hpp.

9.7.2.8.4.2 implementation [1/2] template<typename Device , typename QueueStorage = empty_↩ queue_scope> std::shared_ptr<Implementation> trisycl::detail::shared_ptr_implementation< Parent, Implementation >::implementation

The implementation forward everything to this... implementation.

Definition at line 41 of file shared_ptr_implementation.hpp.

9.7.2.8.4.3 implementation [2/2] template<typename Device , typename QueueStorage = empty_↩ queue_scope> std::shared_ptr<Implementation> trisycl::detail::shared_ptr_implementation< Parent, Implementation >::implementation

The implementation forward everything to this... implementation.

Definition at line 41 of file shared_ptr_implementation.hpp.

9.7.2.8.4.4 spi template<typename Device , typename QueueStorage = empty_queue_scope> friend trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::spi [private]

Definition at line 59 of file queue.hpp.

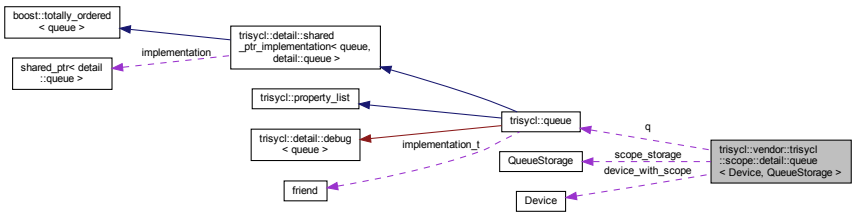
9.7.2.9 class trisycl::vendor::trisycl::scope::detail::queue

template<typename Device, typename QueueStorage>
class trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >

A conceptual queue implementing some queue-scoped storage constructed from a device with some device-scope storage.

Definition at line 29 of file queue.hpp.

Collaboration diagram for trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >:



Public Member Functions

- queue (const ::trisycl::queue &q)

  Construct the queue with some queue-scoped storage on top of a SYCL queue.
- queue (const ::trisycl::queue &q, const Device &d)

  Construct the queue with some queue-scoped storage on top of a SYCL queue.
- auto get_underlying_queue ()

  Get the queue behind the curtain.
- auto & get_storage ()

  Access to the queue-scoped storage.
- auto & get_device ()

  Access to the underlying scoped device.
- queue (const ::trisycl::queue &q)

  Construct the queue with some queue-scoped storage on top of a SYCL queue.
- queue (const ::trisycl::queue &q, const Device &d)

  Construct the queue with some queue-scoped storage on top of a SYCL queue.
- auto get_underlying_queue ()

  Get the queue behind the curtain.
- auto & get_storage ()

  Access to the queue-scoped storage.
- auto & get_device ()

  Access to the underlying scoped device.

Private Attributes

- ::trisycl::queue q

  The storage-less queue behind the scene.
- QueueStorage scope_storage

  The queue-scoped storage default-initialized.
- Device device_with_scope

  The scoped device the scoped queue is built into.

9.7.2.9.1 Constructor & Destructor Documentation

9.7.2.9.1.1 queue() [1/4]

```
template<typename Device , typename QueueStorage >
trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::queue (
            const ::trisycl::queue< Device, QueueStorage > & q )  [inline]
```

Construct the queue with some queue-scoped storage on top of a SYCL queue.

Parameters

in	*q*	is the real queue to use

Definition at line 51 of file queue.hpp.

```
00051 : q { q } {}
```

9.7.2.9.1.2 queue() [2/4]

```
template<typename Device , typename QueueStorage >
trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::queue (
            const ::trisycl::queue< Device, QueueStorage > & q,
            const Device & d )  [inline]
```

Construct the queue with some queue-scoped storage on top of a SYCL queue.

Parameters

in	*q*	is the real queue to use
in	*d*	is the a scoped device to use

Definition at line 60 of file queue.hpp.

```
00061     : q { q }, device_with_scope { d } {}
```

9.7.2.9.1.3 queue() [3/4]

```
template<typename Device , typename QueueStorage >
trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::queue (
            const ::trisycl::queue< Device, QueueStorage > & q )  [inline]
```

Construct the queue with some queue-scoped storage on top of a SYCL queue.

Parameters

in	*q*	is the real queue to use

Definition at line 51 of file queue.hpp.

```
00051 : q { q } {}
```

9.7.2.9.1.4 queue() [4/4]

```
template<typename Device , typename QueueStorage >
trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::queue (
```

```
const ::trisycl::queue< Device, QueueStorage > & q,
const Device & d )  [inline]
```

Construct the queue with some queue-scoped storage on top of a SYCL queue.

Parameters

in	*q*	is the real queue to use
in	*d*	is the a scoped device to use

Definition at line 60 of file queue.hpp.

```
00061     : q { q }, device_with_scope { d } {}
```

9.7.2.9.2 Member Function Documentation

9.7.2.9.2.1 get_device() [1/2] template<typename Device , typename QueueStorage >

```
auto & trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::get_device ( )
[inline]
```

Access to the underlying scoped device.

Definition at line 77 of file queue.hpp.

```
00077                       {
00078     return device_with_scope;
00079   }
```

References trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::device_with_scope.

9.7.2.9.2.2 get_device() [2/2] template<typename Device , typename QueueStorage >

```
auto & trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::get_device ( )
[inline]
```

Access to the underlying scoped device.

Definition at line 77 of file queue.hpp.

```
00077                       {
00078     return device_with_scope;
00079   }
```

References trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::device_with_scope.

9.7.2.9.2.3 get_storage() [1/2] template<typename Device , typename QueueStorage >

```
auto & trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::get_storage ( )
[inline]
```

Access to the queue-scoped storage.

Definition at line 71 of file queue.hpp.

```
00071                       {
00072     return scope_storage;
00073   }
```

References trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::scope_storage.

9.7.2.9.2.4 get_storage() [2/2] template<typename Device , typename QueueStorage >

auto & trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::get_storage ( ) [inline]

Access to the queue-scoped storage.

Definition at line 71 of file queue.hpp.

```
00071                       {
00072     return scope_storage;
00073   }
```

References trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::scope_storage.

9.7.2.9.2.5 get_underlying_queue() [1/2] template<typename Device , typename QueueStorage >

auto trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::get_underlying_↩ queue ( ) [inline]

Get the queue behind the curtain.

Definition at line 65 of file queue.hpp.

```
00065                               {
00066     return q;
00067   }
```

References trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::q.

9.7.2.9.2.6 get_underlying_queue() [2/2] template<typename Device , typename QueueStorage >

auto trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::get_underlying_↩ queue ( ) [inline]

Get the queue behind the curtain.

Definition at line 65 of file queue.hpp.

```
00065                               {
00066     return q;
00067   }
```

References trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::q.

9.7.2.9.3 Member Data Documentation

9.7.2.9.3.1 device_with_scope template<typename Device , typename QueueStorage >

Device trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::device_with_↩ scope [private]

The scoped device the scoped queue is built into.

Definition at line 42 of file queue.hpp.

Referenced by trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::get_device().

9.7.2.9.3.2 q `template<typename Device , typename QueueStorage > ::trisycl::queue trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::q [private]`

The storage-less queue behind the scene.

Definition at line 31 of file queue.hpp.

Referenced by trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::get_underlying_queue().

9.7.2.9.3.3 scope_storage `template<typename Device , typename QueueStorage > QueueStorage trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::scope_storage [private]`

The queue-scoped storage default-initialized.

Todo For now it is allocated on the host in this CPU emulation but a device compiler and run time can create this on a real device

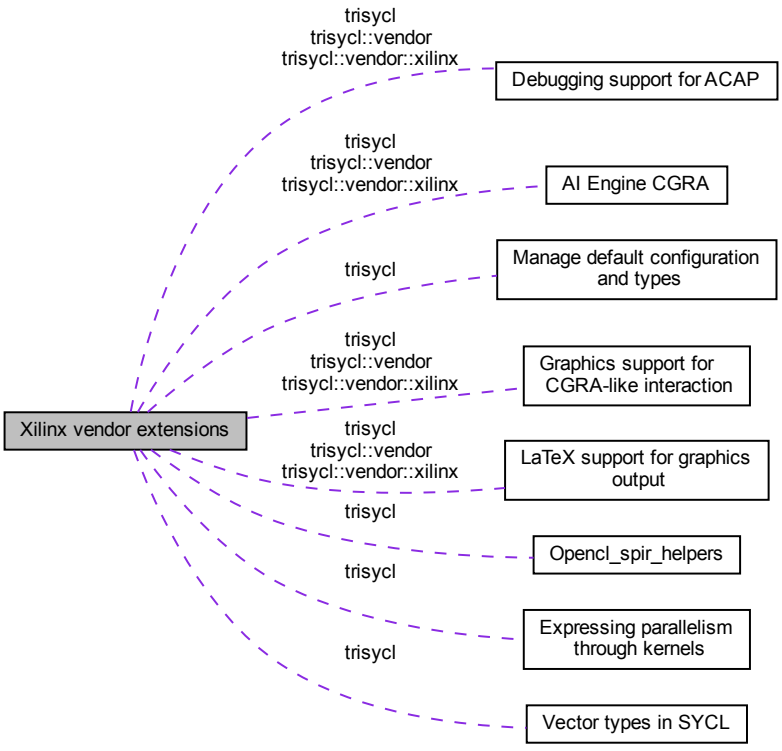
Todo For now it is allocated on the host in this CPU emulation but a device compiler and run time can create this on a real device

Definition at line 39 of file queue.hpp.

Referenced by trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >::get_storage().

9.8 Xilinx vendor extensions

Collaboration diagram for Xilinx vendor extensions:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::partition

  Kind of array partition.

Classes

- struct trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >

  Define an array class with partition feature. More...

Macros

- #define _ssdm_op_SpecDataflowPipeline(...) do { } while (0)
- #define _ssdm_op_SpecPipeline(...) do { } while (0)
- #define _ssdm_SpecArrayPartition(...) do { } while (0)
- #define _ssdm_op_SpecDataflowPipeline(...) do { } while (0)
- #define _ssdm_op_SpecPipeline(...) do { } while (0)
- #define _ssdm_SpecArrayPartition(...) do { } while (0)

9.8.1 Detailed Description

9.8.2 Class Documentation

9.8.2.1 struct trisycl::vendor::xilinx::partition_array

template<typename ValueType, std::size_t Size, typename PartitionType = partition::none>
struct trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >

Define an array class with partition feature.

Since on FPGA, users can customize the memory architecture in the system and within the CU. Array partition help us to partition single array to multiple memories that can be accessed simultaneously getting higher memory bandwidth.

Parameters

ValueType	is the type of element.
Size	is the size of the array.
PartitionType	is the array partition type: cyclic, block, and complete. The default type is none.

Todo Deal with multi-dimension array.

Since on FPGA, users can customize the memory architecture in the system and within the CU. Array partition

help us to partition single array to multiple memories that can be accessed simultaneously getting higher memory bandwidth.

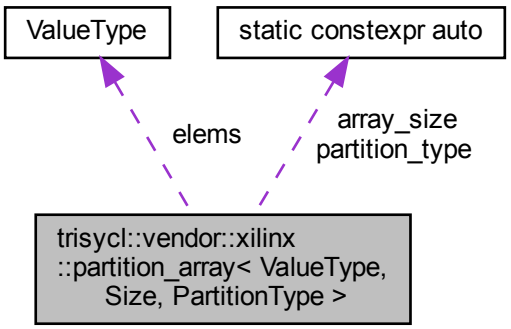
Parameters

ValueType	is the type of element.
Size	is the size of the array.
PartitionType	is the array partition type: cyclic, block, and complete. The default type is none.

Todo Deal with multi-dimension array.

Definition at line 154 of file partition_array.hpp.

Collaboration diagram for trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >:



Public Types

- using element_type = ValueType
  Type of array elements.
- using element_type = ValueType
  Type of array elements.

Public Member Functions

- auto begin ()
  Provide iterator.
- const auto begin () const
- auto end ()
- const auto end () const
- constexpr auto size () const noexcept
  Evaluate size.
- partition_array ()
  Construct an array.

- template<typename SomeContainer >
  partition_array (const SomeContainer &src)

  A constructor from some container.
- template<typename SourceBasicType , typename = std::enable_if_t<std::is_convertible<SourceBasicType, ValueType>::value>>
  constexpr partition_array (std::initializer_list< SourceBasicType > l)

  Construct an array from initializer_list.
- constexpr ValueType & operator[ ] (std::size_t i)

  Provide a subscript operator.
- constexpr const ValueType & operator[ ] (std::size_t i) const
- constexpr auto get_partition_type () const

  Return the partition type of the array.
- auto begin ()

  Provide iterator.
- const auto begin () const
- auto end ()
- const auto end () const
- constexpr auto size () const noexcept

  Evaluate size.
- partition_array ()

  Construct an array.
- template<typename SomeContainer >
  partition_array (const SomeContainer &src)

  A constructor from some container.
- template<typename SourceBasicType , typename = std::enable_if_t<std::is_convertible<SourceBasicType, ValueType>::value>>
  constexpr partition_array (std::initializer_list< SourceBasicType > l)

  Construct an array from initializer_list.
- constexpr ValueType & operator[ ] (std::size_t i)

  Provide a subscript operator.
- constexpr const ValueType & operator[ ] (std::size_t i) const
- constexpr auto get_partition_type () const

  Return the partition type of the array.

Public Attributes

- ValueType elems [Size]

  Store the array elements.

Static Public Attributes

- static constexpr auto array_size = Size

  The number of elements of the 1-D array.
- static constexpr auto partition_type = PartitionType::partition_type

  The kind of partitioning.

9.8.2.1.1 Member Typedef Documentation

9.8.2.1.1.1 element_type [1/2]

```
template<typename ValueType , std::size_t Size, typename Partition↩
Type = partition::none>
using trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::element_type
= ValueType
```

Type of array elements.

Definition at line 166 of file partition_array.hpp.

9.8.2.1.1.2 element_type [2/2]

```
template<typename ValueType , std::size_t Size, typename Partition↩
Type = partition::none>
using trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::element_type
= ValueType
```

Type of array elements.

Definition at line 166 of file partition_array.hpp.

9.8.2.1.2 Constructor & Destructor Documentation

9.8.2.1.2.1 partition_array() [1/6]

```
template<typename ValueType , std::size_t Size, typename
PartitionType = partition::none>
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_array (
)  [inline]
```

Construct an array.

Definition at line 183 of file partition_array.hpp.

```
00183                     {
00184     // Add the intrinsic according expressing to the target compiler the
00185     // partitioning to use
00186     if constexpr (partition_type == partition::type::cyclic)
00187       _ssdm_SpecArrayPartition(&(*this)[0], PartitionType::partition_dim,
00188                                "CYCLIC", PartitionType::physical_mem_num, "");
00189     if constexpr (partition_type == partition::type::block)
00190       _ssdm_SpecArrayPartition(&(*this)[0], PartitionType::partition_dim,
00191                                "BLOCK", PartitionType::ele_in_each_physical_mem,
00192                                "");
00193     if constexpr (partition_type == partition::type::complete)
00194       _ssdm_SpecArrayPartition(&(*this)[0], PartitionType::partition_dim,
00195                                "COMPLETE", 0, "");
00196   }
```

References _ssdm_SpecArrayPartition, trisycl::vendor::xilinx::partition::block, trisycl::vendor::xilinx::partition::complete, trisycl::vendor::xilinx::partition::cyclic, and trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_type.

9.8.2.1.2.2 partition_array() [2/6]

```
template<typename ValueType , std::size_t Size, typename
PartitionType = partition::none>
template<typename SomeContainer >
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_array (
            const SomeContainer & src )  [inline]
```

A constructor from some container.

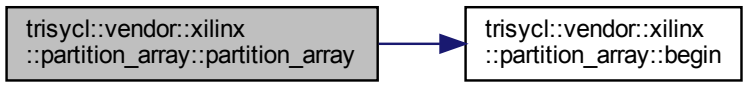
Todo Find a way to specialize this with a safer implementation when the size of src is at least constexpr

Definition at line 201 of file partition_array.hpp.

```
00202     : partition_array { } {
00203 /// \todo Find a way to specialize this with a safer
00204 /// implementation when the size of src is at least constexpr
00205     std::copy_n(std::begin(src), Size, begin());
00206   }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::begin(), and src.

Here is the call graph for this function:



9.8.2.1.2.3 partition_array() [3/6]

```
template<typename ValueType , std::size_t Size, typename
PartitionType = partition::none>
template<typename SourceBasicType , typename = std::enable_if_t<std::is_convertible<Source↩
BasicType, ValueType>::value>>
constexpr trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition↩
_array (
            std::initializer_list< SourceBasicType > l )  [inline], [constexpr]
```

Construct an array from initializer_list.

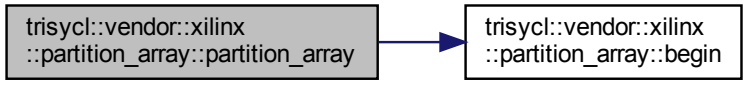
Todo Find a way to specialize this with a safer implementation when the size of src is at least constexpr This does not work... static_assert(l.size() == Size);

Definition at line 216 of file partition_array.hpp.

```
00217     : partition_array { } {
00218 /// \todo Find a way to specialize this with a safer
00219 /// implementation when the size of src is at least constexpr
00220 /// This does not work...
00221 /// static_assert(l.size() == Size);
00222     std::copy_n(std::begin(l), Size, begin());
00223   }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::begin().

Here is the call graph for this function:



9.8.2.1.2.4 partition_array() [4/6] template<typename ValueType , std::size_t Size, typename PartitionType = partition::none>

trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_array ( ) [inline]

Construct an array.

Definition at line 183 of file partition_array.hpp.

```
00183                   {
00184     // Add the intrinsic according expressing to the target compiler the
00185     // partitioning to use
00186     if constexpr (partition_type == partition::type::cyclic)
00187       _ssdm_SpecArrayPartition(&(*this)[0], PartitionType::partition_dim,
00188                                "CYCLIC", PartitionType::physical_mem_num, "");
00189     if constexpr (partition_type == partition::type::block)
00190       _ssdm_SpecArrayPartition(&(*this)[0], PartitionType::partition_dim,
00191                                "BLOCK", PartitionType::ele_in_each_physical_mem,
00192                                "");
00193     if constexpr (partition_type == partition::type::complete)
00194       _ssdm_SpecArrayPartition(&(*this)[0], PartitionType::partition_dim,
00195                                "COMPLETE", 0, "");
00196   }
```

References _ssdm_SpecArrayPartition, trisycl::vendor::xilinx::partition::block, trisycl::vendor::xilinx::partition::complete, trisycl::vendor::xilinx::partition::cyclic, and trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_type.

9.8.2.1.2.5 partition_array() [5/6] template<typename ValueType , std::size_t Size, typename PartitionType = partition::none> template<typename SomeContainer >

trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_array (const SomeContainer & *src*) [inline]

A constructor from some container.

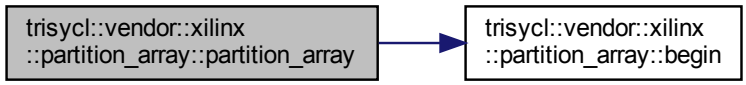
Todo Find a way to specialize this with a safer implementation when the size of src is at least constexpr

Definition at line 201 of file partition_array.hpp.

```
00202     : partition_array { } {
00203 /// \todo Find a way to specialize this with a safer
00204 /// implementation when the size of src is at least constexpr
00205     std::copy_n(std::begin(src), Size, begin());
00206   }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::begin(), and src.

Here is the call graph for this function:



9.8.2.1.2.6 partition_array() [6/6]

```
template<typename ValueType , std::size_t Size, typename
PartitionType = partition::none>
template<typename SourceBasicType , typename = std::enable_if_t<std::is_convertible<Source↩
BasicType, ValueType>::value>>
constexpr trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition↩
_array (
            std::initializer_list< SourceBasicType > l )  [inline], [constexpr]
```

Construct an array from initializer_list.

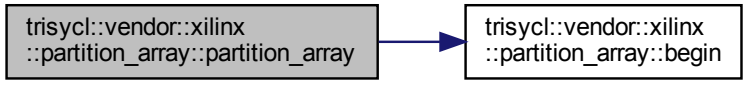
Todo Find a way to specialize this with a safer implementation when the size of src is at least constexpr This does not work... static_assert(l.size() == Size);

Definition at line 216 of file partition_array.hpp.

```
00217     : partition_array { } {
00218 /// \todo Find a way to specialize this with a safer
00219 /// implementation when the size of src is at least constexpr
00220 /// This does not work...
00221 /// static_assert(l.size() == Size);
00222     std::copy_n(std::begin(l), Size, begin());
00223   }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::begin().

Here is the call graph for this function:



9.8.2.1.3 Member Function Documentation

9.8.2.1.3.1 begin() [1/4]

```
template<typename ValueType , std::size_t Size, typename PartitionType
= partition::none>
auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::begin ( )
[inline]
```

Provide iterator.

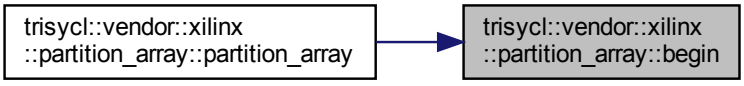
Definition at line 170 of file partition_array.hpp.

```
00170 { return elems; }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::elems.

Referenced by trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_array().

Here is the caller graph for this function:



9.8.2.1.3.2 begin() [2/4]

```
template<typename ValueType , std::size_t Size, typename PartitionType
= partition::none>
auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::begin ( )
[inline]
```

Provide iterator.

Definition at line 170 of file partition_array.hpp.

```
00170 { return elems; }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::elems.

9.8.2.1.3.3 begin() [3/4]

```
template<typename ValueType , std::size_t Size, typename PartitionType
= partition::none>
const auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::begin (
) const  [inline]
```

Definition at line 171 of file partition_array.hpp.

```
00171 { return elems; }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::elems.

9.8.2.1.3.4 begin() [4/4]

```
template<typename ValueType , std::size_t Size, typename PartitionType
= partition::none>
const auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::begin (
) const  [inline]
```

Definition at line 171 of file partition_array.hpp.

```
00171 { return elems; }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::elems.

9.8.2.1.3.5 end() [1/4]

```
template<typename ValueType , std::size_t Size, typename PartitionType
= partition::none>
auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::end ( )
[inline]
```

Definition at line 172 of file partition_array.hpp.

```
00172 { return elems+Size; }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::elems.

9.8.2.1.3.6 end() [2/4]

```
template<typename ValueType , std::size_t Size, typename PartitionType
= partition::none>
auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::end ( )
[inline]
```

Definition at line 172 of file partition_array.hpp.

```
00172 { return elems+Size; }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::elems.

9.8.2.1.3.7 end() [3/4]

```
template<typename ValueType , std::size_t Size, typename PartitionType
= partition::none>
const auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::end ( )
const  [inline]
```

Definition at line 173 of file partition_array.hpp.

```
00173 { return elems+Size; }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::elems.

9.8.2.1.3.8 end() [4/4]

```
template<typename ValueType , std::size_t Size, typename PartitionType
= partition::none>
const auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::end ( )
const  [inline]
```

Definition at line 173 of file partition_array.hpp.

```
00173 { return elems+Size; }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::elems.

9.8.2.1.3.9 get_partition_type() [1/2]

```
template<typename ValueType , std::size_t Size, typename
PartitionType = partition::none>
constexpr auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >↩
::get_partition_type ( ) const  [inline], [constexpr]
```

Return the partition type of the array.

Definition at line 238 of file partition_array.hpp.

```
00238                                       {
00239     return partition_type;
00240   }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_type.

9.8.2.1.3.10 get_partition_type() [2/2]

```
template<typename ValueType , std::size_t Size, typename
PartitionType = partition::none>
constexpr auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >↩
::get_partition_type ( ) const  [inline], [constexpr]
```

Return the partition type of the array.

Definition at line 238 of file partition_array.hpp.

```
00238                                       {
00239     return partition_type;
00240   }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_type.

9.8.2.1.3.11 operator[]() [1/4]

```
template<typename ValueType , std::size_t Size, typename Partition↩
Type = partition::none>
constexpr ValueType & trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType
>::operator[] (
             std::size_t i )  [inline], [constexpr]
```

Provide a subscript operator.

Definition at line 227 of file partition_array.hpp.

```
00227                                         {
00228     return elems[i];
00229   }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::elems, and i.

9.8.2.1.3.12 operator[]() [2/4]

```
template<typename ValueType , std::size_t Size, typename Partition↩
Type = partition::none>
constexpr ValueType & trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType
>::operator[] (
             std::size_t i )  [inline], [constexpr]
```

Provide a subscript operator.

Definition at line 227 of file partition_array.hpp.

```
00227                                         {
00228     return elems[i];
00229   }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::elems, and i.

9.8.2.1.3.13 operator[]() [3/4] template<typename ValueType , std::size_t Size, typename Partition↩ Type = partition::none> constexpr const ValueType & trisycl::vendor::xilinx::partition_array< ValueType, Size, Partition↩ Type >::operator[] (std::size_t *i*) const [inline], [constexpr]

Definition at line 232 of file partition_array.hpp.

```
00232                                                  {
00233     return elems[i];
00234   }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::elems, and i.

9.8.2.1.3.14 operator[]() [4/4] template<typename ValueType , std::size_t Size, typename Partition↩ Type = partition::none> constexpr const ValueType & trisycl::vendor::xilinx::partition_array< ValueType, Size, Partition↩ Type >::operator[] (std::size_t *i*) const [inline], [constexpr]

Definition at line 232 of file partition_array.hpp.

```
00232                                                  {
00233     return elems[i];
00234   }
```

References trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::elems, and i.

9.8.2.1.3.15 size() [1/2] template<typename ValueType , std::size_t Size, typename PartitionType = partition::none> constexpr auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >↩ ::size ( ) const [inline], [constexpr], [noexcept]

Evaluate size.

Definition at line 177 of file partition_array.hpp.

```
00177                                       {
00178     return Size;
00179   }
```

9.8.2.1.3.16 size() [2/2] template<typename ValueType , std::size_t Size, typename PartitionType = partition::none> constexpr auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >↩ ::size ( ) const [inline], [constexpr], [noexcept]

Evaluate size.

Definition at line 177 of file partition_array.hpp.

```
00177                                       {
00178     return Size;
00179   }
```

9.8.2.1.4 Member Data Documentation

9.8.2.1.4.1 array_size

```
template<typename ValueType , std::size_t Size, typename PartitionType =
partition::none>
static constexpr auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType
>::array_size = Size  [static], [constexpr]
```

The number of elements of the 1-D array.

Definition at line 162 of file partition_array.hpp.

9.8.2.1.4.2 elems

```
template<typename ValueType , std::size_t Size, typename PartitionType =
partition::none>
ValueType trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::elems
```

Store the array elements.

Note that it means default initialization for partition_array elements, which is lazily convenient for heterogeneous computing

Definition at line 160 of file partition_array.hpp.

Referenced by trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::begin(), trisycl::vendor::xilinx::partition_array<
and trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::operator[]().

9.8.2.1.4.3 partition_type

```
template<typename ValueType , std::size_t Size, typename Partition↩
Type = partition::none>
static constexpr auto trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType
>::partition_type = PartitionType::partition_type  [static], [constexpr]
```

The kind of partitioning.

Definition at line 164 of file partition_array.hpp.

Referenced by trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::get_partition_type(), and trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_array().

9.8.3 Macro Definition Documentation

9.8.3.1 _ssdm_op_SpecDataflowPipeline [1/2]

```
#define _ssdm_op_SpecDataflowPipeline(
              ...  ) do { } while (0)
```

```
#include <include/sycl/vendor/Xilinx/fpga/ssdm_inst.hpp>
```

Definition at line 29 of file ssdm_inst.hpp.

9.8.3.2 _ssdm_op_SpecDataflowPipeline [2/2]

```
#define _ssdm_op_SpecDataflowPipeline(
              ...  ) do { } while (0)
```

```
#include <include/triSYCL/vendor/Xilinx/fpga/ssdm_inst.hpp>
```

Definition at line 29 of file ssdm_inst.hpp.

9.8.3.3 _ssdm_op_SpecPipeline [1/2]

```
#define _ssdm_op_SpecPipeline(
              ...  ) do { } while (0)
```

```
#include <include/sycl/vendor/Xilinx/fpga/ssdm_inst.hpp>
```

Definition at line 30 of file ssdm_inst.hpp.

9.8.3.4 _ssdm_op_SpecPipeline [2/2]

```
#define _ssdm_op_SpecPipeline(
              ...  ) do { } while (0)
```

```
#include <include/triSYCL/vendor/Xilinx/fpga/ssdm_inst.hpp>
```

Definition at line 30 of file ssdm_inst.hpp.

9.8.3.5 _ssdm_SpecArrayPartition [1/2]

```
#define _ssdm_SpecArrayPartition(
              ...  ) do { } while (0)
```

```
#include <include/sycl/vendor/Xilinx/fpga/ssdm_inst.hpp>
```

Definition at line 31 of file ssdm_inst.hpp.

9.8.3.6 _ssdm_SpecArrayPartition [2/2]

```
#define _ssdm_SpecArrayPartition(
              ...  ) do { } while (0)
```

```
#include <include/triSYCL/vendor/Xilinx/fpga/ssdm_inst.hpp>
```

Definition at line 31 of file ssdm_inst.hpp.

9.9 Data access and storage in old version of SYCL

Classes

- class trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >

  The pipe accessor abstracts the way pipe data are accessed inside a kernel. More...
- class trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >

  The pipe accessor abstracts the way pipe data are accessed inside a kernel. More...
- class trisycl::host_accessor< DataType, Dimensions, AccessMode >

  SYCL 2020 host_accessor. More...
- class trisycl::sycl_2_2::pipe< T >

  A SYCL pipe. More...
- struct trisycl::detail::sycl_2_2::reserve_id< T >

  A private description of a reservation station. More...
- class trisycl::detail::sycl_2_2::pipe< T >

  Implement a pipe object. More...
- class trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >

  The accessor abstracts the way pipe data are accessed inside a kernel. More...
- struct trisycl::sycl_2_2::pipe_reservation< PipeAccessor >

  The pipe reservation station allows to reserve an array-like view inside the pipe for ordered race-free access from various work-items for example. More...
- class trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >

  The implementation of the pipe reservation station. More...
- class trisycl::sycl_2_2::static_pipe< T, Capacity >

  A SYCL static-scoped pipe equivalent to an OpenCL program-scoped pipe. More...

Functions

- template<typename Accessor >
  static auto & trisycl::get_pipe_detail (Accessor &a)

  Top-level function to break circular dependencies on the the types to get the pipe implementation.

9.9.1 Detailed Description

9.9.2 Class Documentation

9.9.2.1 class trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >

template<typename DataType, access::mode AccessMode>
class trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >

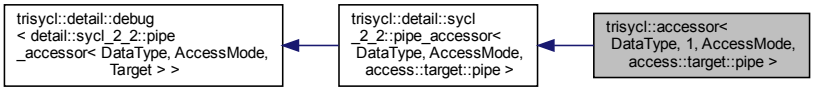
The pipe accessor abstracts the way pipe data are accessed inside a kernel.

This is a proposal for the now abandoned SYCL 2.2 provisional specification. This is still here for historical reasons.

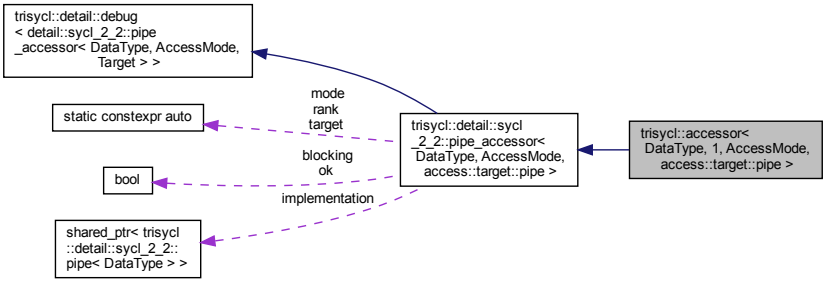
A specialization for an non-blocking pipe

Definition at line 402 of file accessor.hpp.

Inheritance diagram for trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >:



Collaboration diagram for trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >:



Public Types

- using accessor_detail = detail::sycl_2_2::pipe_accessor< DataType, AccessMode, access::target::pipe >

Public Member Functions

- accessor (sycl_2_2::pipe< DataType > &p, handler &command_group_handler)

  Construct a pipe accessor from a pipe using a command group handler object from the command group scope.
- sycl_2_2::pipe_reservation< accessor > reserve (std::size_t size) const

  Make a reservation inside the pipe.
- auto & get_pipe_detail ()

  Get the underlying pipe implementation.

Additional Inherited Members

9.9.2.1.1 Member Typedef Documentation

9.9.2.1.1.1 accessor_detail

```
template<typename DataType , access::mode AccessMode>
using trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >::accessor_detail =
detail::sycl_2_2::pipe_accessor<DataType, AccessMode, access::target::pipe>
```

Definition at line 408 of file accessor.hpp.

9.9.2.1.2 Constructor & Destructor Documentation

9.9.2.1.2.1 accessor() template<typename DataType , access::mode AccessMode> trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >::accessor (sycl_2_2::pipe< DataType > & *p*, handler & *command_group_handler*) [inline]

Construct a pipe accessor from a pipe using a command group handler object from the command group scope.

access_target defines the form of access being obtained.

Definition at line 418 of file accessor.hpp.

```
00419     : accessor_detail { p.implementation, command_group_handler } { }
```

9.9.2.1.3 Member Function Documentation

9.9.2.1.3.1 get_pipe_detail() template<typename DataType , access::mode AccessMode> auto & trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >::get_pipe_detail ( ) [inline]

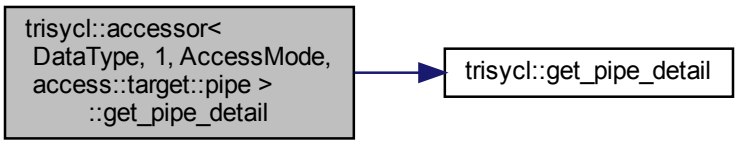
Get the underlying pipe implementation.

Definition at line 428 of file accessor.hpp.

```
00428                                {
00429     return accessor_detail::get_pipe_detail();
00430   }
```

References trisycl::get_pipe_detail().

Here is the call graph for this function:



9.9.2.1.3.2 reserve() template<typename DataType , access::mode AccessMode> sycl_2_2::pipe_reservation< accessor > trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >::reserve (std::size_t *size*) const [inline]

Make a reservation inside the pipe.

Definition at line 422 of file accessor.hpp.

```
00422                                                                {
00423     return accessor_detail::reserve(size);
00424   }
```

9.9.2.2 class trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >

template<typename DataType, access::mode AccessMode>
class trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >

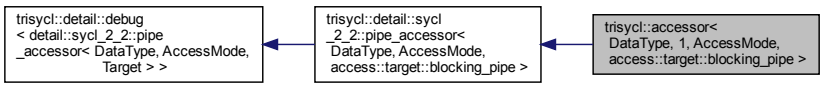
The pipe accessor abstracts the way pipe data are accessed inside a kernel.

This is a proposal for the now abandoned SYCL 2.2 provisional specification. This is still here for historical reasons.

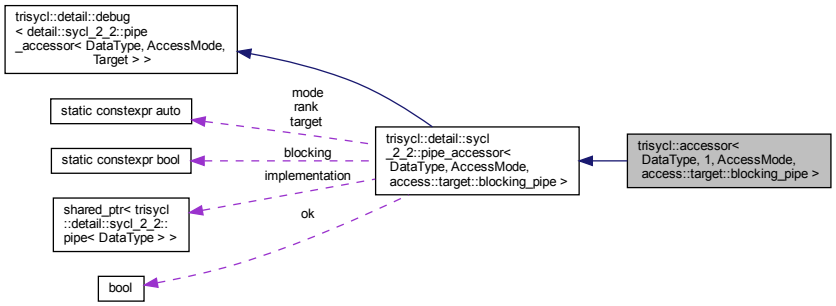
A specialization for a blocking pipe

Definition at line 445 of file accessor.hpp.

Inheritance diagram for trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >:



Collaboration diagram for trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >:



Public Types

- using accessor_detail = detail::sycl_2_2::pipe_accessor< DataType, AccessMode, access::target::blocking_pipe >

Public Member Functions

- accessor (sycl_2_2::pipe< DataType > &p, handler &command_group_handler)

  Construct a pipe accessor from a pipe using a command group handler object from the command group scope.
- accessor (sycl_2_2::pipe< DataType > &p)

  Construct a pipe accessor from a pipe outside of a normal kernel, for example in host code.
- sycl_2_2::pipe_reservation< accessor > reserve (std::size_t size) const

  Make a reservation inside the pipe.
- auto & get_pipe_detail ()

  Get the underlying pipe implementation.

Additional Inherited Members

9.9.2.2.1 Member Typedef Documentation

9.9.2.2.1.1 accessor_detail

```
template<typename DataType , access::mode AccessMode>
using trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >::accessor_↩
detail = detail::sycl_2_2::pipe_accessor<DataType, AccessMode, access::target::blocking_pipe>
```

Definition at line 451 of file accessor.hpp.

9.9.2.2.2 Constructor & Destructor Documentation

9.9.2.2.2.1 accessor() [1/2]

```
template<typename DataType , access::mode AccessMode>
trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >::accessor (
            sycl_2_2::pipe< DataType > & p,
            handler & command_group_handler )  [inline]
```

Construct a pipe accessor from a pipe using a command group handler object from the command group scope.

access_target defines the form of access being obtained.

Definition at line 463 of file accessor.hpp.

```
00464     : accessor_detail { p.implementation, command_group_handler } { }
```

9.9.2.2.2.2 accessor() [2/2]

```
template<typename DataType , access::mode AccessMode>
trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >::accessor (
            sycl_2_2::pipe< DataType > & p )  [inline]
```

Construct a pipe accessor from a pipe outside of a normal kernel, for example in host code.

Definition at line 470 of file accessor.hpp.

```
00471     : accessor_detail { p.implementation } { }
```

9.9.2.2.3 Member Function Documentation

9.9.2.2.3.1 get_pipe_detail() template<typename DataType , access::mode AccessMode> auto & trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >::get_pipe_detail ( ) [inline]

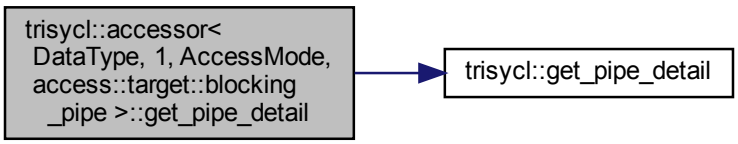
Get the underlying pipe implementation.

Definition at line 481 of file accessor.hpp.

```
00481                          {
00482     return accessor_detail::get_pipe_detail();
00483   }
```

References trisycl::get_pipe_detail().

Here is the call graph for this function:



9.9.2.2.3.2 reserve() template<typename DataType , access::mode AccessMode> sycl_2_2::pipe_reservation< accessor > trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_p >::reserve (std::size_t *size*) const [inline]

Make a reservation inside the pipe.

Definition at line 475 of file accessor.hpp.

```
00475                                                          {
00476     return accessor_detail::reserve(size);
00477   }
```

9.9.2.3 class trisycl::host_accessor

template<typename DataType, int Dimensions, access::mode AccessMode = access::mode::read_write>
class trisycl::host_accessor< DataType, Dimensions, AccessMode >

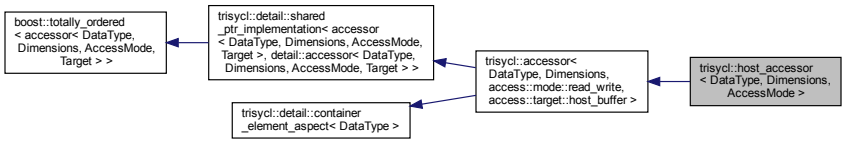
SYCL 2020 host_accessor.

Parameters

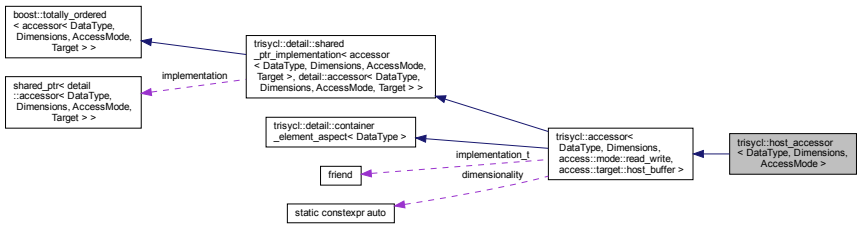
in	*DataType*	is the value type of the elements
in	*Dimensions*	is the rank of the accessor
in	*AccessMode*	precises whether the access is done in read or write mode. By default this is read+write

Definition at line 499 of file accessor.hpp.

Inheritance diagram for trisycl::host_accessor< DataType, Dimensions, AccessMode >:



Collaboration diagram for trisycl::host_accessor< DataType, Dimensions, AccessMode >:



Public Member Functions

- template<typename Allocator >
  host_accessor (buffer< DataType, Dimensions, Allocator > &target_buffer)

  Create an accessor to a buffer to access from the host.
- template<typename Allocator >
  accessor (buffer< DataType, Dimensions, Allocator > &target_buffer, handler &command_group_handler)

  Construct a buffer accessor from a buffer using a command group handler object from the command group scope.
- template<typename Allocator >
  accessor (buffer< DataType, Dimensions, Allocator > &target_buffer)

  Construct a buffer accessor from a buffer.
- template<typename Allocator >
  accessor (buffer< DataType, Dimensions, Allocator > &target_buffer, handler &command_group_handler, const range< Dimensions > &offset, const range< Dimensions > &range)

  Construct a buffer accessor from a buffer given a specific range for access permissions and an offset that provides the starting point for the access range using a command group handler object from the command group scope.
- accessor (const range< Dimensions > &allocation_size, handler &command_group_handler)

  Construct an accessor of dimension Dimensions with elements of type DataType using the passed range to specify the size in each dimension.

Private Types

- using base = accessor< DataType, Dimensions, AccessMode, access::target::host_buffer >

Additional Inherited Members

9.9.2.3.1 Member Typedef Documentation

9.9.2.3.1.1 base

```
template<typename DataType , int Dimensions, access::mode AccessMode = access::mode::read_write>
using trisycl::host_accessor< DataType, Dimensions, AccessMode >::base = accessor<DataType, Dimensions, AccessMode, access::target::host_buffer>  [private]
```

Definition at line 503 of file accessor.hpp.

9.9.2.3.2 Constructor & Destructor Documentation

9.9.2.3.2.1 host_accessor()

```
template<typename DataType , int Dimensions, access::mode AccessMode = access::mode::read_write>
template<typename Allocator >
trisycl::host_accessor< DataType, Dimensions, AccessMode >::host_accessor (
            buffer< DataType, Dimensions, Allocator > & target_buffer )  [inline]
```

Create an accessor to a buffer to access from the host.

Definition at line 512 of file accessor.hpp.

```
00513     : base { target_buffer } {}
```

9.9.2.3.3 Member Function Documentation

9.9.2.3.3.1 accessor() [1/4]

```
template<typename DataType , int Dimensions, access::mode AccessMode = access::mode::read_write>
template<typename Allocator >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            buffer< DataType, Dimensions, Allocator > & target_buffer )  [inline]
```

Construct a buffer accessor from a buffer.

Constructor only available for host_buffer target.

access_target defines the form of access being obtained.

Definition at line 125 of file accessor.hpp.

```
00126     : implementation_t {
00127     new detail::accessor<DataType, Dimensions, AccessMode, Target> {
00128       target_buffer.implementation->implementation }
00129   } {
00130     static_assert(Target == access::target::host_buffer,
00131                   "without a handler, access target should be host_buffer");
00132   }
```

9.9.2.3.3.2 accessor() [2/4]

```
template<typename DataType , int Dimensions, access::mode AccessMode = access::mode::read_write>
template<typename Allocator >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            buffer< DataType, Dimensions, Allocator > & target_buffer,
            handler & command_group_handler )  [inline]
```

Construct a buffer accessor from a buffer using a command group handler object from the command group scope.

Constructor only available for global_buffer or constant_buffer target.

access_target defines the form of access being obtained.

Todo Add template allocator type in all the accessor constructors in the specification or just use a more opaque Buffer type?

Todo fix specification where access mode should be target instead

Definition at line 104 of file accessor.hpp.

```
00105                                       : implementation_t {
00106     new detail::accessor<DataType, Dimensions, AccessMode, Target> {
00107       target_buffer.implementation->implementation, command_group_handler }
00108   } {
00109     static_assert(Target == access::target::global_buffer
00110                   || Target == access::target::constant_buffer,
00111                   "access target should be global_buffer or constant_buffer "
00112                   "when a handler is used");
00113     // Now the implementation is created, register it
00114     implementation->register_accessor();
00115   }
```

9.9.2.3.3.3 accessor() [3/4]

```
template<typename DataType , int Dimensions, access::mode AccessMode = access::mode::read_write>
template<typename Allocator >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            buffer< DataType, Dimensions, Allocator > & target_buffer,
            handler & command_group_handler,
            const range< Dimensions > & offset,
            const range< Dimensions > & range )  [inline]
```

Construct a buffer accessor from a buffer given a specific range for access permissions and an offset that provides the starting point for the access range using a command group handler object from the command group scope.

This accessor limits the processing of the buffer to the [offset, offset+range[for every dimension. Any other parts of the buffer will be unaffected.

Constructor only available for access modes global_buffer, and constant_buffer (see Table "Buffer accessor constructors"). access_target defines the form of access being obtained.

This accessor is recommended for discard-write and discard read write access modes, when the unaffected parts of the processing should be retained.

Definition at line 153 of file accessor.hpp.

```
00156                                       {
00157     TRISYCL_UNIMPL;
00158   }
```

9.9.2.3.3.4 accessor() [4/4]

```
template<typename DataType , int Dimensions, access::mode Access↩
Mode = access::mode::read_write>
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            const range< Dimensions > & allocation_size,
            handler & command_group_handler )  [inline]
```

Construct an accessor of dimension Dimensions with elements of type DataType using the passed range to specify the size in each dimension.

It needs as a parameter a command group handler object from the command group scope. Constructor only available if AccessMode is local, see Table 3.25.

Definition at line 169 of file accessor.hpp.

```
00171     :  implementation_t { new detail::accessor<DataType,
00172                                                Dimensions,
00173                                                AccessMode,
00174                                                access::target::local> {
00175       allocation_size, command_group_handler
00176         }
00177   }
00178   {
00179     static_assert(Target == access::target::local,
00180                   "This accessor constructor requires "
00181                   "access target be local");
00182   }
```

9.9.2.4 class trisycl::sycl_2_2::pipe

template<typename T>
class trisycl::sycl_2_2::pipe< T >

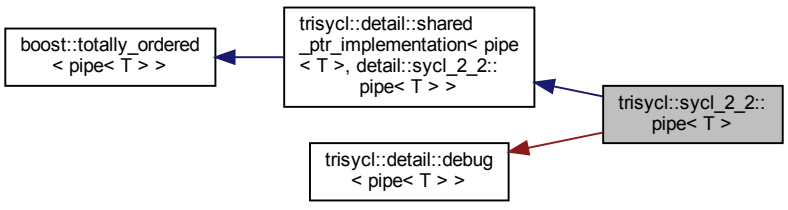
A SYCL pipe.

This is a proposal for the now abandoned SYCL 2.2 provisional specification. This is still here for historical reasons.

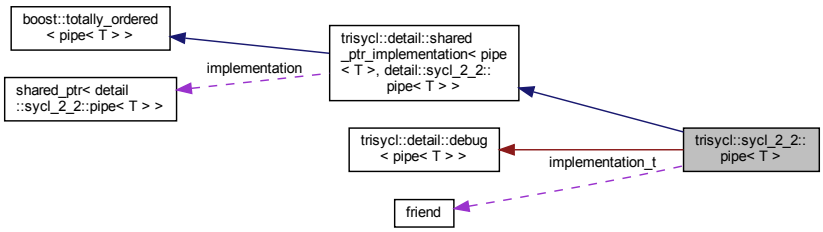
Implement a FIFO-style object that can be used through accessors to send some objects T from the input to the output

Definition at line 38 of file pipe.hpp.

Inheritance diagram for trisycl::sycl_2_2::pipe< T >:



Collaboration diagram for trisycl::sycl_2_2::pipe< T >:



Public Types

- using value_type = T

  The STL-like types.

Public Member Functions

- pipe (std::size_t capacity)

  Construct a pipe able to store up to capacity T objects.
- template<access::mode Mode, access::target Target = access::target::pipe>
  accessor< value_type, 1, Mode, Target > get_access (handler &command_group_handler)

  Get an accessor to the pipe with the required mode.
- std::size_t capacity () const

  Return the maximum number of elements that can fit in the pipe.

Private Types

- using implementation_t = typename pipe::shared_ptr_implementation

Private Attributes

- friend implementation_t

Additional Inherited Members

9.9.2.4.1 Member Typedef Documentation

9.9.2.4.1.1 implementation_t `template<typename T >`

```
using trisycl::sycl_2_2::pipe< T >::implementation_t = typename pipe::shared_ptr_implementation
[private]
```

Definition at line 46 of file pipe.hpp.

9.9.2.4.1.2 value_type `template<typename T >`

```
using trisycl::sycl_2_2::pipe< T >::value_type = T
```

The STL-like types.

Definition at line 59 of file pipe.hpp.

9.9.2.4.2 Constructor & Destructor Documentation

9.9.2.4.2.1 pipe() `template<typename T >`

```
trisycl::sycl_2_2::pipe< T >::pipe (
            std::size_t capacity )  [inline]
```

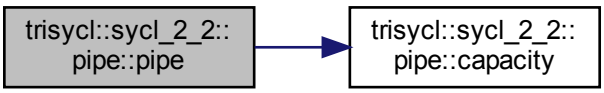
Construct a pipe able to store up to capacity T objects.

Definition at line 63 of file pipe.hpp.

```
00064     :  implementation_t { new detail::sycl_2_2::pipe<T> { capacity } } { }
```

References trisycl::sycl_2_2::pipe< T >::capacity().

Here is the call graph for this function:



9.9.2.4.3 Member Function Documentation

9.9.2.4.3.1 capacity() `template<typename T >`

```
std::size_t trisycl::sycl_2_2::pipe< T >::capacity ( ) const  [inline]
```

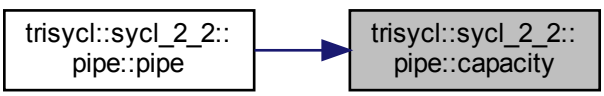
Return the maximum number of elements that can fit in the pipe.

Definition at line 89 of file pipe.hpp.

```
00089                                {
00090     return implementation->capacity();
00091     }
```

Referenced by trisycl::sycl_2_2::pipe< T >::pipe().

Here is the caller graph for this function:



9.9.2.4.3.2 get_access() template<typename T >

```
template<access::mode Mode, access::target Target = access::target::pipe>
accessor< value_type, 1, Mode, Target > trisycl::sycl_2_2::pipe< T >::get_access (
            handler & command_group_handler )  [inline]
```

Get an accessor to the pipe with the required mode.

Parameters

	Mode	is the requested access mode
	Target	is the type of pipe access required
in	*command_group_handler*	is the command group handler in which the kernel is to be executed

Definition at line 79 of file pipe.hpp.

```
00079                                          {
00080     static_assert(Target == access::target::pipe
00081                   || Target == access::target::blocking_pipe,
00082                   "get_access(handler) with pipes can only deal with "
00083                   "access::pipe or access::blocking_pipe");
00084     return { implementation, command_group_handler };
00085   }
```

References trisycl::access::blocking_pipe, and trisycl::access::pipe.

9.9.2.4.4 Member Data Documentation

9.9.2.4.4.1 implementation_t template<typename T >

```
friend trisycl::sycl_2_2::pipe< T >::implementation_t  [private]
```

Definition at line 49 of file pipe.hpp.

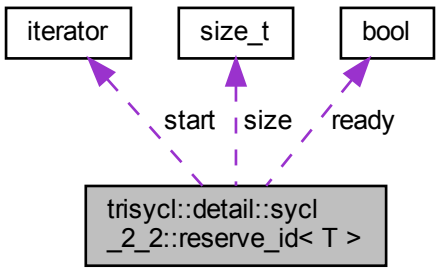
9.9.2.5 struct trisycl::detail::sycl_2_2::reserve_id

template<typename T>
struct trisycl::detail::sycl_2_2::reserve_id< T >

A private description of a reservation station.

Definition at line 41 of file pipe.hpp.

Collaboration diagram for trisycl::detail::sycl_2_2::reserve_id< T >:



Public Member Functions

- reserve_id (typename boost::circular_buffer< T >::iterator start, std::size_t size)

  Track a reservation not committed yet.

Public Attributes

- boost::circular_buffer< T >::iterator start

  Start of the reservation in the pipe storage.
- std::size_t size

  Number of elements in the reservation.
- bool ready = false

9.9.2.5.1 Constructor & Destructor Documentation

9.9.2.5.1.1 reserve_id() `template<typename T >`

```
trisycl::detail::sycl_2_2::reserve_id< T >::reserve_id (
            typename boost::circular_buffer< T >::iterator start,
            std::size_t size ) [inline]
```

Track a reservation not committed yet.

Parameters

in	*start*	point to the start of the reservation in the pipe storage
in	*size*	is the number of elements in the reservation

Definition at line 59 of file pipe.hpp.

```
00060                               : start { start }, size { size } {}
```

9.9.2.5.2 Member Data Documentation

9.9.2.5.2.1 ready `template<typename T >`

```
bool trisycl::detail::sycl_2_2::reserve_id< T >::ready = false
```

Definition at line 50 of file pipe.hpp.

9.9.2.5.2.2 size `template<typename T >`

```
std::size_t trisycl::detail::sycl_2_2::reserve_id< T >::size
```

Number of elements in the reservation.

Definition at line 46 of file pipe.hpp.

9.9.2.5.2.3 start `template<typename T >`

`boost::circular_buffer<T>::iterator trisycl::detail::sycl_2_2::reserve_id< T >::start`

Start of the reservation in the pipe storage.

Definition at line 43 of file pipe.hpp.

9.9.2.6 class trisycl::detail::sycl_2_2::pipe

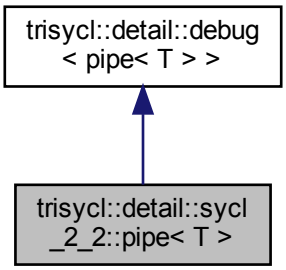
template<typename T>
class trisycl::detail::sycl_2_2::pipe< T >

Implement a pipe object.

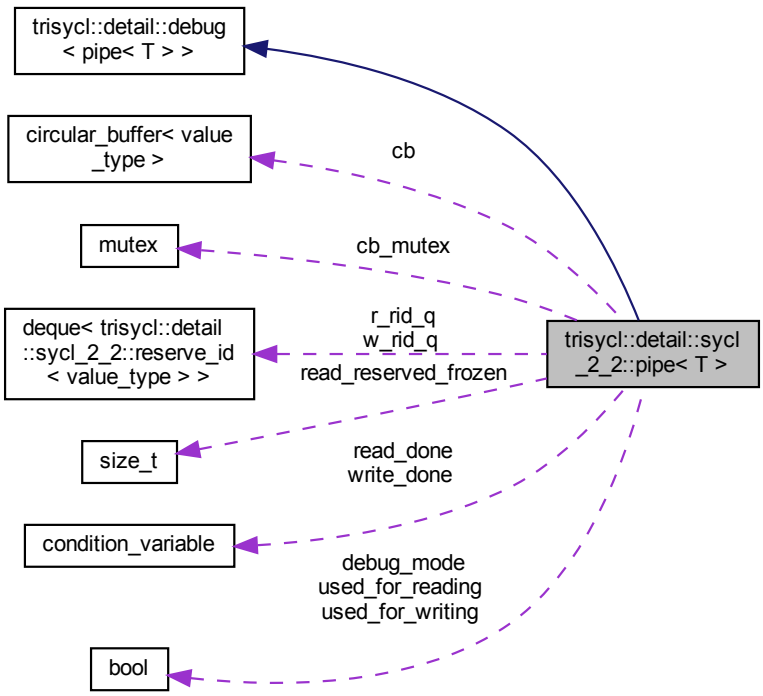
Use some mutable members so that the pipe object can be changed even when the accessors are captured in a lambda.

Definition at line 71 of file pipe.hpp.

Inheritance diagram for trisycl::detail::sycl_2_2::pipe< T >:



Collaboration diagram for trisycl::detail::sycl_2_2::pipe< T >:



Public Types

- using value_type = T
- using implementation_t = boost::circular_buffer< value_type >

  Implement the pipe with a circular buffer.
- using rid_iterator = typename decltype(w_rid_q)::iterator

Public Member Functions

- pipe (std::size_t capacity)

  Create a pipe as a circular buffer of the required capacity.
- std::size_t capacity () const

  Return the maximum number of elements that can fit in the pipe.
- std::size_t size_with_lock () const

  The size() method used outside needs to lock the datastructure.
- bool empty_with_lock () const

  The empty() method used outside needs to lock the datastructure.
- bool full_with_lock () const
- bool write (const T &value, bool blocking=false)

  Try to write a value to the pipe.
- bool read (T &value, bool blocking=false)

  Try to read a value from the pipe.
- std::size_t reserved_for_reading () const

Compute the amount of elements blocked by read reservations, not yet committed.

- std::size_t reserved_for_writing () const

  Compute the amount of elements blocked by write reservations, not yet committed.
- bool reserve_read (std::size_t s, rid_iterator &rid, bool blocking=false)

  Reserve some part of the pipe for reading.
- bool reserve_write (std::size_t s, rid_iterator &rid, bool blocking=false)

  Reserve some part of the pipe for writing.
- void move_read_reservation_forward ()

  Process the read reservations that are ready to be released in the reservation queue.
- void move_write_reservation_forward ()

  Process the write reservations that are ready to be released in the reservation queue.

Public Attributes

- bool used_for_reading = false

  True when the pipe is currently used for reading.
- bool used_for_writing = false

  True when the pipe is currently used for writing.

Private Member Functions

- std::size_t size () const

  Get the current number of elements in the pipe that can be read.
- bool empty () const

  Test if the pipe is empty.
- bool full () const

  Test if the pipe is full.

Private Attributes

- boost::circular_buffer< value_type > cb

  The circular buffer to store the elements.
- std::mutex cb_mutex

  To protect the access to the circular buffer.
- std::deque< reserve_id< value_type > > w_rid_q

  The queue of pending write reservations.
- std::deque< reserve_id< value_type > > r_rid_q

  The queue of pending read reservations.
- std::size_t read_reserved_frozen

  Track the number of frozen elements related to read reservations.
- std::condition_variable read_done

  To signal that a read has been successful.
- std::condition_variable write_done

  To signal that a write has been successful.
- bool debug_mode = false

  To control the debug mode, disabled by default.

Additional Inherited Members

9.9.2.6.1 Member Typedef Documentation

9.9.2.6.1.1 implementation_t `template<typename T >`

```
using trisycl::detail::sycl_2_2::pipe< T >::implementation_t = boost::circular_buffer<value_type>
```

Implement the pipe with a circular buffer.

Definition at line 78 of file pipe.hpp.

9.9.2.6.1.2 rid_iterator `template<typename T >`

```
using trisycl::detail::sycl_2_2::pipe< T >::rid_iterator = typename decltype(w_rid_q)::iterator
```

Definition at line 96 of file pipe.hpp.

9.9.2.6.1.3 value_type `template<typename T >`

```
using trisycl::detail::sycl_2_2::pipe< T >::value_type = T
```

Definition at line 75 of file pipe.hpp.

9.9.2.6.2 Constructor & Destructor Documentation

9.9.2.6.2.1 pipe() `template<typename T >`

```
trisycl::detail::sycl_2_2::pipe< T >::pipe (
            std::size_t capacity )  [inline]
```

Create a pipe as a circular buffer of the required capacity.

Definition at line 127 of file pipe.hpp.

```
00127 :  cb { capacity }, read_reserved_frozen { 0 } { }
```

9.9.2.6.3 Member Function Documentation

9.9.2.6.3.1 capacity() template<typename T >

```
std::size_t trisycl::detail::sycl_2_2::pipe< T >::capacity ( ) const  [inline]
```

Return the maximum number of elements that can fit in the pipe.

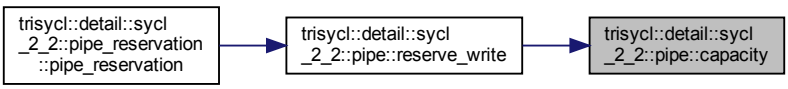
Definition at line 132 of file pipe.hpp.

```
00132                                {
00133     // No lock required since it is fixed and set at construction time
00134     return cb.capacity();
00135   }
```

References trisycl::detail::sycl_2_2::pipe< T >::cb.

Referenced by trisycl::detail::sycl_2_2::pipe< T >::reserve_write().

Here is the caller graph for this function:



9.9.2.6.3.2 empty() template<typename T >

```
bool trisycl::detail::sycl_2_2::pipe< T >::empty ( ) const  [inline], [private]
```

Test if the pipe is empty.

This is obviously a volatile value which is constrained by restricted relativity.

Note that on some devices it may be costly to implement on the write side (for example on FPGA).

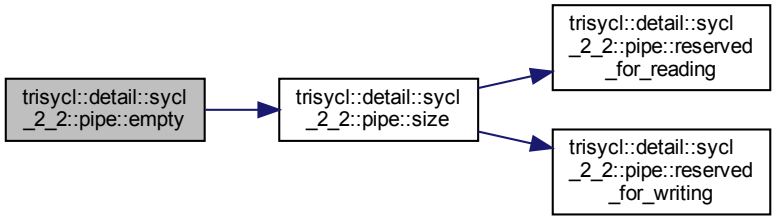
Definition at line 167 of file pipe.hpp.

```
00167                         {
00168     TRISYCL_DUMP_T("empty() cb.size() = " << cb.size()
00169                    << " size() = " << size());
00170     // It is empty when the size is zero, taking into account reservations
00171     return size() ==  0;
00172   }
```

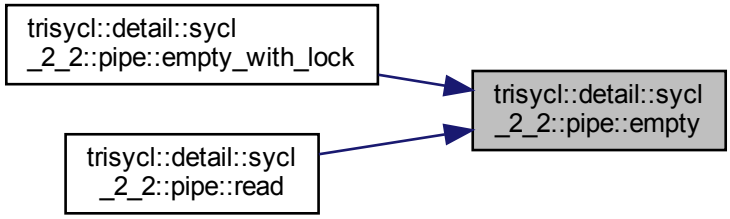
References trisycl::detail::sycl_2_2::pipe< T >::cb, trisycl::detail::sycl_2_2::pipe< T >::size(), and TRISYCL_DUMP_T.

Referenced by trisycl::detail::sycl_2_2::pipe< T >::empty_with_lock(), and trisycl::detail::sycl_2_2::pipe< T >::read().

Here is the call graph for this function:



Here is the caller graph for this function:



9.9.2.6.3.3 empty_with_lock() template<typename T >

bool trisycl::detail::sycl_2_2::pipe< T >::empty_with_lock ( ) const [inline]

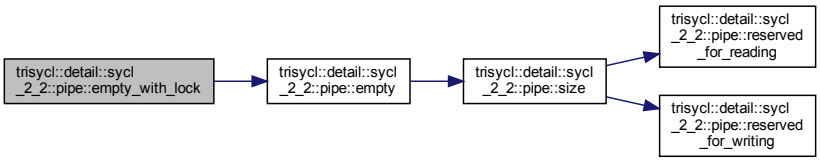
The empty() method used outside needs to lock the datastructure.

Definition at line 198 of file pipe.hpp.

```
00198                                {
00199     std::lock_guard<std::mutex> lg { cb_mutex };
00200     return empty();
00201   }
```

References trisycl::detail::sycl_2_2::pipe< T >::cb_mutex, and trisycl::detail::sycl_2_2::pipe< T >::empty().

Here is the call graph for this function:



9.9.2.6.3.4 full() template<typename T >

bool trisycl::detail::sycl_2_2::pipe< T >::full ( ) const [inline], [private]

Test if the pipe is full.

This is obviously a volatile value which is constrained by restricted relativity.

Note that on some devices it may be costly to implement on the read side (for example on FPGA).

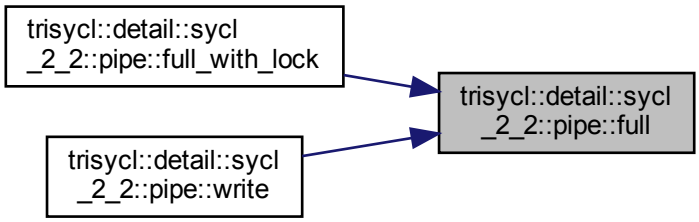
Definition at line 183 of file pipe.hpp.

```
00183                     {
00184     return cb.full();
00185   }
```

References trisycl::detail::sycl_2_2::pipe< T >::cb.

Referenced by trisycl::detail::sycl_2_2::pipe< T >::full_with_lock(), and trisycl::detail::sycl_2_2::pipe< T >::write().

Here is the caller graph for this function:



9.9.2.6.3.5 full_with_lock() template<typename T >

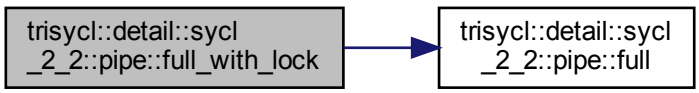
```
bool trisycl::detail::sycl_2_2::pipe< T >::full_with_lock ( ) const  [inline]
```

Definition at line 205 of file pipe.hpp.

```
00205                               {
00206     std::lock_guard<std::mutex> lg { cb_mutex };
00207     return full();
00208   }
```

References trisycl::detail::sycl_2_2::pipe< T >::cb_mutex, and trisycl::detail::sycl_2_2::pipe< T >::full().

Here is the call graph for this function:



9.9.2.6.3.6 move_read_reservation_forward() template<typename T >

void trisycl::detail::sycl_2_2::pipe< T >::move_read_reservation_forward ( ) [inline]

Process the read reservations that are ready to be released in the reservation queue.

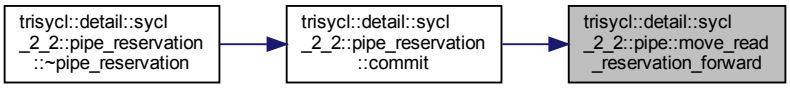
Definition at line 432 of file pipe.hpp.

```
00432                                       {
00433     // Lock the pipe to avoid nuisance
00434     std::unique_lock<std::mutex> lock { cb_mutex };
00435
00436     for (;;) {
00437       if (r_rid_q.empty())
00438         // No pending reservation, so nothing to do
00439         break;
00440       if (!r_rid_q.front().ready)
00441         /* If the first reservation is not ready to be released, stop
00442 because it is blocking all the following in the queue
00443 anyway */
00444         break;
00445       // Remove the reservation to be released from the queue
00446       r_rid_q.pop_front();
00447       std::size_t n_to_pop;
00448       if (r_rid_q.empty())
00449         // If it was the last one, remove all the reservation
00450         n_to_pop = read_reserved_frozen;
00451       else
00452         // Else remove everything up to the next reservation
00453         n_to_pop =  r_rid_q.front().start - cb.begin();
00454       // No longer take into account these reserved slots
00455       read_reserved_frozen -= n_to_pop;
00456       // Release the elements from the FIFO
00457       while (n_to_pop--)
00458         cb.pop_front();
00459       // Notify the clients waiting for some room to write in the pipe
00460       read_done.notify_all();
00461       /* ...and process the next reservation to see if it is ready to
00462 be released too */
00463     }
00464   }
```

References trisycl::detail::sycl_2_2::pipe< T >::cb, trisycl::detail::sycl_2_2::pipe< T >::cb_mutex, trisycl::detail::sycl_2_2::pipe< T >
trisycl::detail::sycl_2_2::pipe< T >::read_done, and trisycl::detail::sycl_2_2::pipe< T >::read_reserved_frozen.

Referenced by trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::commit().

Here is the caller graph for this function:



9.9.2.6.3.7 move_write_reservation_forward() template<typename T >

void trisycl::detail::sycl_2_2::pipe< T >::move_write_reservation_forward ( ) [inline]

Process the write reservations that are ready to be released in the reservation queue.

Definition at line 470 of file pipe.hpp.

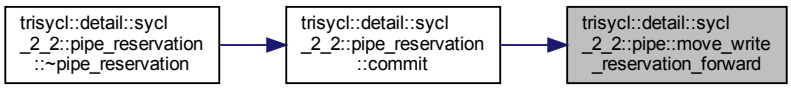
```
00470                                       {
00471     // Lock the pipe to avoid nuisance
00472     std::lock_guard<std::mutex> lg { cb_mutex };
00473
00474     for (;;) {
```

```
00475       if (w_rid_q.empty())
00476         // No pending reservation, so nothing to do
00477         break;
00478       // Get the first reservation
00479       const auto &rid = w_rid_q.front();
00480       if (!rid.ready)
00481         /* If the reservation is not ready to be released, stop
00482 because it is blocking all the following in the queue
00483 anyway */
00484         break;
00485       // Remove the reservation to be released from the queue
00486       w_rid_q.pop_front();
00487       // Notify the clients waiting to read something from the pipe
00488       write_done.notify_all();
00489       /* ...and process the next reservation to see if it is ready to
00490 be released too */
00491     }
00492   }
```

References trisycl::detail::sycl_2_2::pipe< T >::cb_mutex, trisycl::detail::sycl_2_2::pipe< T >::w_rid_q, and trisycl::detail::sycl_2_2::pipe< T >::write_done.

Referenced by trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::commit().

Here is the caller graph for this function:



9.9.2.6.3.8 read() template<typename T >

```
bool trisycl::detail::sycl_2_2::pipe< T >::read (
              T & value,
              bool blocking = false )  [inline]
```

Try to read a value from the pipe.

Parameters

out	*value*	is the reference to where to store what is read
in	*blocking*	specify if the call wait for the operation to succeed

Returns

true on success

If there is a pending reservation, read the next element to be read and update the number of reserved elements

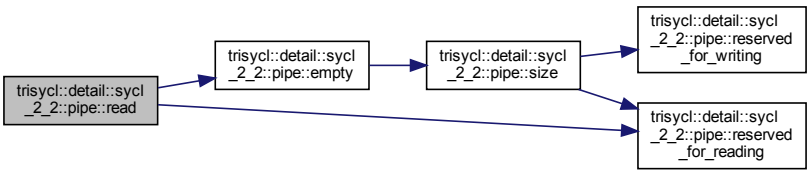
Definition at line 262 of file pipe.hpp.

```
00262                                          {
00263     // Lock the pipe to avoid being disturbed
00264     std::unique_lock<std::mutex> ul { cb_mutex };
00265     TRISYCL_DUMP_T("Read pipe empty = " « empty());
00266
```

```
00267     if (blocking)
00268       /* If in blocking mode, wait for the not empty condition, that
00269 may be changed when a write is done */
00270       write_done.wait(ul, [&] { return !empty(); });
00271     else if (empty())
00272       return false;
00273
00274     TRISYCL_DUMP_T("Read pipe front = " « cb.front()
00275                    « " back = " « cb.back()
00276                    « " reserved_for_reading() = " « reserved_for_reading());
00277     if (read_reserved_frozen)
00278 /** If there is a pending reservation, read the next element to
00279 be read and update the number of reserved elements */
00280       value = cb.begin()[read_reserved_frozen++];
00281     else {
00282       /* There is no pending read reservation, so pop the read value
00283 from the pipe */
00284       value = cb.front();
00285       cb.pop_front();
00286     }
00287
00288     TRISYCL_DUMP_T("Read pipe value = " « value);
00289     // Micro-optimization:  unlock before the notification
00290     // https://en.cppreference.com/w/cpp/thread/condition_variable/notify_all
00291     ul.unlock();
00292     // Notify the clients waiting for some room to write in the pipe
00293     read_done.notify_all();
00294     return true;
00295   }
```

References trisycl::detail::sycl_2_2::pipe< T >::cb, trisycl::detail::sycl_2_2::pipe< T >::cb_mutex, trisycl::detail::sycl_2_2::pipe< T >
trisycl::detail::sycl_2_2::pipe< T >::read_done, trisycl::detail::sycl_2_2::pipe< T >::read_reserved_frozen,
trisycl::detail::sycl_2_2::pipe< T >::reserved_for_reading(), TRISYCL_DUMP_T, and trisycl::detail::sycl_2_2::pipe< T >::write_done

Here is the call graph for this function:



9.9.2.6.3.9 reserve_read() template<typename T >

```
bool trisycl::detail::sycl_2_2::pipe< T >::reserve_read (
            std::size_t s,
            rid_iterator & rid,
            bool blocking = false )  [inline]
```

Reserve some part of the pipe for reading.

Parameters

in	*s*	is the number of element to reserve
out	*rid*	is an iterator to a description of the reservation that has been done if successful
in	*blocking*	specify if the call wait for the operation to succeed

Returns

true if the reservation was successful

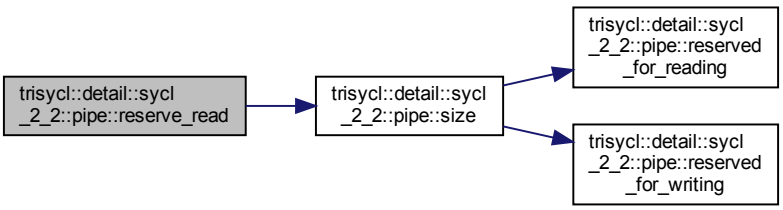
Definition at line 342 of file pipe.hpp.

```
00344                                           {
00345     // Lock the pipe to avoid being disturbed
00346     std::unique_lock<std::mutex> ul { cb_mutex };
00347
00348     TRISYCL_DUMP_T("Before read reservation cb.size() = " « cb.size()
00349                    « " size() = " « size());
00350     if (s == 0)
00351       // Empty reservation requested, so nothing to do
00352       return false;
00353
00354     if (blocking)
00355       /* If in blocking mode, wait for enough elements to read in the
00356 pipe for the reservation.  This condition can change when a
00357 write is done */
00358       write_done.wait(ul, [&] { return s <= size(); });
00359     else if (s > size())
00360       // Not enough elements to read in the pipe for the reservation
00361       return false;
00362
00363     // Compute the location of the first element of the reservation
00364     auto first = cb.begin() + read_reserved_frozen;
00365     // Increment the number of frozen elements
00366     read_reserved_frozen += s;
00367     /* Add a description of the reservation at the end of the
00368 reservation queue */
00369     r_rid_q.emplace_back(first, s);
00370     // Return the iterator to the last reservation descriptor
00371     rid = r_rid_q.end() - 1;
00372     TRISYCL_DUMP_T("After reservation cb.size() = " « cb.size()
00373                    « " size() = " « size());
00374     return true;
00375   }
```

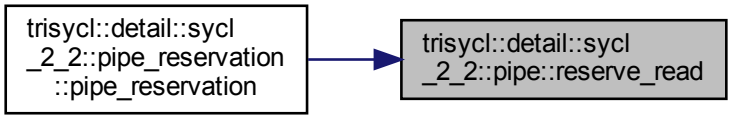
References trisycl::detail::sycl_2_2::pipe< T >::cb, trisycl::detail::sycl_2_2::pipe< T >::cb_mutex, trisycl::detail::sycl_2_2::pipe< T >
trisycl::detail::sycl_2_2::pipe< T >::read_reserved_frozen, trisycl::detail::sycl_2_2::pipe< T >::size(), TRISYCL_DUMP_T,
and trisycl::detail::sycl_2_2::pipe< T >::write_done.

Referenced by trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::pipe_reservation().

Here is the call graph for this function:



Here is the caller graph for this function:



9.9.2.6.3.10 reserve_write() template<typename T >

```
bool trisycl::detail::sycl_2_2::pipe< T >::reserve_write (
            std::size_t s,
            rid_iterator & rid,
            bool blocking = false )  [inline]
```

Reserve some part of the pipe for writing.

Parameters

in	*s*	is the number of element to reserve
out	*rid*	is an iterator to a description of the reservation that has been done if successful
in	*blocking*	specify if the call wait for the operation to succeed

Returns

true if the reservation was successful

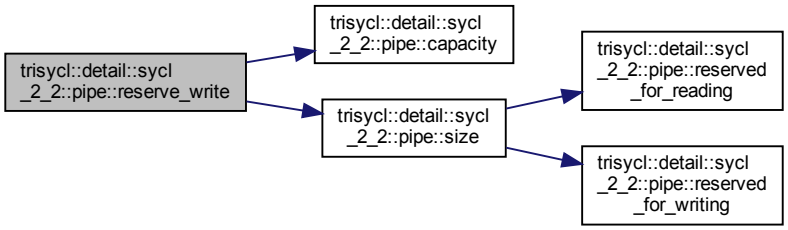
Definition at line 390 of file pipe.hpp.

```
00392                                                {
00393     // Lock the pipe to avoid being disturbed
00394     std::unique_lock<std::mutex> ul { cb_mutex };
00395
00396     TRISYCL_DUMP_T("Before write reservation cb.size() = " « cb.size()
00397                    « " size() = " « size());
00398     if (s == 0)
00399       // Empty reservation requested, so nothing to do
00400       return false;
00401
00402     if (blocking)
00403       /* If in blocking mode, wait for enough room in the pipe, that
00404 may be changed when a read is done.  Do not use a difference
00405 here because it is only about unsigned values */
00406       read_done.wait(ul, [&] { return cb.size() + s <= capacity(); });
00407     else if (cb.size() + s > capacity())
00408       // Not enough room in the pipe for the reservation
00409       return false;
00410
00411     /* If there is enough room in the pipe, just create default values
00412 in it to do the reservation */
00413     for (std::size_t i = 0; i != s; ++i)
00414       cb.push_back();
00415     /* Compute the location of the first element a posteriori since it
00416 may not exist a priori if cb was empty before */
00417     auto first = cb.end() - s;
00418     /* Add a description of the reservation at the end of the
00419 reservation queue */
00420     w_rid_q.emplace_back(first, s);
00421     // Return the iterator to the last reservation descriptor
00422     rid = w_rid_q.end() - 1;
00423     TRISYCL_DUMP_T("After reservation cb.size() = " « cb.size()
00424                    « " size() = " « size());
00425     return true;
00426   }
```

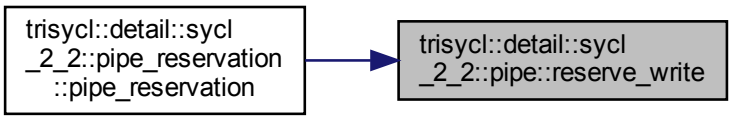
References trisycl::detail::sycl_2_2::pipe< T >::capacity(), trisycl::detail::sycl_2_2::pipe< T >::cb, trisycl::detail::sycl_2_2::pipe< T >::i, trisycl::detail::sycl_2_2::pipe< T >::read_done, trisycl::detail::sycl_2_2::pipe< T >::size(), TRISYCL_DUMP_T, and trisycl::detail::sycl_2_2::pipe< T >::w_rid_q.

Referenced by trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::pipe_reservation().

Here is the call graph for this function:



Here is the caller graph for this function:



9.9.2.6.3.11 reserved_for_reading() `template<typename T >`

`std::size_t trisycl::detail::sycl_2_2::pipe< T >::reserved_for_reading ( ) const [inline]`

Compute the amount of elements blocked by read reservations, not yet committed.

This includes some normal reads to pipes between/after un-committed reservations

This function assumes that the data structure is locked

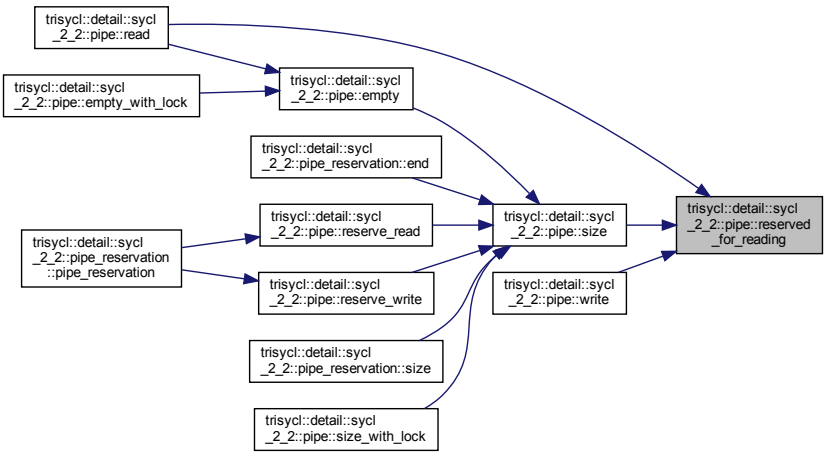
Definition at line 306 of file pipe.hpp.

```
00306                                       {
00307     return read_reserved_frozen;
00308   }
```

References trisycl::detail::sycl_2_2::pipe< T >::read_reserved_frozen.

Referenced by trisycl::detail::sycl_2_2::pipe< T >::read(), trisycl::detail::sycl_2_2::pipe< T >::size(), and trisycl::detail::sycl_2_2::pipe< T >::write().

Here is the caller graph for this function:



9.9.2.6.3.12 reserved_for_writing() `template<typename T >`

`std::size_t trisycl::detail::sycl_2_2::pipe< T >::reserved_for_writing ( ) const [inline]`

Compute the amount of elements blocked by write reservations, not yet committed.

This includes some normal writes to pipes between/after un-committed reservations

This function assumes that the data structure is locked

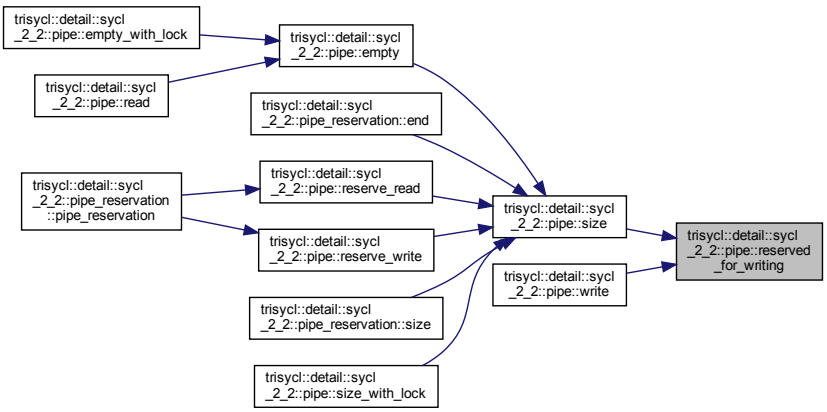
Definition at line 319 of file pipe.hpp.

```
00319                                          {
00320     if (w_rid_q.empty())
00321       // No on-going reservation
00322       return 0;
00323     else
00324       /* The reserved size is from the first element of the first
00325 on-going reservation up to the end of the pipe content */
00326       return cb.end() - w_rid_q.front().start;
00327   }
```

References trisycl::detail::sycl_2_2::pipe< T >::cb, and trisycl::detail::sycl_2_2::pipe< T >::w_rid_q.

Referenced by trisycl::detail::sycl_2_2::pipe< T >::size(), and trisycl::detail::sycl_2_2::pipe< T >::write().

Here is the caller graph for this function:



9.9.2.6.3.13 size() template<typename T >

```
std::size_t trisycl::detail::sycl_2_2::pipe< T >::size ( ) const  [inline], [private]
```

Get the current number of elements in the pipe that can be read.

This is obviously a volatile value which is constrained by the theory of restricted relativity.

Note that on some devices it may be costly to implement (for example on FPGA).

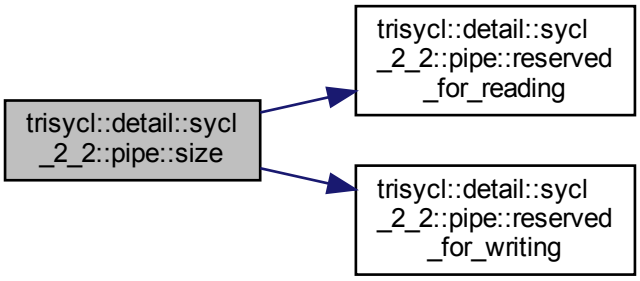
Definition at line 147 of file pipe.hpp.

```
00147                          {
00148     TRISYCL_DUMP_T("size() cb.size() = " « cb.size()
00149                    « " cb.end() = " « (void *)&*cb.end()
00150                    « " reserved_for_reading() = " « reserved_for_reading()
00151                    « " reserved_for_writing() = " « reserved_for_writing());
00152     /* The actual number of available elements depends from the
00153 elements blocked by some reservations.
00154 This prevents a consumer to read into reserved area.  */
00155     return cb.size() - reserved_for_reading() - reserved_for_writing();
00156   }
```

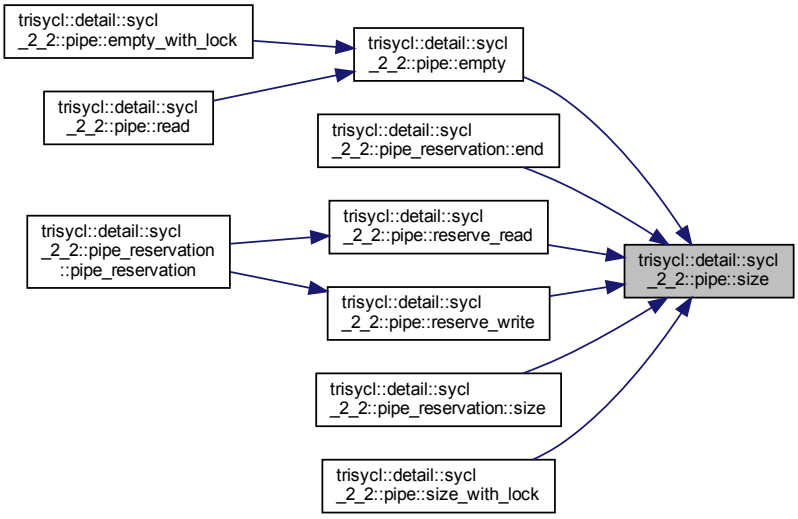
References trisycl::detail::sycl_2_2::pipe< T >::cb, trisycl::detail::sycl_2_2::pipe< T >::reserved_for_reading(), trisycl::detail::sycl_2_2::pipe< T >::reserved_for_writing(), and TRISYCL_DUMP_T.

Referenced by trisycl::detail::sycl_2_2::pipe< T >::empty(), trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::end(), trisycl::detail::sycl_2_2::pipe< T >::reserve_read(), trisycl::detail::sycl_2_2::pipe< T >::reserve_write(), trisycl::detail::sycl_2_2::pipe and trisycl::detail::sycl_2_2::pipe< T >::size_with_lock().

Here is the call graph for this function:



Here is the caller graph for this function:



9.9.2.6.3.14 size_with_lock() `template<typename T >`

`std::size_t trisycl::detail::sycl_2_2::pipe< T >::size_with_lock ( ) const [inline]`

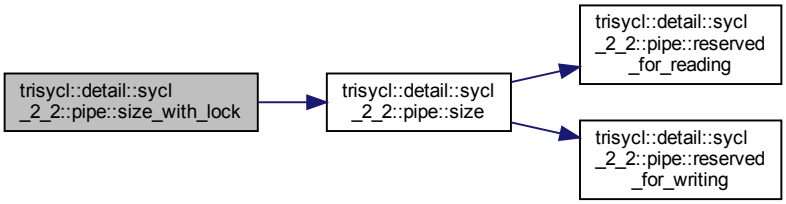
The size() method used outside needs to lock the datastructure.

Definition at line 191 of file pipe.hpp.

```
00191                                  {
00192     std::lock_guard<std::mutex> lg { cb_mutex };
00193     return size();
00194   }
```

References trisycl::detail::sycl_2_2::pipe< T >::cb_mutex, and trisycl::detail::sycl_2_2::pipe< T >::size().

Here is the call graph for this function:



9.9.2.6.3.15 write() template<typename T >

```
bool trisycl::detail::sycl_2_2::pipe< T >::write (
            const T & value,
            bool blocking = false )  [inline]
```

Try to write a value to the pipe.

Parameters

in	*value*	is what we want to write
in	*blocking*	specify if the call wait for the operation to succeed

Returns

true on success

Todo provide a && version

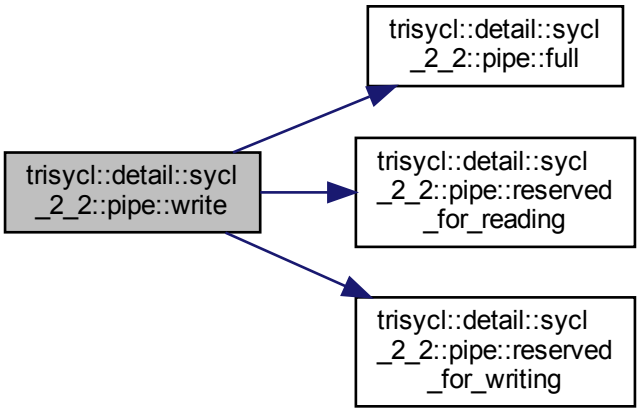
Definition at line 222 of file pipe.hpp.

```
00222                                                {
00223     // Lock the pipe to avoid being disturbed
00224     std::unique_lock<std::mutex> ul { cb_mutex };
00225     TRISYCL_DUMP_T("Write pipe full = " « full()
00226                    « " value = " « value);
00227
00228     if (blocking)
00229       /* If in blocking mode, wait for the not full condition, that
00230 may be changed when a read is done */
00231       read_done.wait(ul, [&] { return !full(); });
00232     else if (full())
00233       return false;
00234
00235     cb.push_back(value);
00236     TRISYCL_DUMP_T("Write pipe front = " « cb.front()
00237                    « " back = " « cb.back()
00238                    « " cb.begin() = " « (void *)&*cb.begin()
00239                    « " cb.size() = " « cb.size()
00240                    « " cb.end() = " « (void *)&*cb.end()
00241                    « " reserved_for_reading() = " « reserved_for_reading()
00242                    « " reserved_for_writing() = " « reserved_for_writing());
00243     // Micro-optimization:  unlock before the notification
00244     // https://en.cppreference.com/w/cpp/thread/condition_variable/notify_all
```

```
00245     ul.unlock();
00246     // Notify the clients waiting to read something from the pipe
00247     write_done.notify_all();
00248     return true;
00249   }
```

References trisycl::detail::sycl_2_2::pipe< T >::cb, trisycl::detail::sycl_2_2::pipe< T >::cb_mutex, trisycl::detail::sycl_2_2::pipe< T >::read_done, trisycl::detail::sycl_2_2::pipe< T >::reserved_for_reading(), trisycl::detail::sycl_2_2::pipe< T >::reserved_for_writing(), TRISYCL_DUMP_T, and trisycl::detail::sycl_2_2::pipe< T >::write_done.

Here is the call graph for this function:



9.9.2.6.4 Member Data Documentation

9.9.2.6.4.1 cb `template<typename T >`

`boost::circular_buffer<value_type> trisycl::detail::sycl_2_2::pipe< T >::cb [private]`

The circular buffer to store the elements.

Definition at line 83 of file pipe.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe< T >::capacity(), trisycl::detail::sycl_2_2::pipe< T >::empty(), trisycl::detail::sycl_2_2::pipe< T >::full(), trisycl::detail::sycl_2_2::pipe< T >::move_read_reservation_forward(), trisycl::detail::sycl_2_2::pipe< T >::read(), trisycl::detail::sycl_2_2::pipe< T >::reserve_read(), trisycl::detail::sycl_2_2::pipe< T >::reserved_for_writing(), trisycl::detail::sycl_2_2::pipe< T >::size(), and trisycl::detail::sycl_2_2::pipe< T >::write().

9.9.2.6.4.2 cb_mutex `template<typename T >`

`std::mutex trisycl::detail::sycl_2_2::pipe< T >::cb_mutex [mutable], [private]`

To protect the access to the circular buffer.

In case the object is capture in a lambda per copy, make it mutable.

Definition at line 89 of file pipe.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe< T >::empty_with_lock(), trisycl::detail::sycl_2_2::pipe< T >::full_with_lock(), trisycl::detail::sycl_2_2::pipe< T >::move_read_reservation_forward(), trisycl::detail::sycl_2_2::pipe< T >::move_write_reservation_, trisycl::detail::sycl_2_2::pipe< T >::read(), trisycl::detail::sycl_2_2::pipe< T >::reserve_read(), trisycl::detail::sycl_2_2::pipe< T >::re, trisycl::detail::sycl_2_2::pipe< T >::size_with_lock(), and trisycl::detail::sycl_2_2::pipe< T >::write().

9.9.2.6.4.3 debug_mode template<typename T >

bool trisycl::detail::sycl_2_2::pipe< T >::debug_mode = false [private]

To control the debug mode, disabled by default.

Definition at line 116 of file pipe.hpp.

9.9.2.6.4.4 r_rid_q template<typename T >

std::deque<reserve_id<value_type> > trisycl::detail::sycl_2_2::pipe< T >::r_rid_q [private]

The queue of pending read reservations.

Definition at line 104 of file pipe.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe< T >::move_read_reservation_forward(), and trisycl::detail::sycl_2_2::pipe< T >::reserv

9.9.2.6.4.5 read_done template<typename T >

std::condition_variable trisycl::detail::sycl_2_2::pipe< T >::read_done [private]

To signal that a read has been successful.

Definition at line 110 of file pipe.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe< T >::move_read_reservation_forward(), trisycl::detail::sycl_2_2::pipe< T >::read(), trisycl::detail::sycl_2_2::pipe< T >::reserve_write(), and trisycl::detail::sycl_2_2::pipe< T >::write().

9.9.2.6.4.6 read_reserved_frozen template<typename T >

std::size_t trisycl::detail::sycl_2_2::pipe< T >::read_reserved_frozen [private]

Track the number of frozen elements related to read reservations.

Definition at line 107 of file pipe.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe< T >::move_read_reservation_forward(), trisycl::detail::sycl_2_2::pipe< T >::read(), trisycl::detail::sycl_2_2::pipe< T >::reserve_read(), and trisycl::detail::sycl_2_2::pipe< T >::reserved_for_reading().

9.9.2.6.4.7 used_for_reading template<typename T >

bool trisycl::detail::sycl_2_2::pipe< T >::used_for_reading = false

True when the pipe is currently used for reading.

Definition at line 121 of file pipe.hpp.

9.9.2.6.4.8 used_for_writing template<typename T >

bool trisycl::detail::sycl_2_2::pipe< T >::used_for_writing = false

True when the pipe is currently used for writing.

Definition at line 124 of file pipe.hpp.

9.9.2.6.4.9 w_rid_q template<typename T >

std::deque<reserve_id<value_type> > trisycl::detail::sycl_2_2::pipe< T >::w_rid_q [private]

The queue of pending write reservations.

Definition at line 92 of file pipe.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe< T >::move_write_reservation_forward(), trisycl::detail::sycl_2_2::pipe< T >::reserve_w and trisycl::detail::sycl_2_2::pipe< T >::reserved_for_writing().

9.9.2.6.4.10 write_done template<typename T >

std::condition_variable trisycl::detail::sycl_2_2::pipe< T >::write_done [private]

To signal that a write has been successful.

Definition at line 113 of file pipe.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe< T >::move_write_reservation_forward(), trisycl::detail::sycl_2_2::pipe< T >::read(), trisycl::detail::sycl_2_2::pipe< T >::reserve_read(), and trisycl::detail::sycl_2_2::pipe< T >::write().

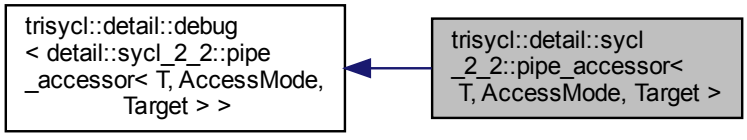
9.9.2.7 class trisycl::detail::sycl_2_2::pipe_accessor

template<typename T, access::mode AccessMode, access::target Target>
class trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >

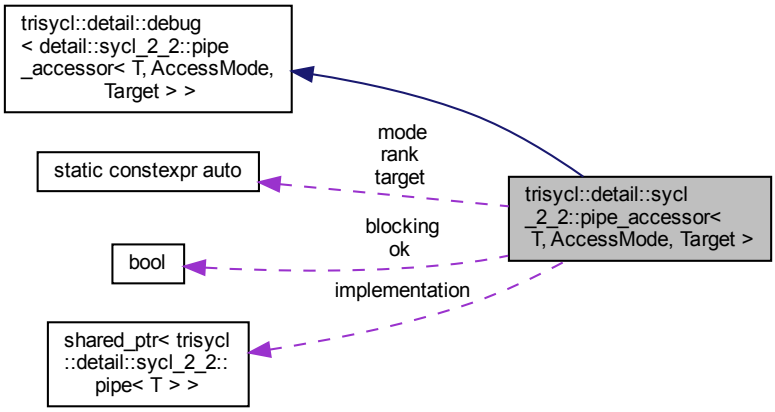
The accessor abstracts the way pipe data are accessed inside a kernel.

Definition at line 49 of file pipe_accessor.hpp.

Inheritance diagram for trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >:



Collaboration diagram for trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >:



Public Types

- using value_type = T

  The STL-like types.
- using reference = value_type &
- using const_reference = const value_type &

Public Member Functions

- pipe_accessor (const std::shared_ptr< detail::sycl_2_2::pipe< T > > &p)

  Construct a pipe accessor from an existing pipe.
- pipe_accessor (const std::shared_ptr< detail::sycl_2_2::pipe< T > > &p, handler &command_group_↩ handler)

  Construct a pipe accessor from an existing pipe.
- pipe_accessor ()=default
- std::size_t capacity () const

  Return the maximum number of elements that can fit in the pipe.
- std::size_t size () const

  Get the current number of elements in the pipe.
- bool empty () const

  Test if the pipe is empty.
- bool full () const

  Test if the pipe is full.
- operator bool () const

  In an explicit bool context, the accessor gives the success status of the last access.
- const pipe_accessor & write (const value_type &value) const

  Try to write a value to the pipe.
- const pipe_accessor & operator<< (const value_type &value) const

  Some syntactic sugar to use.
- const pipe_accessor & read (value_type &value) const

  Try to read a value from the pipe.

- value_type read () const

  Read a value from a blocking pipe.
- const pipe_accessor & operator>> (value_type &value) const

  Some syntactic sugar to use.
- detail::sycl_2_2::pipe_reservation< pipe_accessor > reserve (std::size_t size) const
- void set_debug (bool enable) const

  Set debug mode.
- auto & get_pipe_detail ()
- ~pipe_accessor ()

Static Public Attributes

- static constexpr auto rank = 1
- static constexpr auto mode = AccessMode
- static constexpr auto target = Target
- static constexpr bool blocking

Private Attributes

- std::shared_ptr< detail::sycl_2_2::pipe< T > > implementation

  The real pipe implementation behind the hood.
- bool ok = false

  Store the success status of last pipe operation.

Additional Inherited Members

9.9.2.7.1 Member Typedef Documentation

9.9.2.7.1.1 const_reference

```
template<typename T , access::mode AccessMode, access::target Target>
using trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::const_reference =
const value_type&
```

Definition at line 64 of file pipe_accessor.hpp.

9.9.2.7.1.2 reference

```
template<typename T , access::mode AccessMode, access::target Target>
using trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::reference = value_type&
```

Definition at line 63 of file pipe_accessor.hpp.

9.9.2.7.1.3 value_type

```
template<typename T , access::mode AccessMode, access::target Target>
using trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::value_type = T
```

The STL-like types.

Definition at line 62 of file pipe_accessor.hpp.

9.9.2.7.2 Constructor & Destructor Documentation

9.9.2.7.2.1 pipe_accessor() [1/3]

```
template<typename T , access::mode AccessMode, access::target Target>
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::pipe_accessor (
            const std::shared_ptr< detail::sycl_2_2::pipe< T > > & p )  [inline]
```

Construct a pipe accessor from an existing pipe.

Todo Use pipe_exception instead

Definition at line 88 of file pipe_accessor.hpp.

```
00089     :  implementation { p } {
00090     //    TRISYCL_DUMP_T("Create a kernel pipe accessor write = "
00091     //                   << is_write_access());
00092     // Verify that the pipe is not already used in the requested mode
00093     if (mode == access::mode::write)
00094       if (implementation->used_for_writing)
00095 /// \todo Use pipe_exception instead
00096         throw std::logic_error { "The pipe is already used for writing." };
00097       else
00098         implementation->used_for_writing = true;
00099     else
00100       if (implementation->used_for_reading)
00101         throw std::logic_error { "The pipe is already used for reading." };
00102       else
00103         implementation->used_for_reading = true;
00104   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::implementation, trisycl::detail::sycl_2_2::pipe_access and trisycl::access::write.

9.9.2.7.2.2 pipe_accessor() [2/3]

```
template<typename T , access::mode AccessMode, access::target Target>
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::pipe_accessor (
            const std::shared_ptr< detail::sycl_2_2::pipe< T > > & p,
            handler & command_group_handler )  [inline]
```

Construct a pipe accessor from an existing pipe.

For now the handler is not used.

Definition at line 111 of file pipe_accessor.hpp.

```
00113     :  pipe_accessor(p) {}
```

9.9.2.7.2.3 pipe_accessor() [3/3]

```
template<typename T , access::mode AccessMode, access::target Target>
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::pipe_accessor ( )  [default]
```

9.9.2.7.2.4 ~pipe_accessor() template<typename T , access::mode AccessMode, access::target Target>
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::~pipe_accessor ( ) [inline]

Free the pipe for a future usage for the current mode

Definition at line 286 of file pipe_accessor.hpp.
```
00286                   {
00287 /// Free the pipe for a future usage for the current mode
00288     if (mode == access::mode::write)
00289       implementation->used_for_writing = false;
00290     else
00291       implementation->used_for_reading = false;
00292   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::implementation, trisycl::detail::sycl_2_2::pipe_access and trisycl::access::write.

9.9.2.7.3 Member Function Documentation

9.9.2.7.3.1 capacity() template<typename T , access::mode AccessMode, access::target Target>
std::size_t trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::capacity ( ) const [inline]

Return the maximum number of elements that can fit in the pipe.

Definition at line 120 of file pipe_accessor.hpp.
```
00120                             {
00121     return implementation->capacity();
00122   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::implementation.

9.9.2.7.3.2 empty() template<typename T , access::mode AccessMode, access::target Target>
bool trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::empty ( ) const [inline]

Test if the pipe is empty.

This is obviously a volatile value which is constrained by restricted relativity.

Note that on some devices it may be costly to implement on the write side (for example on FPGA).

Definition at line 145 of file pipe_accessor.hpp.
```
00145                   {
00146     return implementation->empty_with_lock();
00147   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::implementation.

9.9.2.7.3.3 full() template<typename T , access::mode AccessMode, access::target Target> bool trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::full ( ) const [inline]

Test if the pipe is full.

This is obviously a volatile value which is constrained by restricted relativity.

Note that on some devices it may be costly to implement on the read side (for example on FPGA).

Definition at line 158 of file pipe_accessor.hpp.

```
00158                     {
00159     return implementation->full_with_lock();
00160   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::implementation.

9.9.2.7.3.4 get_pipe_detail() template<typename T , access::mode AccessMode, access::target Target> auto & trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::get_pipe_detail ( ) [inline]

Definition at line 281 of file pipe_accessor.hpp.

```
00281                               {
00282     return implementation;
00283   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::implementation.

9.9.2.7.3.5 operator bool() template<typename T , access::mode AccessMode, access::target Target> trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::operator bool ( ) const [inline], [explicit]

In an explicit bool context, the accessor gives the success status of the last access.

It is not impacted by reservation success.

The explicitness is related to avoid

```
 some_pipe «
some_value
```

to be interpreted as

```
 some_bool «
some_value
```

when the type of

```
some_value
```

is not the same type as the pipe type.

Returns

true on success of the previous read or write operation

Definition at line 175 of file pipe_accessor.hpp.

```
00175                                {
00176     return ok;
00177   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::ok.

9.9.2.7.3.6 operator<<()

```
template<typename T , access::mode AccessMode, access::target Target>
const pipe_accessor & trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >↩
::operator<< (
              const value_type & value ) const  [inline]
```

Some syntactic sugar to use.

```
a « v
```

instead of

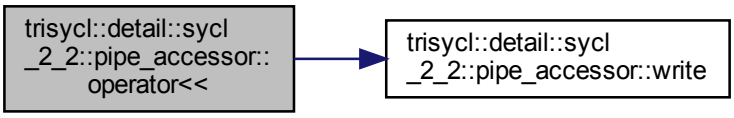
```
a.write(v)
```

Definition at line 205 of file pipe_accessor.hpp.

```
00205                                                                    {
00206     static_assert(mode == access::mode::write,
00207                   "'«' operator on a pipe accessor is only possible"
00208                   " with write access mode");
00209     // Return a reference to *this so we can apply a sequence of «
00210     return write(value);
00211   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::mode, trisycl::access::write, and trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::write().

Here is the call graph for this function:



9.9.2.7.3.7 operator>>()

```
template<typename T , access::mode AccessMode, access::target Target>
const pipe_accessor & trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >↩
::operator>> (
              value_type & value ) const  [inline]
```

Some syntactic sugar to use.

```
a » v
```

instead of

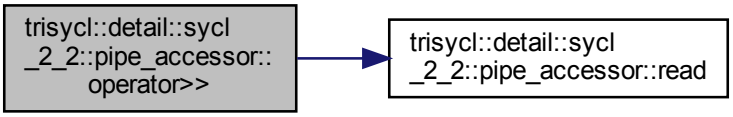
```
a.read(v)
```

Definition at line 260 of file pipe_accessor.hpp.

```
00260                                                                    {
00261     static_assert(mode == access::mode::read,
00262                   "'»' operator on a pipe accessor is only possible"
00263                   " with read access mode");
00264     // Return a reference to *this so we can apply a sequence of »
00265     return read(value);
00266   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::mode, trisycl::access::read, and trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::read().

Here is the call graph for this function:



9.9.2.7.3.8 read() [1/2]

```
template<typename T , access::mode AccessMode, access::target Target>
value_type trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::read ( ) const
[inline]
```

Read a value from a blocking pipe.

Returns

the read value directly, since it cannot fail on blocking pipe

This function is const so it can work when the accessor is passed by copy in the [=] kernel lambda, which is not mutable by default

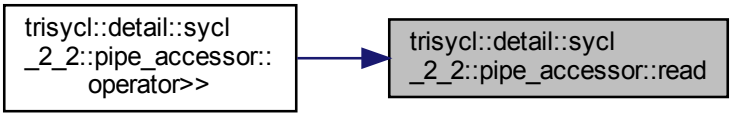
Definition at line 245 of file pipe_accessor.hpp.

```
00245                                {
00246     static_assert(mode == access::mode::read,
00247                   "'.read()' method on a pipe accessor is only possible"
00248                   " with read access mode");
00249     static_assert(blocking,
00250                   "'.read()' method on a pipe accessor is only possible"
00251                   " with a blocking pipe");
00252     value_type value;
00253     implementation->read(value, blocking);
00254     return value;
00255   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::blocking, trisycl::detail::sycl_2_2::pipe_accessor< T, trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::mode, and trisycl::access::read.

Referenced by trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::operator>>().

Here is the caller graph for this function:



9.9.2.7.3.9 read() [2/2]

```
template<typename T , access::mode AccessMode, access::target Target>
const pipe_accessor & trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::read
(
            value_type & value ) const  [inline]
```

Try to read a value from the pipe.

Parameters

out	*value*	is the reference to where to store what is read

Returns

`this`

so we can apply a sequence of read for example (but do not do this on a non blocking pipe...)

This function is const so it can work when the accessor is passed by copy in the [=] kernel lambda, which is not mutable by default

Definition at line 226 of file pipe_accessor.hpp.

```
00226                                                {
00227     static_assert(mode == access::mode::read,
00228                   "'.read(value_type &value)' method on a pipe accessor"
00229                   " is only possible with read access mode");
00230     ok = implementation->read(value, blocking);
00231     // Return a reference to *this so we can apply a sequence of read
00232     return *this;
00233   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::blocking, trisycl::detail::sycl_2_2::pipe_accessor< T, trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::mode, trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, and trisycl::access::read.

9.9.2.7.3.10 reserve()

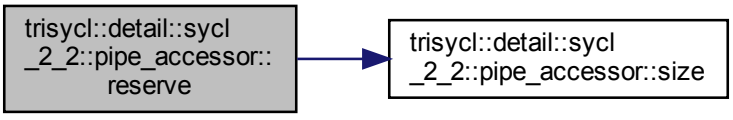
```
template<typename T , access::mode AccessMode, access::target Target>
detail::sycl_2_2::pipe_reservation< pipe_accessor > trisycl::detail::sycl_2_2::pipe_accessor<
T, AccessMode, Target >::reserve (
            std::size_t size ) const  [inline]
```

Definition at line 270 of file pipe_accessor.hpp.

```
00270                                        {
00271     return { *implementation, size };
00272   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::implementation, and trisycl::detail::sycl_2_2::pipe_ac

Here is the call graph for this function:



9.9.2.7.3.11 set_debug()

```
template<typename T , access::mode AccessMode, access::target Target>
void trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::set_debug (
            bool enable ) const  [inline]
```

Set debug mode.

Definition at line 276 of file pipe_accessor.hpp.

```
00276                                    {
00277     implementation->debug_mode = enable;
00278   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::implementation.

9.9.2.7.3.12 size()

```
template<typename T , access::mode AccessMode, access::target Target>
std::size_t trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::size ( ) const
[inline]
```

Get the current number of elements in the pipe.

This is obviously a volatile value which is constrained by restricted relativity.

Note that on some devices it may be costly to implement (for example on FPGA).

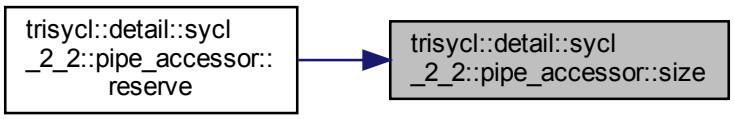
Definition at line 132 of file pipe_accessor.hpp.

```
00132                          {
00133     return implementation->size_with_lock();
00134   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::implementation.

Referenced by trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::reserve().

Here is the caller graph for this function:



9.9.2.7.3.13 write()

```
template<typename T , access::mode AccessMode, access::target Target>
const pipe_accessor & trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::write
(
            const value_type & value ) const  [inline]
```

Try to write a value to the pipe.

Parameters

in	*value*	is what we want to write

Returns

this so we can apply a sequence of write for example (but do not do this on a non blocking pipe...)

Todo provide a && version

This function is const so it can work when the accessor is passed by copy in the [=] kernel lambda, which is not mutable by default

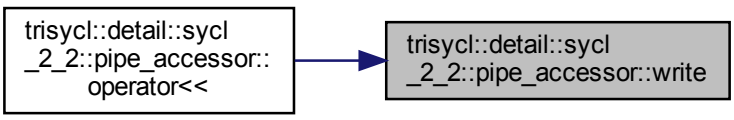
Definition at line 193 of file pipe_accessor.hpp.

```
00193                                                                {
00194     static_assert(mode == access::mode::write,
00195                   "'.write(const value_type &value)' method on a pipe accessor"
00196                   " is only possible with write access mode");
00197     ok = implementation->write(value, blocking);
00198     // Return a reference to *this so we can apply a sequence of write
00199     return *this;
00200   }
```

References trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::blocking, trisycl::detail::sycl_2_2::pipe_accessor< T, trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::mode, trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, and trisycl::access::write.

Referenced by trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::operator<<().

Here is the caller graph for this function:



9.9.2.7.4 Member Data Documentation

9.9.2.7.4.1 blocking

```
template<typename T , access::mode AccessMode, access::target Target>
constexpr bool trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::blocking
[static], [constexpr]
```

Initial value:

```
=
    (target == trisycl::access::target::blocking_pipe)
```

Definition at line 58 of file pipe_accessor.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::read(), and trisycl::detail::sycl_2_2::pipe_accesso

9.9.2.7.4.2 implementation

`template<typename T , access::mode AccessMode, access::target Target> std::shared_ptr<detail::sycl_2_2::pipe<T> > trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::implementation [private]`

The real pipe implementation behind the hood.

Definition at line 69 of file pipe_accessor.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::capacity(), trisycl::detail::sycl_2_2::pipe_accessor
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::full(), trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, T
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::pipe_accessor(), trisycl::detail::sycl_2_2::pipe_accessor< T, Acc
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::reserve(), trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMod
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::size(), trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode,
and trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::~pipe_accessor().

9.9.2.7.4.3 mode

`template<typename T , access::mode AccessMode, access::target Target> constexpr auto trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::mode = AccessMode [static], [constexpr]`

Definition at line 55 of file pipe_accessor.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::operator<<(), trisycl::detail::sycl_2_2::pipe_acces
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::pipe_accessor(), trisycl::detail::sycl_2_2::pipe_accessor< T, Acc
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::write(), and trisycl::detail::sycl_2_2::pipe_accessor< T, AccessM

9.9.2.7.4.4 ok

`template<typename T , access::mode AccessMode, access::target Target> bool trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::ok = false [mutable], [private]`

Store the success status of last pipe operation.

It is not impacted by reservation success.

It does exist even if the pipe accessor is not evaluated in a boolean context for, but a use-def analysis can optimise it out in that case and not use some storage

Use a mutable state here so that it can work with a [=] lambda capture without having to declare the whole lambda as mutable

Definition at line 82 of file pipe_accessor.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::operator bool(), trisycl::detail::sycl_2_2::pipe_acce
and trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::write().

9.9.2.7.4.5 rank

`template<typename T , access::mode AccessMode, access::target Target> constexpr auto trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::rank = 1 [static], [constexpr]`

Definition at line 54 of file pipe_accessor.hpp.

9.9.2.7.4.6 target `template<typename T , access::mode AccessMode, access::target Target> constexpr auto trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >::target = Target [static], [constexpr]`

Definition at line 56 of file pipe_accessor.hpp.

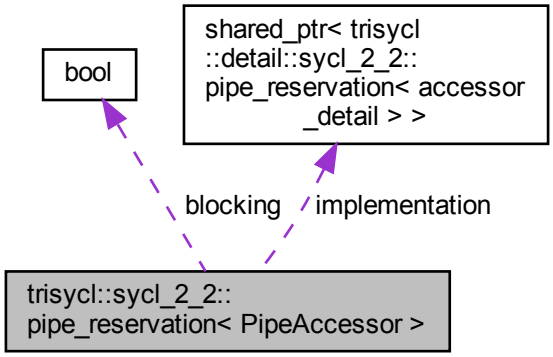
9.9.2.8 struct trisycl::sycl_2_2::pipe_reservation

template<typename PipeAccessor>
struct trisycl::sycl_2_2::pipe_reservation< PipeAccessor >

The pipe reservation station allows to reserve an array-like view inside the pipe for ordered race-free access from various work-items for example.

Definition at line 32 of file pipe_reservation.hpp.

Collaboration diagram for trisycl::sycl_2_2::pipe_reservation< PipeAccessor >:



Public Types

- using accessor_type = PipeAccessor
- using accessor_detail = typename accessor_type::accessor_detail
- using value_type = typename accessor_type::value_type

  The STL-like types.
- using reference = value_type &
- using const_reference = const value_type &
- using pointer = value_type *
- using const_pointer = const value_type *
- using size_type = std::size_t
- using difference_type = ptrdiff_t
- using iterator = typename detail::sycl_2_2::pipe_reservation< accessor_detail >::iterator
- using const_iterator = typename detail::sycl_2_2::pipe_reservation< accessor_detail >::const_iterator
- using reverse_iterator = std::reverse_iterator< iterator >
- using const_reverse_iterator = std::reverse_iterator< const_iterator >

Public Member Functions

- pipe_reservation ()=default

  Use default constructors so that we can create a new buffer copy from another one, with either a l-value or a r-value (for std::move() for example).
- pipe_reservation (accessor_type &accessor, std::size_t s)

  Create a pipe_reservation for an accessor and a number of elements.
- pipe_reservation (detail::sycl_2_2::pipe_reservation< accessor_detail > &&pr)

  Create a pipe_reservation from the implementation detail.
- operator bool () const

  Test if the pipe_reservation has been correctly allocated.
- std::size_t size () const

  Get the number of reserved element(s)
- reference operator[ ] (std::size_t index) const

  Access to a given element of the reservation.
- void commit () const

  Force a commit operation.
- iterator begin () const

  Get an iterator on the first element of the reservation station.
- iterator end () const

  Get an iterator past the end of the reservation station.
- const_iterator cbegin () const

  Build a constant iterator on the first element of the reservation station.
- const_iterator cend () const

  Build a constant iterator past the end of the reservation station.
- reverse_iterator rbegin () const

  Get a reverse iterator on the last element of the reservation station.
- reverse_iterator rend () const

  Get a reverse iterator on the first element past the end of the reservation station.
- const_reverse_iterator crbegin () const

  Get a constant reverse iterator on the last element of the reservation station.
- const_reverse_iterator crend () const

  Get a constant reverse iterator on the first element past the end of the reservation station.

Public Attributes

- std::shared_ptr< detail::sycl_2_2::pipe_reservation< accessor_detail > > implementation

  Point to the underlying implementation that can be shared in the SYCL model with a handler semantics.

Static Public Attributes

- static constexpr bool blocking

9.9.2.8.1 Member Typedef Documentation

9.9.2.8.1.1 accessor_detail `template<typename PipeAccessor >`

```
using trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::accessor_detail = typename accessor↩
_type::accessor_detail
```

Definition at line 36 of file pipe_reservation.hpp.

9.9.2.8.1.2 accessor_type

```
template<typename PipeAccessor >
using trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::accessor_type = PipeAccessor
```

Definition at line 33 of file pipe_reservation.hpp.

9.9.2.8.1.3 const_iterator

```
template<typename PipeAccessor >
using trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::const_iterator = typename detail::sycl_2_2::pipe_r
::const_iterator
```

Definition at line 47 of file pipe_reservation.hpp.

9.9.2.8.1.4 const_pointer

```
template<typename PipeAccessor >
using trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::const_pointer = const value_type*
```

Definition at line 42 of file pipe_reservation.hpp.

9.9.2.8.1.5 const_reference

```
template<typename PipeAccessor >
using trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::const_reference = const value_type&
```

Definition at line 40 of file pipe_reservation.hpp.

9.9.2.8.1.6 const_reverse_iterator

```
template<typename PipeAccessor >
using trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::const_reverse_iterator = std↩
::reverse_iterator<const_iterator>
```

Definition at line 50 of file pipe_reservation.hpp.

9.9.2.8.1.7 difference_type

```
template<typename PipeAccessor >
using trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::difference_type = ptrdiff_t
```

Definition at line 44 of file pipe_reservation.hpp.

9.9.2.8.1.8 iterator

```
template<typename PipeAccessor >
using trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::iterator = typename detail::sycl_2_2::pipe_reserva
::iterator
```

Definition at line 45 of file pipe_reservation.hpp.

9.9.2.8.1.9 pointer template<typename PipeAccessor >

```
using trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::pointer = value_type*
```

Definition at line 41 of file pipe_reservation.hpp.

9.9.2.8.1.10 reference template<typename PipeAccessor >

```
using trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::reference = value_type&
```

Definition at line 39 of file pipe_reservation.hpp.

9.9.2.8.1.11 reverse_iterator template<typename PipeAccessor >

```
using trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::reverse_iterator = std::reverse_↩
iterator<iterator>
```

Definition at line 49 of file pipe_reservation.hpp.

9.9.2.8.1.12 size_type template<typename PipeAccessor >

```
using trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::size_type = std::size_t
```

Definition at line 43 of file pipe_reservation.hpp.

9.9.2.8.1.13 value_type template<typename PipeAccessor >

```
using trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::value_type = typename accessor_↩
type::value_type
```

The STL-like types.

Definition at line 38 of file pipe_reservation.hpp.

9.9.2.8.2 Constructor & Destructor Documentation

9.9.2.8.2.1 pipe_reservation() [1/3] template<typename PipeAccessor >

```
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::pipe_reservation ( ) [default]
```

Use default constructors so that we can create a new buffer copy from another one, with either a l-value or a r-value (for std::move() for example).

Since we just copy the shared_ptr<> above, this is where/how the sharing magic is happening with reference counting in this case.

9.9.2.8.2.2 pipe_reservation() [2/3] template<typename PipeAccessor >

```
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::pipe_reservation (
            accessor_type & accessor,
            std::size_t s )  [inline]
```

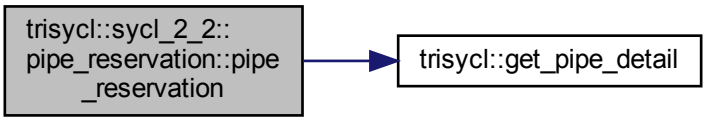
Create a pipe_reservation for an accessor and a number of elements.

Definition at line 68 of file pipe_reservation.hpp.

```
00069     : implementation {
00070     new detail::sycl_2_2::pipe_reservation<accessor_detail> {
00071       get_pipe_detail(accessor), s }
00072   } {}
```

References trisycl::get_pipe_detail().

Here is the call graph for this function:



9.9.2.8.2.3 pipe_reservation() [3/3] template<typename PipeAccessor >

```
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::pipe_reservation (
            detail::sycl_2_2::pipe_reservation< accessor_detail > && pr )  [inline]
```

Create a pipe_reservation from the implementation detail.

This is an internal constructor to allow reserve() on the implementation to lift a full-fledged object through accessor::reserve().

Todo Make it private and add required friends

Definition at line 83 of file pipe_reservation.hpp.

```
00084     : implementation {
00085     new detail::sycl_2_2::pipe_reservation<accessor_detail> { std::move(pr) } }
00086   {}
```

9.9.2.8.3 Member Function Documentation

9.9.2.8.3.1 begin() template<typename PipeAccessor >

```
iterator trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::begin ( ) const  [inline]
```

Get an iterator on the first element of the reservation station.

Definition at line 121 of file pipe_reservation.hpp.

```
00121                          {
00122     return implementation->begin();
00123   }
```

9.9.2.8.3.2 cbegin() template<typename PipeAccessor >

const_iterator trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::cbegin ( ) const [inline]

Build a constant iterator on the first element of the reservation station.

Definition at line 133 of file pipe_reservation.hpp.

```
00133                                  {
00134     return implementation->begin();
00135   }
```

9.9.2.8.3.3 cend() template<typename PipeAccessor >

const_iterator trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::cend ( ) const [inline]

Build a constant iterator past the end of the reservation station.

Definition at line 139 of file pipe_reservation.hpp.

```
00139                                {
00140     return implementation->end();
00141   }
```

9.9.2.8.3.4 commit() template<typename PipeAccessor >

void trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::commit ( ) const [inline]

Force a commit operation.

Normally the commit is implicitly done in the destructor, but sometime it is useful to do it earlier.

Definition at line 115 of file pipe_reservation.hpp.

```
00115                          {
00116     return implementation->commit();
00117   }
```

9.9.2.8.3.5 crbegin() template<typename PipeAccessor >

const_reverse_iterator trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::crbegin ( ) const [inline]

Get a constant reverse iterator on the last element of the reservation station.

Definition at line 159 of file pipe_reservation.hpp.

```
00159                                          {
00160     return std::make_reverse_iterator(cend());
00161   }
```

9.9.2.8.3.6 crend() template<typename PipeAccessor >

const_reverse_iterator trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::crend ( ) const [inline]

Get a constant reverse iterator on the first element past the end of the reservation station.

Definition at line 166 of file pipe_reservation.hpp.

```
00166                                        {
00167     return std::make_reverse_iterator(cbegin());
00168   }
```

9.9.2.8.3.7 end() template<typename PipeAccessor >

iterator trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::end ( ) const [inline]

Get an iterator past the end of the reservation station.

Definition at line 127 of file pipe_reservation.hpp.

```
00127                         {
00128     return implementation->end();
00129   }
```

9.9.2.8.3.8 operator bool() template<typename PipeAccessor >

trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::operator bool ( ) const [inline]

Test if the pipe_reservation has been correctly allocated.

Returns

true if the pipe_reservation can be used and committed

Definition at line 93 of file pipe_reservation.hpp.

```
00093                         {
00094     return *implementation;
00095   }
```

9.9.2.8.3.9 operator[]() template<typename PipeAccessor >

reference trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::operator[] (
std::size_t *index*) const [inline]

Access to a given element of the reservation.

Definition at line 105 of file pipe_reservation.hpp.

```
00105                                          {
00106     return (*implementation)[index];
00107   }
```

9.9.2.8.3.10 rbegin() template<typename PipeAccessor >

reverse_iterator trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::rbegin ( ) const [inline]

Get a reverse iterator on the last element of the reservation station.

Definition at line 145 of file pipe_reservation.hpp.

```
00145                                  {
00146     return std::make_reverse_iterator(end());
00147   }
```

9.9.2.8.3.11 rend() template<typename PipeAccessor >

reverse_iterator trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::rend ( ) const [inline]

Get a reverse iterator on the first element past the end of the reservation station.

Definition at line 152 of file pipe_reservation.hpp.

```
00152                                {
00153     return std::make_reverse_iterator(begin());
00154   }
```

9.9.2.8.3.12 size() template<typename PipeAccessor >

std::size_t trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::size ( ) const [inline]

Get the number of reserved element(s)

Definition at line 99 of file pipe_reservation.hpp.

```
00099                          {
00100     return implementation->size();
00101   }
```

9.9.2.8.4 Member Data Documentation

9.9.2.8.4.1 blocking template<typename PipeAccessor >

constexpr bool trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::blocking [static], [constexpr]

Initial value:

```
=
    (accessor_type::target == trisycl::access::target::blocking_pipe)
```

Definition at line 34 of file pipe_reservation.hpp.

9.9.2.8.4.2 implementation template<typename PipeAccessor >

std::shared_ptr<detail::sycl_2_2::pipe_reservation<accessor_detail> > trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::implementation

Point to the underlying implementation that can be shared in the SYCL model with a handler semantics.

Definition at line 55 of file pipe_reservation.hpp.

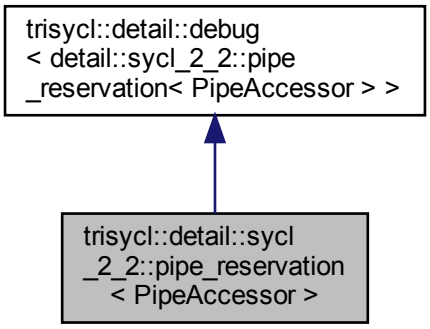
9.9.2.9 class trisycl::detail::sycl_2_2::pipe_reservation

template<typename PipeAccessor>
class trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >

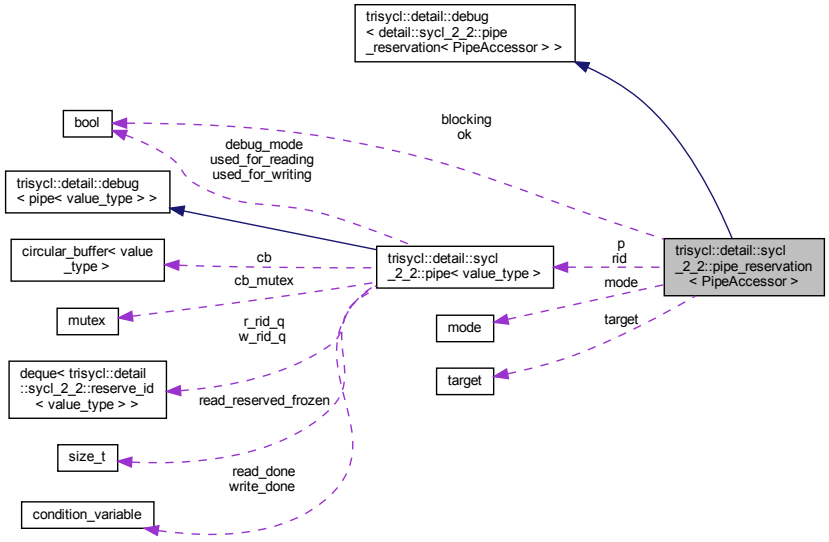
The implementation of the pipe reservation station.

Definition at line 36 of file pipe_reservation.hpp.

Inheritance diagram for trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >:



Collaboration diagram for trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >:



Public Types

- using iterator = typename detail::sycl_2_2::pipe< value_type >::implementation_t::iterator
- using const_iterator = typename detail::sycl_2_2::pipe< value_type >::implementation_t::const_iterator

Public Member Functions

- void assume_validity ()

  Test that the reservation is in a usable state.
- pipe_reservation (detail::sycl_2_2::pipe< value_type > &p, std::size_t s)

  Create a pipe reservation station that reserves the pipe itself.
- pipe_reservation (const pipe_reservation &)=delete

  No copy constructor with some spurious commit in the destructor of the original object.
- pipe_reservation (pipe_reservation &&orig)

  Only a move constructor is required to move it into the shared_ptr.
- pipe_reservation ()=default

  Keep the default constructors too.
- operator bool ()

  Test if the reservation succeeded and thus if the reservation can be committed.
- iterator begin ()

  Start of the reservation area.
- iterator end ()

  Past the end of the reservation area.
- std::size_t size ()

  Get the number of elements in the reservation station.
- reference operator[ ] (std::size_t index)

  Access to an element of the reservation.
- void commit ()

  Commit the reservation station.
- ~pipe_reservation ()

  An implicit commit is made in the destructor.

Public Attributes

- bool ok = false

  True if the reservation was successful and still uncommitted.
- detail::sycl_2_2::pipe< value_type >::rid_iterator rid

  Point into the reservation buffer. Only valid if ok is true.
- detail::sycl_2_2::pipe< value_type > & p

  Keep a reference on the pipe to access to the data and methods.

Static Public Attributes

- static constexpr access::mode mode = accessor_type::mode
- static constexpr access::target target = accessor_type::target

Private Types

- using accessor_type = PipeAccessor
- using value_type = typename accessor_type::value_type
- using reference = typename accessor_type::reference

Static Private Attributes

- static constexpr bool blocking

Additional Inherited Members

9.9.2.9.1 Member Typedef Documentation

9.9.2.9.1.1 accessor_type `template<typename PipeAccessor >`

```
using trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::accessor_type = Pipe↩
Accessor  [private]
```

Definition at line 38 of file pipe_reservation.hpp.

9.9.2.9.1.2 const_iterator `template<typename PipeAccessor >`

```
using trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::const_iterator = typename
detail::sycl_2_2::pipe<value_type>::implementation_t::const_iterator
```

Definition at line 48 of file pipe_reservation.hpp.

9.9.2.9.1.3 iterator `template<typename PipeAccessor >`

```
using trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::iterator = typename detail::sycl_2_2::pipe
::implementation_t::iterator
```

Definition at line 46 of file pipe_reservation.hpp.

9.9.2.9.1.4 reference `template<typename PipeAccessor >`

```
using trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::reference = typename accessor↩
_type::reference  [private]
```

Definition at line 42 of file pipe_reservation.hpp.

9.9.2.9.1.5 value_type `template<typename PipeAccessor >`

```
using trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::value_type = typename
accessor_type::value_type  [private]
```

Definition at line 41 of file pipe_reservation.hpp.

9.9.2.9.2 Constructor & Destructor Documentation

9.9.2.9.2.1 pipe_reservation() [1/4] template<typename PipeAccessor >

```
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::pipe_reservation (
            detail::sycl_2_2::pipe< value_type > & p,
            std::size_t s )  [inline]
```

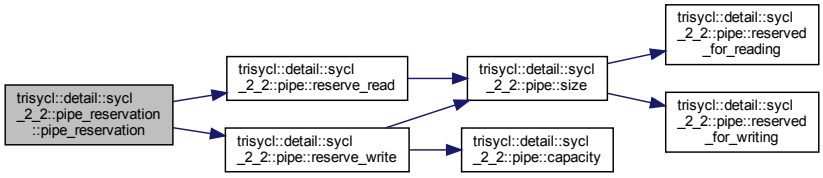
Create a pipe reservation station that reserves the pipe itself.

Definition at line 81 of file pipe_reservation.hpp.

```
00083     :  p { p } {
00084     static_assert(mode == access::mode::write
00085                   || mode == access::mode::read,
00086                   "A pipe can only be accesed in read or write mode,"
00087                   " exclusively");
00088
00089     /* Since this test is constexpr and dependent of a template
00090 parameter, it should be equivalent to a specialization of the
00091 method but in a clearer way */
00092     if (mode == access::mode::write)
00093       ok = p.reserve_write(s, rid, blocking);
00094     else
00095       ok = p.reserve_read(s, rid, blocking);
00096   }
```

References trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::blocking, trisycl::detail::sycl_2_2::pipe_reservation< PipeAc
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::ok, trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::p,
trisycl::access::read, trisycl::detail::sycl_2_2::pipe< T >::reserve_read(), trisycl::detail::sycl_2_2::pipe< T >::reserve_write(),
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::rid, and trisycl::access::write.

Here is the call graph for this function:



9.9.2.9.2.2 pipe_reservation() [2/4] template<typename PipeAccessor >

```
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::pipe_reservation (
            const pipe_reservation< PipeAccessor > &  )  [delete]
```

No copy constructor with some spurious commit in the destructor of the original object.

9.9.2.9.2.3 pipe_reservation() [3/4] template<typename PipeAccessor >

```
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::pipe_reservation (
            pipe_reservation< PipeAccessor > && orig )  [inline]
```

Only a move constructor is required to move it into the shared_ptr.

Definition at line 106 of file pipe_reservation.hpp.

```
00106                                         :
00107     ok {orig.ok },
00108     rid {orig.rid },
00109     p { orig.p } {
00110       /* Even when an object is moved, the destructor of the old
00111 object is eventually called, so leave the old object in a
00112 destructable state but without any commit capability */
00113       orig.ok = false;
00114     }
```

9.9.2.9.2.4 pipe_reservation() [4/4] template<typename PipeAccessor >

trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::pipe_reservation ( ) [default]

Keep the default constructors too.

Otherwise there is no move semantics and the copy is made by creating a new reservation and destructing the old one with a spurious commit in the meantime...

9.9.2.9.2.5 ~pipe_reservation() template<typename PipeAccessor >

trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::~pipe_reservation ( ) [inline]

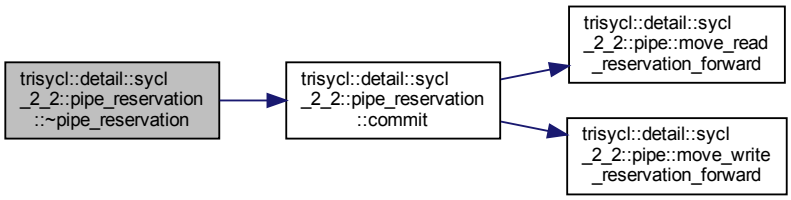
An implicit commit is made in the destructor.

Definition at line 190 of file pipe_reservation.hpp.

```
00190                                {
00191     commit();
00192   }
```

References trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::commit().

Here is the call graph for this function:



9.9.2.9.3 Member Function Documentation

9.9.2.9.3.1 assume_validity() template<typename PipeAccessor >

void trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::assume_validity ( ) [inline]

Test that the reservation is in a usable state.

Todo Throw exception instead

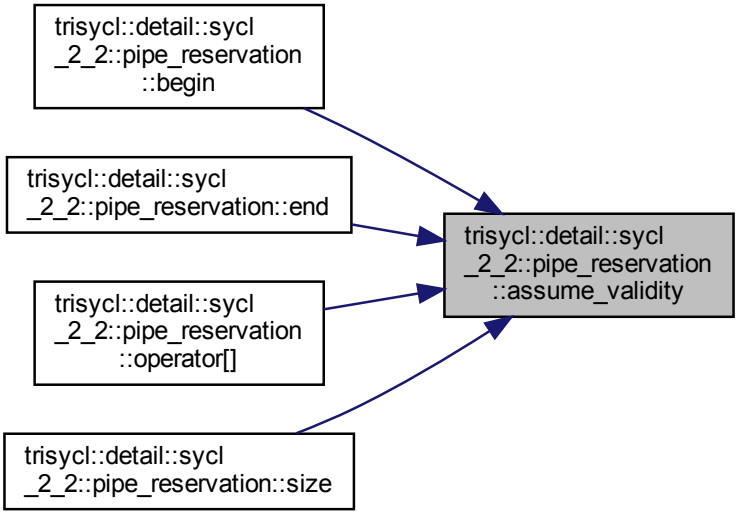
Definition at line 74 of file pipe_reservation.hpp.

```
00074                                {
00075     assert(ok);
00076   }
```

References trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::ok.

Referenced by trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::begin(), trisycl::detail::sycl_2_2::pipe_reservation< PipeA
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::operator[](), and trisycl::detail::sycl_2_2::pipe_reservation< PipeAccess

Here is the caller graph for this function:



9.9.2.9.3.2 begin() `template<typename PipeAccessor >`

`iterator trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::begin ( ) [inline]`

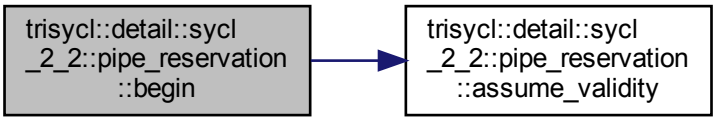
Start of the reservation area.

Definition at line 139 of file pipe_reservation.hpp.

```
00139                 {
00140     assume_validity();
00141     return rid->start;
00142   }
```

References trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::assume_validity(), and trisycl::detail::sycl_2_2::pipe_reserva

Here is the call graph for this function:



9.9.2.9.3.3 commit() template<typename PipeAccessor >

void trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::commit ( ) [inline]

Commit the reservation station.

Todo Add to the specification that for simplicity a reservation can be commited several times but only the first one is taken into account

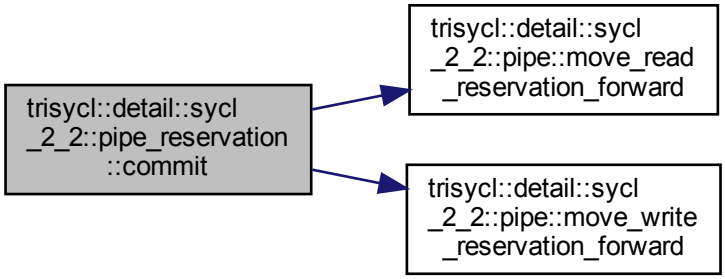
Definition at line 175 of file pipe_reservation.hpp.

```
00175                 {
00176     if (ok) {
00177       // If the reservation is in a committable state, commit
00178       TRISYCL_DUMP_T("Commit");
00179       rid->ready = true;
00180       if (mode == access::mode::write)
00181         p.move_write_reservation_forward();
00182       else
00183         p.move_read_reservation_forward();
00184       ok = false;
00185     }
00186   }
```

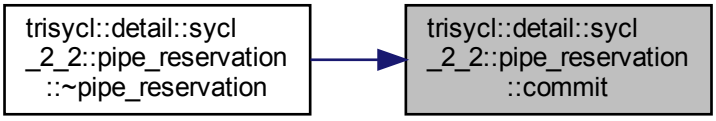
References trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::mode, trisycl::detail::sycl_2_2::pipe< T >::move_read_rese trisycl::detail::sycl_2_2::pipe< T >::move_write_reservation_forward(), trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >:: trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::p, trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::rid, TRISYCL_DUMP_T, and trisycl::access::write.

Referenced by trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::~pipe_reservation().

Here is the call graph for this function:



Here is the caller graph for this function:



9.9.2.9.3.4 end() template<typename PipeAccessor >

iterator trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::end ( ) [inline]

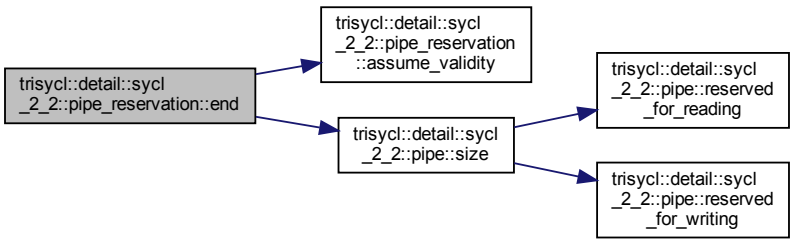
Past the end of the reservation area.

Definition at line 146 of file pipe_reservation.hpp.

```
00146                 {
00147     assume_validity();
00148     return rid->start + rid->size;
00149   }
```

References trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::assume_validity(), trisycl::detail::sycl_2_2::pipe_reservation and trisycl::detail::sycl_2_2::pipe< T >::size().

Here is the call graph for this function:



9.9.2.9.3.5 operator bool() template<typename PipeAccessor >

trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::operator bool ( ) [inline]

Test if the reservation succeeded and thus if the reservation can be committed.

Note that it is up to the user to ensure that all the reservation elements have been initialized correctly in the case of a write for example

Definition at line 133 of file pipe_reservation.hpp.

```
00133                  {
00134     return ok;
00135   }
```

References trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::ok.

9.9.2.9.3.6 operator[]() template<typename PipeAccessor >

```
reference trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::operator[] (
            std::size_t index )  [inline]
```

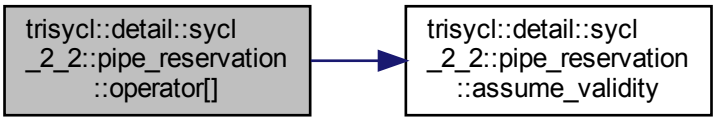
Access to an element of the reservation.

Definition at line 160 of file pipe_reservation.hpp.

```
00160                                       {
00161     assume_validity();
00162     TRISYCL_DUMP_T("[] index = " « index
00163                    « " Reservation write address = " « &(rid->start[index]));
00164
00165     return rid->start[index];
00166   }
```

References trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::assume_validity(), trisycl::detail::sycl_2_2::pipe_reservation and TRISYCL_DUMP_T.

Here is the call graph for this function:



9.9.2.9.3.7 size() template<typename PipeAccessor >

```
std::size_t trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::size ( )  [inline]
```

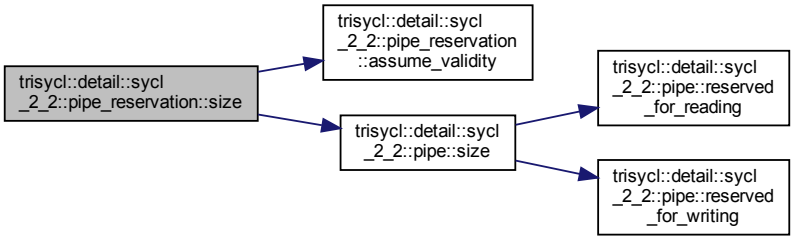
Get the number of elements in the reservation station.

Definition at line 153 of file pipe_reservation.hpp.

```
00153                     {
00154     assume_validity();
00155     return rid->size;
00156   }
```

References trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::assume_validity(), trisycl::detail::sycl_2_2::pipe_reservation and trisycl::detail::sycl_2_2::pipe< T >::size().

Here is the call graph for this function:



9.9.2.9.4 Member Data Documentation

9.9.2.9.4.1 blocking `template<typename PipeAccessor >`

`constexpr bool trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::blocking [static], [constexpr], [private]`

Initial value:

```
=
    (accessor_type::target == trisycl::access::target::blocking_pipe)
```

Definition at line 39 of file pipe_reservation.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::pipe_reservation().

9.9.2.9.4.2 mode `template<typename PipeAccessor >`

`constexpr access::mode trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::mode = accessor_type::mode [static], [constexpr]`

Definition at line 52 of file pipe_reservation.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::commit(), and trisycl::detail::sycl_2_2::pipe_reservation<

9.9.2.9.4.3 ok `template<typename PipeAccessor >`

`bool trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::ok = false`

True if the reservation was successful and still uncommitted.

B default a pipe_reservation is not reserved and cannot be committed

Definition at line 58 of file pipe_reservation.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::assume_validity(), trisycl::detail::sycl_2_2::pipe_reservat trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::operator bool(), and trisycl::detail::sycl_2_2::pipe_reservation< PipeAcc

9.9.2.9.4.4 p `template<typename PipeAccessor >`

`detail::sycl_2_2::pipe<value_type>& trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::p`

Keep a reference on the pipe to access to the data and methods.

Note that with inlining and CSE it should not use more register when compiler optimization is in use.

Definition at line 67 of file pipe_reservation.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::commit(), and trisycl::detail::sycl_2_2::pipe_reservation<

9.9.2.9.4.5 rid `template<typename PipeAccessor >`

`detail::sycl_2_2::pipe<value_type>::rid_iterator trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::rid`

Point into the reservation buffer. Only valid if ok is true.

Definition at line 61 of file pipe_reservation.hpp.

Referenced by trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::begin(), trisycl::detail::sycl_2_2::pipe_reservation< PipeA trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::end(), trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::oper trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::pipe_reservation(), and trisycl::detail::sycl_2_2::pipe_reservation< Pipe

9.9.2.9.4.6 target `template<typename PipeAccessor >`

`constexpr access::target trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >::target = accessor_type::target [static], [constexpr]`

Definition at line 53 of file pipe_reservation.hpp.

9.9.2.10 class trisycl::sycl_2_2::static_pipe

template< typename T, std::size_t Capacity >
class trisycl::sycl_2_2::static_pipe< T, Capacity >

A SYCL static-scoped pipe equivalent to an OpenCL program-scoped pipe.

Implement a FIFO-style object that can be used through accessors to send some objects T from the input to the output.

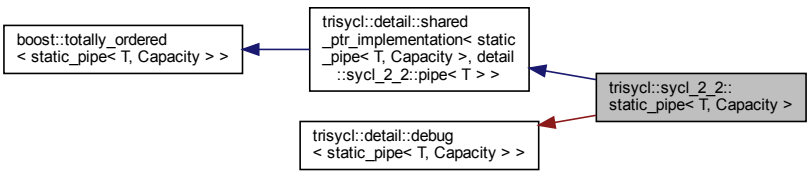
Compared to a normal pipe, a static_pipe takes a constexpr size and is expected to be declared in a compile-unit static context so the compiler can generate everything at compile time.

This is useful to generate a fixed and optimized hardware implementation on FPGA for example, where the inter-connection graph can be also inferred at compile time.

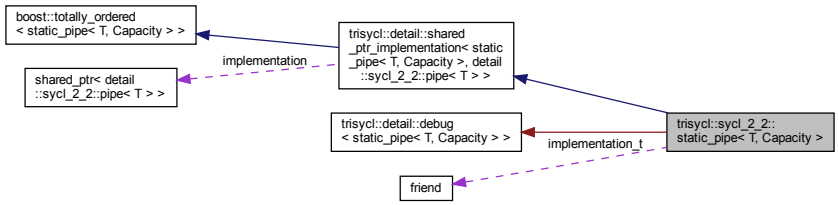
It is not directly mapped to the OpenCL program-scoped pipe because in SYCL there is not this concept of separated program. But the SYCL device compiler is expected to generate some OpenCL program(s) with program-scoped pipes when a SYCL static-scoped pipe is used. These details are implementation defined.

Definition at line 52 of file static_pipe.hpp.

Inheritance diagram for trisycl::sycl_2_2::static_pipe< T, Capacity >:



Collaboration diagram for trisycl::sycl_2_2::static_pipe< T, Capacity >:



Public Types

- using value_type = T

  The STL-like types.

Public Member Functions

- static_pipe ()

  Construct a static-scoped pipe able to store up to Capacity T objects.
- template<access::mode Mode, access::target Target = access::target::pipe>
  accessor< value_type, 1, Mode, Target > get_access (handler &command_group_handler) const

  Get an accessor to the pipe with the required mode.
- template<access::mode Mode, access::target Target = access::target::pipe>
  accessor< value_type, 1, Mode, Target > get_access () const

  Get an accessor to the pipe with the required mode, outside of a normal kernel, for example in host code.
- std::size_t constexpr capacity () const

  Return the maximum number of elements that can fit in the pipe.

Private Types

- using implementation_t = typename static_pipe::shared_ptr_implementation

Private Attributes

- friend implementation_t

Additional Inherited Members

9.9.2.10.1 Member Typedef Documentation

9.9.2.10.1.1 implementation_t `template<typename T , std::size_t Capacity>`

```
using trisycl::sycl_2_2::static_pipe< T, Capacity >::implementation_t = typename static_pipe::shared_ptr_impl
[private]
```

Definition at line 60 of file static_pipe.hpp.

9.9.2.10.1.2 value_type `template<typename T , std::size_t Capacity>`

`using trisycl::sycl_2_2::static_pipe< T, Capacity >::value_type = T`

The STL-like types.

Definition at line 71 of file static_pipe.hpp.

9.9.2.10.2 Constructor & Destructor Documentation

9.9.2.10.2.1 static_pipe() `template<typename T , std::size_t Capacity>`

`trisycl::sycl_2_2::static_pipe< T, Capacity >::static_pipe ( ) [inline]`

Construct a static-scoped pipe able to store up to Capacity T objects.

Definition at line 75 of file static_pipe.hpp.

```
00076     : implementation_t { new detail::sycl_2_2::pipe<T> { Capacity } } { }
```

9.9.2.10.3 Member Function Documentation

9.9.2.10.3.1 capacity() `template<typename T , std::size_t Capacity>`

`std::size_t constexpr trisycl::sycl_2_2::static_pipe< T, Capacity >::capacity ( ) const [inline], [constexpr]`

Return the maximum number of elements that can fit in the pipe.

This is a constexpr since the capacity is in the type.

Definition at line 123 of file static_pipe.hpp.

```
00123                                       {
00124     return Capacity;
00125     }
```

9.9.2.10.3.2 get_access() [1/2] `template<typename T , std::size_t Capacity>`

`template<access::mode Mode, access::target Target = access::target::pipe>`
`accessor< value_type, 1, Mode, Target > trisycl::sycl_2_2::static_pipe< T, Capacity >::get_access ( ) const [inline]`

Get an accessor to the pipe with the required mode, outside of a normal kernel, for example in host code.

Parameters

Mode	is the requested access mode
Target	is the type of pipe access required

Definition at line 110 of file static_pipe.hpp.

```
00110                                       {
00111     static_assert(Target == access::target::pipe
00112                   || Target == access::target::blocking_pipe,
00113                   "get_access(handler) with pipes can only deal with "
00114                   "access::pipe or access::blocking_pipe");
00115     return { implementation };
```

```
00116   }
```

References trisycl::access::blocking_pipe, and trisycl::access::pipe.

9.9.2.10.3.3 get_access() [2/2] template<typename T , std::size_t Capacity> template<access::mode Mode, access::target Target = access::target::pipe> accessor< value_type, 1, Mode, Target > trisycl::sycl_2_2::static_pipe< T, Capacity >::get_access (handler & command_group_handler) const [inline]

Get an accessor to the pipe with the required mode.

Parameters

	Mode	is the requested access mode
	Target	is the type of pipe access required
in	*command_group_handler*	is the command group handler in which the kernel is to be executed

Definition at line 91 of file static_pipe.hpp.

```
00091                                                {
00092     static_assert(Target == access::target::pipe
00093                   || Target == access::target::blocking_pipe,
00094                   "get_access(handler) with pipes can only deal with "
00095                   "access::pipe or access::blocking_pipe");
00096     return { implementation, command_group_handler };
00097   }
```

References trisycl::access::blocking_pipe, and trisycl::access::pipe.

9.9.2.10.4 Member Data Documentation

9.9.2.10.4.1 implementation_t template<typename T , std::size_t Capacity> friend trisycl::sycl_2_2::static_pipe< T, Capacity >::implementation_t [private]

Definition at line 66 of file static_pipe.hpp.

9.9.3 Function Documentation

9.9.3.1 get_pipe_detail()

```
template<typename Accessor >
static auto & trisycl::get_pipe_detail (
            Accessor & a ) [inline], [static]
```

`#include <include/triSYCL/accessor.hpp>`

Top-level function to break circular dependencies on the the types to get the pipe implementation.

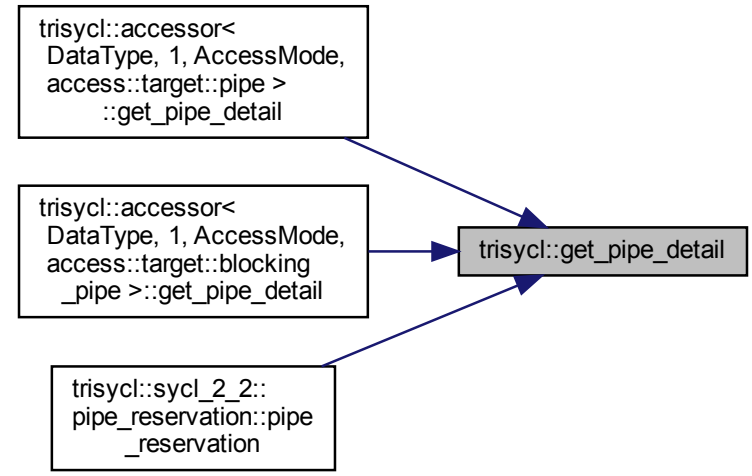
Definition at line 519 of file accessor.hpp.

```
00519                                                {
00520   return a.get_pipe_detail();
00521 }
```

References a.

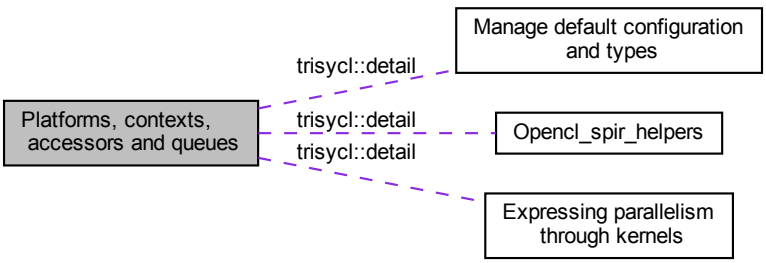
Referenced by trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >::get_pipe_detail(), trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >::get_pipe_detail(), and trisycl::sycl_2_2::pipe_reservation< PipeAccessor >::pipe_reservation().

Here is the caller graph for this function:



9.10 Platforms, contexts, accessors and queues

Collaboration diagram for Platforms, contexts, accessors and queues:



Namespaces

- namespace trisycl::detail

  Some global triSYCL configuration.
- namespace trisycl::info

Classes

- class trisycl::facade::accessor< AccessorMixIn >

  SYCL accessor facade providing multi-dimensional access features. More...
- class trisycl::mixin::accessor< T, Dimensions >

  SYCL accessor mixin providing multi-dimensional access features. More...
- class trisycl::context

  SYCL context. More...
- class trisycl::detail::context
- class trisycl::detail::host_context

  SYCL host context. More...
- class trisycl::device

  SYCL device. More...
- class trisycl::detail::device

  An abstract class representing various models of SYCL devices. More...
- class trisycl::facade::device< UserFacing, ImplementationDetail >

  SYCL device. More...
- class trisycl::device_selector

  The SYCL heuristics to select a device. More...
- class trisycl::device_type_selector

  A device selector by device_type. More...
- class trisycl::device_typename_selector< DeviceType >

  Select a device by template device_type parameter. More...
- class trisycl::handler

  Command group handler class. More...
- class trisycl::kernel

  SYCL kernel. More...
- class trisycl::detail::kernel

  Abstract SYCL kernel. More...
- class trisycl::platform

  Abstract the OpenCL platform. More...
- class trisycl::detail::host_platform

  SYCL host platform. More...
- class trisycl::detail::opencl_platform

  SYCL OpenCL platform. More...
- class trisycl::detail::platform

  An abstract class representing various models of SYCL platforms. More...
- class trisycl::queue

  SYCL queue, similar to the OpenCL queue concept. More...

Macros

- #define PLATFORM_GET_INFO_STRING(X)

Typedefs

- using trisycl::default_selector = device_typename_selector< info::device_type::automatic >

  Devices selected by heuristics of the system.
- using trisycl::gpu_selector = device_typename_selector< info::device_type::gpu >

  Select devices according to device type info::device::device_type::gpu from all the available OpenCL devices.
- using trisycl::cpu_selector = device_typename_selector< info::device_type::cpu >

  Select devices according to device type info::device::device_type::cpu from all the available devices and heuristics.
- using trisycl::host_selector = device_typename_selector< info::device_type::host >

  Selects the SYCL host CPU device that does not require an OpenCL runtime.
- using trisycl::info::device_fp_config = unsigned int
- using trisycl::info::device_exec_capabilities = unsigned int
- using trisycl::info::device_queue_properties = unsigned int

Enumerations

- enum class trisycl::info::context : int { trisycl::info::context::reference_count , trisycl::info::context::platform , trisycl::info::context::devices }

  Context information descriptors.
- enum class trisycl::info::device_type : unsigned int { trisycl::info::device_type::cpu , trisycl::info::device_type::gpu , trisycl::info::device_type::accelerator , trisycl::info::device_type::custom , trisycl::info::device_type::automatic , trisycl::info::device_type::host , trisycl::info::device_type::opencl , trisycl::info::device_type::all }

  Type of devices.
- enum class trisycl::info::device : int { trisycl::info::device::device_type , trisycl::info::device::vendor_id , trisycl::info::device::max_compute_units , trisycl::info::device::max_work_item_dimensions , trisycl::info::device::max_work_item_sizes , trisycl::info::device::max_work_group_size , trisycl::info::device::preferred_vector_w , trisycl::info::device::preferred_vector_width_short , trisycl::info::device::preferred_vector_width_int , trisycl::info::device::preferred_vector_width_long , trisycl::info::device::preferrec , trisycl::info::device::preferred_vector_width_double , trisycl::info::device::preferred_vector_width_half , trisycl::info::device::native_vector_width_char , trisycl::info::device::native_vec , trisycl::info::device::native_vector_width_int , trisycl::info::device::native_vector_width_long , trisycl::info::device::native_vector_width_float , trisycl::info::device::native_vector , trisycl::info::device::native_vector_width_half , trisycl::info::device::max_clock_frequency , trisycl::info::device::address_bits , trisycl::info::device::max_mem_alloc_size , trisycl::info::device::image_support , trisycl::info::device::max_read_image_args , trisycl::info::device::max_write_image_args , trisycl::info::device::image2d_max_wi , trisycl::info::device::image2d_max_height , trisycl::info::device::image3d_max_width , trisycl::info::device::image3d_max_height , trisycl::info::device::image3d_max_depth , trisycl::info::device::image_max_buffer_size , trisycl::info::device::image_max_array_size , trisycl::info::device::max_samplers , trisycl::info::device::max_parameter_size , trisycl::info::device::mem_base_addr_align , trisycl::info::device::half_fp_config , trisycl::info::device::single_fp_config , trisycl::info::device::double_fp_config , trisycl::info::device::global_mem_cache_type , trisycl::info::device::global_mem_cache_line_size , trisycl::info::device::global_mem_cache_size , trisycl::info::device::global_m , trisycl::info::device::max_constant_buffer_size , trisycl::info::device::max_constant_args , trisycl::info::device::local_mem_type , trisycl::info::device::local_mem_size , trisycl::info::device::error_correction_support , trisycl::info::device::host_unified_memory , trisycl::info::device::profiling_timer_resolution , trisycl::info::device::is_endian_little , trisycl::info::device::is_available , trisycl::info::device::is_compiler_available , trisycl::info::device::is_linker_available , trisycl::info::device::execution_capabilities , trisycl::info::device::queue_profiling , trisycl::info::device::built_in_kernels , trisycl::info::device::platform , trisycl::info::device::name , trisycl::info::device::vendor

,
trisycl::info::device::driver_version , trisycl::info::device::profile , trisycl::info::device::version , trisycl::info::device::opencl_c_versi
,
trisycl::info::device::extensions , trisycl::info::device::printf_buffer_size , trisycl::info::device::preferred_interop_user_sync
, trisycl::info::device::parent_device ,
trisycl::info::device::partition_max_sub_devices , trisycl::info::device::partition_properties , trisycl::info::device::partition_affinity_
, trisycl::info::device::partition_type_property ,
trisycl::info::device::partition_type_affinity_domain , trisycl::info::device::reference_count }

Device information descriptors.

- enum class trisycl::info::partition_property : int { trisycl::info::partition_property::no_partition , trisycl::info::partition_property::par
, trisycl::info::partition_property::partition_by_counts , trisycl::info::partition_property::partition_by_affinity_domain
}
- enum class trisycl::info::partition_affinity_domain : int {
trisycl::info::partition_affinity_domain::not_applicable , trisycl::info::partition_affinity_domain::numa ,
trisycl::info::partition_affinity_domain::L4_cache , trisycl::info::partition_affinity_domain::L3_cache ,
trisycl::info::partition_affinity_domain::L2_cache , trisycl::info::partition_affinity_domain::L1_cache , trisycl::info::partition_affinity
}
- enum class trisycl::info::local_mem_type : int { trisycl::info::local_mem_type::none , trisycl::info::local_mem_type::local
, trisycl::info::local_mem_type::global }
- enum class trisycl::info::fp_config : int {
trisycl::info::fp_config::denorm , trisycl::info::fp_config::inf_nan , trisycl::info::fp_config::round_to_nearest ,
trisycl::info::fp_config::round_to_zero ,
trisycl::info::fp_config::round_to_inf , trisycl::info::fp_config::fma , trisycl::info::fp_config::correctly_rounded_divide_sqrt
, trisycl::info::fp_config::soft_float }
- enum class trisycl::info::global_mem_cache_type : int { trisycl::info::global_mem_cache_type::none ,
trisycl::info::global_mem_cache_type::read_only , trisycl::info::global_mem_cache_type::write_only }
- enum class trisycl::info::execution_capability : unsigned int { trisycl::info::execution_capability::exec_kernel ,
trisycl::info::execution_capability::exec_native_kernel }
- enum class trisycl::info::platform : unsigned int {
trisycl::info::platform::TRISYCL_OPENCL_ONLY =(= CL_PLATFORM_PROFILE) , trisycl::info::platform::TRISYCL_OPENCL_C
=(= CL_PLATFORM_VERSION) , trisycl::info::platform::TRISYCL_OPENCL_ONLY =(= CL_PLATFORM_↩
NAME) , trisycl::info::platform::TRISYCL_OPENCL_ONLY =(= CL_PLATFORM_VENDOR) ,
trisycl::info::platform::TRISYCL_OPENCL_ONLY =(= CL_PLATFORM_EXTENSIONS) }

Platform information descriptors.

Functions

- template<> auto trisycl::context::get_info< info::context::reference_count > () const
- template<> auto trisycl::context::get_info< info::context::platform > () const
- template<> auto trisycl::context::get_info< info::context::devices > () const
- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image2d_max_width, size_t)

Query the return type for get_info() on context stuff.

- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image2d_max_height, size_t)
- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image3d_max_width, size_t)
- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image3d_max_height, size_t)
- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image3d_max_depth, size_t)
- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image_max_buffer_size, size_t)
- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image_max_array_size, size_t)
- template<> auto trisycl::platform::get_info< info::platform::extensions > () const
- template<> auto trisycl::queue::get_info< info::queue::device > () const
- template<> auto trisycl::queue::get_info< info::queue::context > () const
- template<> auto trisycl::queue::get_info< info::queue::reference_count > () const
- device trisycl::device_selector::select_device (vector_class< platform > platforms) const
- device trisycl::device_selector::select_device () const

Returns a selected device using the functor operator defined in sub-classes operator()(const device &dev)

- vector_class< trisycl::device > trisycl::detail::host_platform::get_devices (const device_selector &device_selector) const override

  Get all the available devices for the host platform.

- vector_class< trisycl::device > trisycl::detail::opencl_platform::get_devices (const device_selector &device_selector) const override

  Get all the available devices for this OpenCL platform.

Variables

- TRISYCL_WEAK_ATTRIB_PREFIX detail::cache< cl_context, detail::opencl_context > opencl_context::cache trisycl::detail::TRISYCL_WEAK_ATTRIB_SUFFIX

9.10.1 Detailed Description

9.10.2 Class Documentation

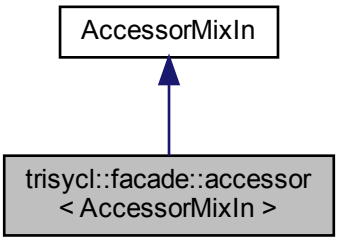
9.10.2.1 class trisycl::facade::accessor

template<typename AccessorMixIn>
class trisycl::facade::accessor< AccessorMixIn >

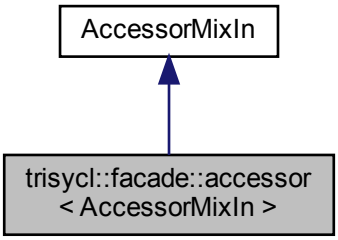
SYCL accessor facade providing multi-dimensional access features.

Definition at line 28 of file accessor.hpp.

Inheritance diagram for trisycl::facade::accessor< AccessorMixIn >:



Collaboration diagram for trisycl::facade::accessor< AccessorMixIn >:



Public Types

- using iterator = typename mixin::pointer
- using const_iterator = typename mixin::const_pointer
- using reverse_iterator = typename std::reverse_iterator< iterator >
- using const_reverse_iterator = typename std::reverse_iterator< const_iterator >

Public Member Functions

- template<std::integral... T>
  decltype(auto) operator[ ] (T... indices)

  Use the accessor with integers à la [i1][i2][i3] or C++23 [i1, i2,...].
- template<std::integral... T>
  decltype(auto) operator[ ] (T... indices) const

  Use the accessor with integers à la [i1][i2][i3] or C++23 [i1, i2,...].
- decltype(auto) operator[ ] (const id< mixin::rank()> &index)

  To use the accessor with [id<>].
- decltype(auto) operator[ ] (const id< mixin::rank()> &index) const

  To use the accessor with [id<>].
- decltype(auto) operator[ ] (const item< mixin::rank()> &index)

  To use an accessor with [item<>].
- decltype(auto) operator[ ] (const item< mixin::rank()> &index) const

  To use an accessor with [item<>].
- decltype(auto) operator[ ] (const nd_item< mixin::rank()> &index)

  To use an accessor with an [nd_item<>].
- decltype(auto) operator[ ] (const nd_item< mixin::rank()> &index) const

  To use an accessor with an [nd_item<>].
- mixin::reference operator* ()

  Get the first element of the accessor.
- mixin::reference operator* () const

  Get the first element of the accessor.
- auto get_pointer ()

  Return the pointer to the data.
- iterator begin ()

Forward all the iterator functions to the implementation.

- iterator end ()
- const_iterator cbegin ()
- const_iterator cend ()
- reverse_iterator rbegin ()
- reverse_iterator rend ()
- const_reverse_iterator crbegin () const
- const_reverse_iterator crend () const

Private Types

- using mixin = AccessorMixIn

9.10.2.1.1 Member Typedef Documentation

9.10.2.1.1.1 const_iterator `template<typename AccessorMixIn >`

```
using trisycl::facade::accessor< AccessorMixIn >::const_iterator = typename mixin::const_↩
pointer
```

Definition at line 34 of file accessor.hpp.

9.10.2.1.1.2 const_reverse_iterator `template<typename AccessorMixIn >`

```
using trisycl::facade::accessor< AccessorMixIn >::const_reverse_iterator = typename std↩
::reverse_iterator<const_iterator>
```

Definition at line 36 of file accessor.hpp.

9.10.2.1.1.3 iterator `template<typename AccessorMixIn >`

```
using trisycl::facade::accessor< AccessorMixIn >::iterator = typename mixin::pointer
```

Definition at line 33 of file accessor.hpp.

9.10.2.1.1.4 mixin `template<typename AccessorMixIn >`

```
using trisycl::facade::accessor< AccessorMixIn >::mixin = AccessorMixIn  [private]
```

Definition at line 30 of file accessor.hpp.

9.10.2.1.1.5 reverse_iterator `template<typename AccessorMixIn >`

```
using trisycl::facade::accessor< AccessorMixIn >::reverse_iterator = typename std::reverse_↩
iterator<iterator>
```

Definition at line 35 of file accessor.hpp.

9.10.2.1.2 Member Function Documentation

9.10.2.1.2.1 begin() `template<typename AccessorMixIn >`

`iterator trisycl::facade::accessor< AccessorMixIn >::begin ( ) [inline]`

Forward all the iterator functions to the implementation.

Todo Add these functions to the specification

Todo The fact that the lambda capture make a const copy of the accessor is not yet elegantly managed... The issue is that begin()/end() dispatch is made according to the accessor constness and not from the array member constness...

Todo try to solve it by using some enable_if on array constness?

Todo The issue is that the end may not be known if it is implemented by a raw OpenCL cl_mem... So only provide on the device the iterators related to the start? Actually the accessor needs to know a part of the shape to have the multidimentional addressing. So this only require a size_t more...

Todo Factor out these in a template helper

Todo Do we need this in detail::accessor too or only in accessor?

Definition at line 155 of file accessor.hpp.

```
00155 { return mixin::data(); }
```

Referenced by trisycl::facade::accessor< AccessorMixIn >::rend().

Here is the caller graph for this function:



9.10.2.1.2.2 cbegin() template<typename AccessorMixIn >

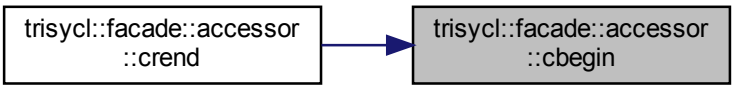
const_iterator trisycl::facade::accessor< AccessorMixIn >::cbegin ( ) [inline]

Definition at line 159 of file accessor.hpp.

```
00159 { return mixin::data(); }
```

Referenced by trisycl::facade::accessor< AccessorMixIn >::crend().

Here is the caller graph for this function:



9.10.2.1.2.3 cend() template<typename AccessorMixIn >

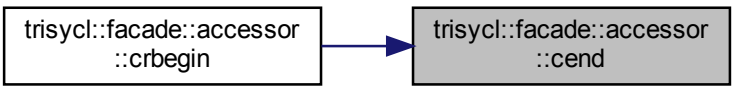
const_iterator trisycl::facade::accessor< AccessorMixIn >::cend ( ) [inline]

Definition at line 161 of file accessor.hpp.

```
00161 { return mixin::data() + mixin::get_count(); }
```

Referenced by trisycl::facade::accessor< AccessorMixIn >::crbegin().

Here is the caller graph for this function:



9.10.2.1.2.4 crbegin() template<typename AccessorMixIn >

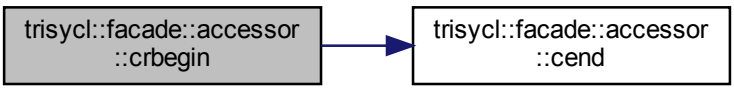
const_reverse_iterator trisycl::facade::accessor< AccessorMixIn >::crbegin ( ) const [inline]

Definition at line 170 of file accessor.hpp.

```
00170                                          {
00171     return std::reverse_iterator(cend());
00172   }
```

References trisycl::facade::accessor< AccessorMixIn >::cend().

Here is the call graph for this function:



9.10.2.1.2.5 crend() template<typename AccessorMixIn >

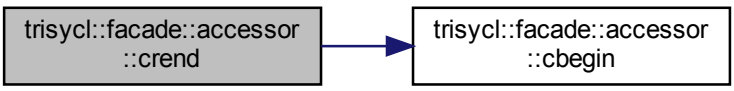
const_reverse_iterator trisycl::facade::accessor< AccessorMixIn >::crend ( ) const [inline]

Definition at line 174 of file accessor.hpp.

```
00174                                        {
00175     return std::reverse_iterator(cbegin());
00176   }
```

References trisycl::facade::accessor< AccessorMixIn >::cbegin().

Here is the call graph for this function:



9.10.2.1.2.6 end() template<typename AccessorMixIn >

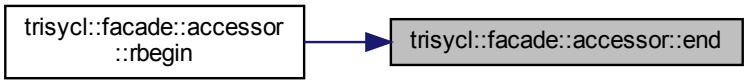
```
iterator trisycl::facade::accessor< AccessorMixIn >::end ( )  [inline]
```

Definition at line 157 of file accessor.hpp.

```
00157 { return mixin::data() + mixin::get_count(); }
```

Referenced by trisycl::facade::accessor< AccessorMixIn >::rbegin().

Here is the caller graph for this function:



9.10.2.1.2.7 get_pointer() template<typename AccessorMixIn >

```
auto trisycl::facade::accessor< AccessorMixIn >::get_pointer ( )  [inline]
```

Return the pointer to the data.

Todo Implement the various pointer address spaces

Definition at line 130 of file accessor.hpp.

```
00130 { return mixin::data(); }
```

9.10.2.1.2.8 operator*() [1/2] template<typename AccessorMixIn >

```
mixin::reference trisycl::facade::accessor< AccessorMixIn >::operator* ( )  [inline]
```

Get the first element of the accessor.

Useful with an accessor on a scalar for example.

Todo Add to the specification

Definition at line 112 of file accessor.hpp.

```
00112 { return *mixin::data(); }
```

9.10.2.1.2.9 operator*() [2/2] template<typename AccessorMixIn >

```
mixin::reference trisycl::facade::accessor< AccessorMixIn >::operator* ( ) const  [inline]
```

Get the first element of the accessor.

Useful with an accessor on a scalar for example.

Todo Add to the specification?

Todo Add the concept of 0-dim buffer and accessor for scalar and use an implicit conversion to value_type reference to access the value with the accessor?

Definition at line 124 of file accessor.hpp.

```
00124 { return *mixin::data(); }
```

9.10.2.1.2.10 operator[]() [1/8] template<typename AccessorMixIn >

```
decltype(auto) trisycl::facade::accessor< AccessorMixIn >::operator[] (
            const id< mixin::rank()> & index )  [inline]
```

To use the accessor with [id<>].

Definition at line 71 of file accessor.hpp.

```
00071                                                  {
00072     return mixin::tuple_indexed_access(index);
00073   }
```

9.10.2.1.2.11 operator[]() [2/8] template<typename AccessorMixIn >

```
decltype(auto) trisycl::facade::accessor< AccessorMixIn >::operator[] (
            const id< mixin::rank()> & index ) const  [inline]
```

To use the accessor with [id<>].

Definition at line 76 of file accessor.hpp.

```
00076                                                        {
00077     return mixin::tuple_indexed_access(index);
00078   }
```

9.10.2.1.2.12 operator[]() [3/8] template<typename AccessorMixIn >

```
decltype(auto) trisycl::facade::accessor< AccessorMixIn >::operator[] (
            const item< mixin::rank()> & index )  [inline]
```

To use an accessor with [item<>].

Definition at line 81 of file accessor.hpp.

```
00081                                                    {
00082     return (*this)[index.get()];
00083   }
```

9.10.2.1.2.13 operator[]() [4/8] template<typename AccessorMixIn >

```
decltype(auto) trisycl::facade::accessor< AccessorMixIn >::operator[] (
              const item< mixin::rank()> & index ) const  [inline]
```

To use an accessor with [item<>].

Definition at line 86 of file accessor.hpp.

```
00086                                                        {
00087     return (*this)[index.get()];
00088   }
```

9.10.2.1.2.14 operator[]() [5/8] template<typename AccessorMixIn >

```
decltype(auto) trisycl::facade::accessor< AccessorMixIn >::operator[] (
              const nd_item< mixin::rank()> & index )  [inline]
```

To use an accessor with an [nd_item<>].

Todo Add to the specification because used by HPC-GPU slide 22

Definition at line 94 of file accessor.hpp.

```
00094                                                           {
00095     return (*this)[index.get_global()];
00096   }
```

9.10.2.1.2.15 operator[]() [6/8] template<typename AccessorMixIn >

```
decltype(auto) trisycl::facade::accessor< AccessorMixIn >::operator[] (
              const nd_item< mixin::rank()> & index ) const  [inline]
```

To use an accessor with an [nd_item<>].

Todo Add to the specification because used by HPC-GPU slide 22

Definition at line 102 of file accessor.hpp.

```
00102                                                           {
00103     return (*this)[index.get_global()];
00104   }
```

9.10.2.1.2.16 operator[]() [7/8] template<typename AccessorMixIn >

```
template<std::integral...  T>
decltype(auto) trisycl::facade::accessor< AccessorMixIn >::operator[] (
              T...  indices )  [inline]
```

Use the accessor with integers à la [i1][i2][i3] or C++23 [i1, i2,...].

Returns

decltype(auto) to return either a reference to the final element when the indexing has been fully resolved or a proxy object to handle the remaining []

Definition at line 43 of file accessor.hpp.

```
00043                                                           {
00044     if constexpr (sizeof...(T) == 1)
00045       /* The [][][] case uses a proxy object to track all the [index]
00046 and aggregate them to resolve the indexing */
00047       return
00048           typename mixin::template track_index<1> { mixin::access }[indices...];
00049     else
00050       // Or just the C++23 [i1, i2,...]  case
00051       return mixin::access[indices...];
00052   }
```

9.10.2.1.2.17 operator[]() [8/8] template<typename AccessorMixIn > template<std::integral... T> decltype(auto) trisycl::facade::accessor< AccessorMixIn >::operator[] (T... *indices*) const [inline]

Use the accessor with integers à la [i1][i2][i3] or C++23 [i1, i2,...].

Returns

decltype(auto) to return either a reference to the final element when the indexing has been fully resolved or a proxy object to handle the remaining []

Definition at line 59 of file accessor.hpp.

```
00059                                                                                {
00060     if constexpr (sizeof...(T) == 1)
00061       /* The [][][] case uses a proxy object to track all the [index]
00062 and aggregate them to resolve the indexing */
00063       return
00064           typename mixin::template track_index<1> { mixin::access }[indices...];
00065     else
00066       // Or just the C++23 [i1, i2,...]  case
00067       return mixin::access[indices...];
00068   }
```

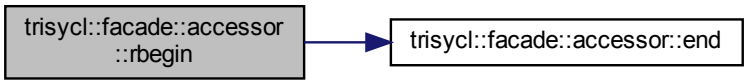
9.10.2.1.2.18 rbegin() template<typename AccessorMixIn > reverse_iterator trisycl::facade::accessor< AccessorMixIn >::rbegin ( ) [inline]

Definition at line 163 of file accessor.hpp.

```
00163 { return std::reverse_iterator(end()); }
```

References trisycl::facade::accessor< AccessorMixIn >::end().

Here is the call graph for this function:



9.10.2.1.2.19 rend() template<typename AccessorMixIn >

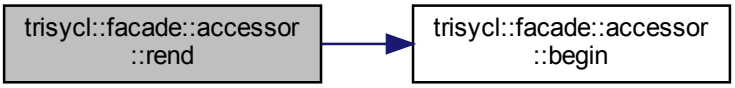
reverse_iterator trisycl::facade::accessor< AccessorMixIn >::rend ( ) [inline]

Definition at line 165 of file accessor.hpp.

```
00165                           {
00166     return std::reverse_iterator(begin());
00167     ;
00168   }
```

References trisycl::facade::accessor< AccessorMixIn >::begin().

Here is the call graph for this function:



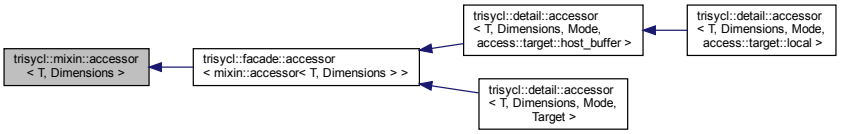
9.10.2.2 class trisycl::mixin::accessor

template<typename T, int Dimensions>
class trisycl::mixin::accessor< T, Dimensions >

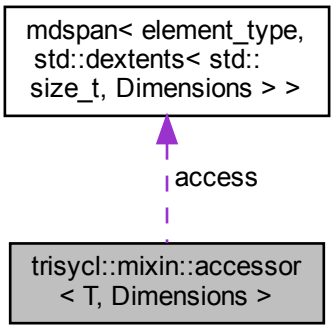
SYCL accessor mixin providing multi-dimensional access features.

Definition at line 26 of file accessor.hpp.

Inheritance diagram for trisycl::mixin::accessor< T, Dimensions >:



Collaboration diagram for trisycl::mixin::accessor< T, Dimensions >:



Public Types

- using element_type = T

  Extension to SYCL: provide pieces of STL container interface from mdspan.
- using value_type = std::remove_cv_t< element_type >

  Even if the buffer is read-only use a non-const type so at least the current implementation can copy the data too.
- using non_const_pointer = value_type *
- using pointer = typename mdspan::accessor_type::data_handle_type

  Pointer type to element.
- using const_pointer = const element_type *

  Pointer type to const element.
- using reference = typename mdspan::reference

  Reference type to the elements.

Public Member Functions

- accessor ()=default

  Used by the local accessor hack on top of host accessor.
- accessor (pointer data, const range< rank()> &r)

  Create an accessor of dimensions r on top of data storage.
- accessor (const mdspan &m)

  Create an accessor from another mdspan.
- void update (pointer data, const range< rank()> &r)

  Update the accessor to target somewhere else.
- auto get_range () const

  Return a range object representing the size of the buffer in terms of number of elements in each dimension as passed to the constructor.
- std::size_t get_count () const

  Returns the total number of elements in the buffer.
- std::size_t get_size () const

  Returns the size of the buffer storage in bytes.
- auto data ()

  Get the underlying storage.
- decltype(auto) tuple_indexed_access (const auto &tuple_like_indices)

  Access to an element with indices implementing a tuple interface.

Static Public Member Functions

- static auto constexpr rank ()

  Get the number of dimensions of the buffer.
- static decltype(auto) tuple_indexed_mdspan_access (auto &&some_mdspan, const auto &tuple_like_indices)

  Access to an mdspan element with indices implementing a tuple interface.

Protected Types

- using mdspan = std::mdspan< element_type, std::dextents< std::size_t, Dimensions > >

  The memory lay-out of a buffer is a dynamic multidimensional array.

Protected Member Functions

- template<typename BasicType , typename FinalType >
  const std::array< typename mdspan::size_type, rank()> & extents_cast (const detail::small_array< Basic↩
  Type, FinalType, rank()> &sa)

  Cast a SYCL range/id-like into a mdspan index array, which is an array of std::size_t into an array of std::ptrdiff_t.
- void set_access (const mdspan &a)

  Set later the mdspan associated to this accessor.

Protected Attributes

- mdspan access

  This is the multi-dimensional interface to the data that may point to either allocation in the case of storage managed by SYCL itself or to some other memory location in the case of host memory or storage<> abstraction use.

9.10.2.2.1 Member Typedef Documentation

9.10.2.2.1.1 const_pointer `template<typename T , int Dimensions>`

```
using trisycl::mixin::accessor< T, Dimensions >::const_pointer = const element_type*
```

Pointer type to const element.

Definition at line 75 of file accessor.hpp.

9.10.2.2.1.2 element_type `template<typename T , int Dimensions>`

```
using trisycl::mixin::accessor< T, Dimensions >::element_type = T
```

Extension to SYCL: provide pieces of STL container interface from mdspan.

Definition at line 31 of file accessor.hpp.

9.10.2.2.1.3 mdspan template<typename T , int Dimensions>

```
using trisycl::mixin::accessor< T, Dimensions >::mdspan = std::mdspan<element_type, std↩
::dextents<std::size_t, Dimensions> >  [protected]
```

The memory lay-out of a buffer is a dynamic multidimensional array.

Definition at line 47 of file accessor.hpp.

9.10.2.2.1.4 non_const_pointer template<typename T , int Dimensions>

```
using trisycl::mixin::accessor< T, Dimensions >::non_const_pointer = value_type*
```

Definition at line 37 of file accessor.hpp.

9.10.2.2.1.5 pointer template<typename T , int Dimensions>

```
using trisycl::mixin::accessor< T, Dimensions >::pointer = typename mdspan::accessor_type↩
::data_handle_type
```

Pointer type to element.

Definition at line 72 of file accessor.hpp.

9.10.2.2.1.6 reference template<typename T , int Dimensions>

```
using trisycl::mixin::accessor< T, Dimensions >::reference = typename mdspan::reference
```

Reference type to the elements.

Definition at line 78 of file accessor.hpp.

9.10.2.2.1.7 value_type template<typename T , int Dimensions>

```
using trisycl::mixin::accessor< T, Dimensions >::value_type = std::remove_cv_t<element_type>
```

Even if the buffer is read-only use a non-const type so at least the current implementation can copy the data too.

Definition at line 34 of file accessor.hpp.

9.10.2.2.2 Constructor & Destructor Documentation

9.10.2.2.2.1 accessor() [1/3] template<typename T , int Dimensions>

```
trisycl::mixin::accessor< T, Dimensions >::accessor ( )  [default]
```

Used by the local accessor hack on top of host accessor.

9.10.2.2.2 accessor() [2/3] template<typename T , int Dimensions>

```
trisycl::mixin::accessor< T, Dimensions >::accessor (
            pointer data,
            const range< rank()> & r )  [inline]
```

Create an accessor of dimensions r on top of data storage.

Definition at line 84 of file accessor.hpp.

```
00085       :  access { data, extents_cast(r) } {}
```

9.10.2.2.3 accessor() [3/3] template<typename T , int Dimensions>

```
trisycl::mixin::accessor< T, Dimensions >::accessor (
            const mdspan & m )  [inline]
```

Create an accessor from another mdspan.

Definition at line 88 of file accessor.hpp.

```
00089       :  access { m } {}
```

9.10.2.2.3 Member Function Documentation

9.10.2.2.3.1 data() template<typename T , int Dimensions>

```
auto trisycl::mixin::accessor< T, Dimensions >::data ( )  [inline]
```

Get the underlying storage.

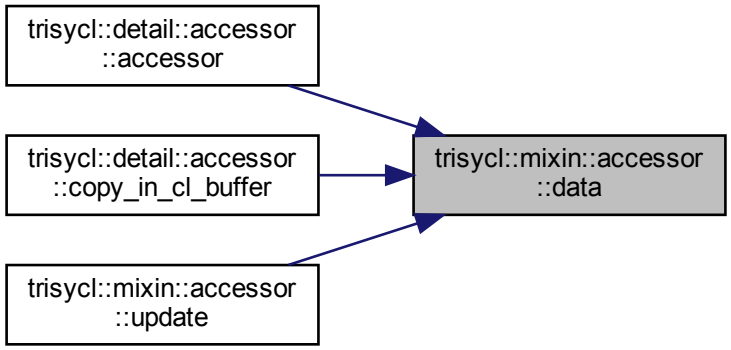
Definition at line 133 of file accessor.hpp.

```
00133 { return access.data_handle(); }
```

References trisycl::mixin::accessor< T, Dimensions >::access.

Referenced by trisycl::detail::accessor< T, Dimensions, Mode, Target >::accessor(), trisycl::detail::accessor< T, Dimensions, Mode, T
and trisycl::mixin::accessor< T, Dimensions >::update().

Here is the caller graph for this function:



9.10.2.2.3.2 extents_cast() template<typename T , int Dimensions>

```
template<typename BasicType , typename FinalType >
const std::array< typename mdspan::size_type, rank()> & trisycl::mixin::accessor< T, Dimensions
>::extents_cast (
            const detail::small_array< BasicType, FinalType, rank()> & sa )  [inline], [protected]
```

Cast a SYCL range/id-like into a mdspan index array, which is an array of std::size_t into an array of std::ptrdiff_t.

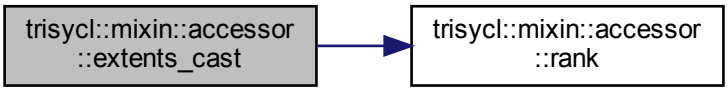
Definition at line 62 of file accessor.hpp.

```
00062                                                                    {
00063     return reinterpret_cast<
00064         const std::array<typename mdspan::size_type, rank()>&>(sa);
00065   }
```

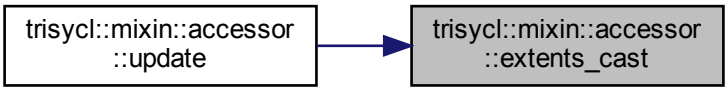
References trisycl::mixin::accessor< T, Dimensions >::rank().

Referenced by trisycl::mixin::accessor< T, Dimensions >::update().

Here is the call graph for this function:



Here is the caller graph for this function:



9.10.2.2.3.3 get_count() template<typename T , int Dimensions>

```
std::size_t trisycl::mixin::accessor< T, Dimensions >::get_count ( ) const  [inline]
```

Returns the total number of elements in the buffer.

Equal to get_range()[0] * ... * get_range()[Dimensions-1].

Todo Move these kinds of functions into a mixin between buffers and accessors?

Todo Cache it since it is const?

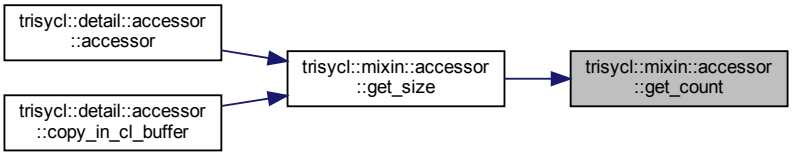
Definition at line 118 of file accessor.hpp.

```
00118                                {
00119     return access.mapping().required_span_size();
00120   }
```

References trisycl::mixin::accessor< T, Dimensions >::access.

Referenced by trisycl::mixin::accessor< T, Dimensions >::get_size().

Here is the caller graph for this function:



9.10.2.2.3.4 get_range() template<typename T , int Dimensions>

auto trisycl::mixin::accessor< T, Dimensions >::get_range ( ) const [inline]

Return a range object representing the size of the buffer in terms of number of elements in each dimension as passed to the constructor.

Todo Cache it since it is const?

Definition at line 102 of file accessor.hpp.

```
00102                                {
00103     range<Dimensions> r;
00104     for (std::size_t i = 0; i < Dimensions; ++i)
00105       r[i] = access.extent(i);
00106     return r;
00107   }
```

References trisycl::mixin::accessor< T, Dimensions >::access, and i.

9.10.2.2.3.5 get_size() template<typename T , int Dimensions>

```
std::size_t trisycl::mixin::accessor< T, Dimensions >::get_size ( ) const  [inline]
```

Returns the size of the buffer storage in bytes.

Todo rename to something else. In http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/p0122r0.pdf it is named bytes() for example

Todo Cache it since it is const?

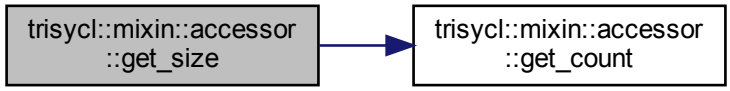
Definition at line 130 of file accessor.hpp.

```
00130 { return get_count() * sizeof(value_type); }
```

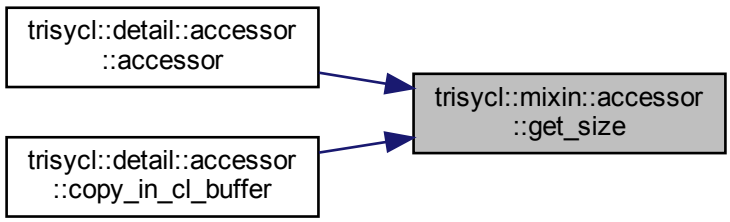
References trisycl::mixin::accessor< T, Dimensions >::get_count().

Referenced by trisycl::detail::accessor< T, Dimensions, Mode, Target >::accessor(), and trisycl::detail::accessor< T, Dimensions, Mo

Here is the call graph for this function:



Here is the caller graph for this function:



9.10.2.2.3.6 rank() template<typename T , int Dimensions>

```
static auto constexpr trisycl::mixin::accessor< T, Dimensions >::rank ( )  [inline], [static],
[constexpr]
```

Get the number of dimensions of the buffer.

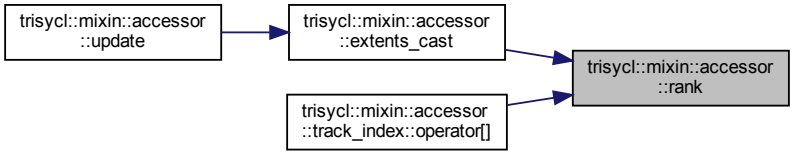
Name inspired from ISO C++ P0009 mdspan papers

Definition at line 43 of file accessor.hpp.

```
00043 { return Dimensions; }
```

Referenced by trisycl::mixin::accessor< T, Dimensions >::extents_cast(), and trisycl::mixin::accessor< T, Dimensions >::track_index

Here is the caller graph for this function:



9.10.2.2.3.7 set_access() template<typename T , int Dimensions>

```
void trisycl::mixin::accessor< T, Dimensions >::set_access (
              const mdspan & a )  [inline], [protected]
```

Set later the mdspan associated to this accessor.

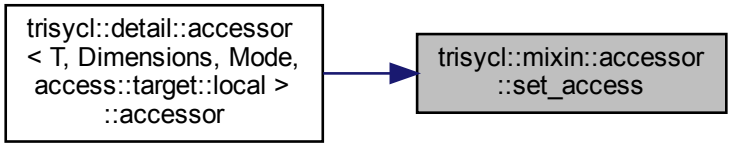
Definition at line 68 of file accessor.hpp.

```
00068 { access = a; }
```

References a, and trisycl::mixin::accessor< T, Dimensions >::access.

Referenced by trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >::accessor().

Here is the caller graph for this function:



9.10.2.2.3.8 tuple_indexed_access() template<typename T , int Dimensions>

```
decltype(auto) trisycl::mixin::accessor< T, Dimensions >::tuple_indexed_access (
              const auto & tuple_like_indices )  [inline]
```

Access to an element with indices implementing a tuple interface.

Parameters

in	*tuple_like_indices*	are the indices to use

Returns

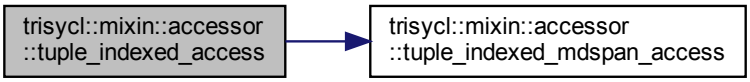
a reference to the accessor element

Definition at line 160 of file accessor.hpp.

```
00160                                                                    {
00161     return tuple_indexed_mdspan_access(access, tuple_like_indices);
00162   }
```

References trisycl::mixin::accessor< T, Dimensions >::access, and trisycl::mixin::accessor< T, Dimensions >::tuple_indexed_mdspa

Here is the call graph for this function:



9.10.2.2.3.9 tuple_indexed_mdspan_access()

```
template<typename T , int Dimensions>
static decltype(auto) trisycl::mixin::accessor< T, Dimensions >::tuple_indexed_mdspan_access (
              auto && some_mdspan,
              const auto & tuple_like_indices )  [inline], [static]
```

Access to an mdspan element with indices implementing a tuple interface.

Parameters

in,out	*some_mdspan*	is the mdspan to access
in	*tuple_like_indices*	are the indices to use

Returns

a reference to the element

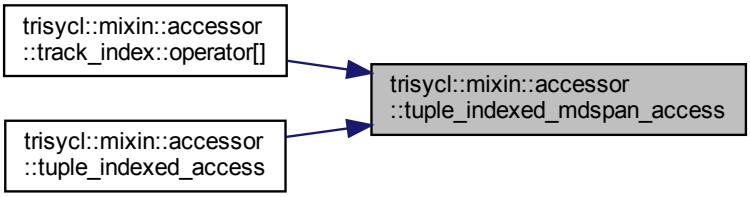
Definition at line 145 of file accessor.hpp.

```
00146                                                                    {
00147     // Otherwise use the C++23 mdspan[i1, i2,...]  indexing syntax
00148     return std::invoke(
00149         [&](auto...  i) -> decltype(auto) { return some_mdspan[i...]; },
00150         tuple_like_indices);
00151   }
```

References i.

Referenced by trisycl::mixin::accessor< T, Dimensions >::track_index< N >::operator[ ](), and trisycl::mixin::accessor< T, Dimensior

Here is the caller graph for this function:



9.10.2.2.3.10 update() template<typename T , int Dimensions>

```
void trisycl::mixin::accessor< T, Dimensions >::update (
            pointer data,
            const range< rank()> & r )  [inline]
```

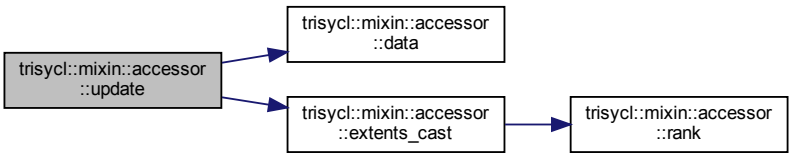
Update the accessor to target somewhere else.

Definition at line 92 of file accessor.hpp.

```
00092                                                {
00093     access = mdspan { data, extents_cast(r) };
00094   }
```

References trisycl::mixin::accessor< T, Dimensions >::access, trisycl::mixin::accessor< T, Dimensions >::data(), and trisycl::mixin::accessor< T, Dimensions >::extents_cast().

Here is the call graph for this function:



9.10.2.2.4 Member Data Documentation

9.10.2.2.4.1 access template<typename T , int Dimensions>

```
mdspan trisycl::mixin::accessor< T, Dimensions >::access  [protected]
```

This is the multi-dimensional interface to the data that may point to either allocation in the case of storage managed by SYCL itself or to some other memory location in the case of host memory or storage<> abstraction use.

Definition at line 55 of file accessor.hpp.

Referenced by trisycl::mixin::accessor< T, Dimensions >::data(), trisycl::mixin::accessor< T, Dimensions >::get_count(), trisycl::mixin::accessor< T, Dimensions >::get_range(), trisycl::mixin::accessor< T, Dimensions >::set_access(), trisycl::mixin::accessor< T, Dimensions >::tuple_indexed_access(), and trisycl::mixin::accessor< T, Dimensions >::update().

9.10.2.3 class trisycl::context

SYCL context.

The context class encapsulates an OpenCL context, which is implicitly created and the lifetime of the context instance defines the lifetime of the underlying OpenCL context instance.

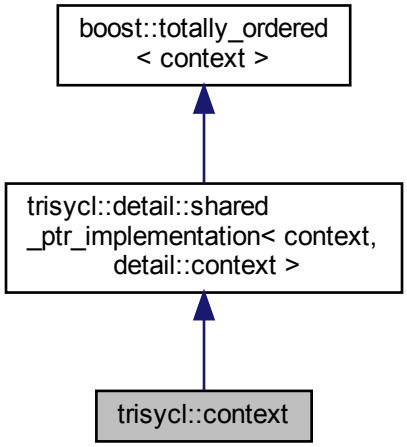
On destruction clReleaseContext is called.

The default context is the SYCL host context containing only the SYCL host device.

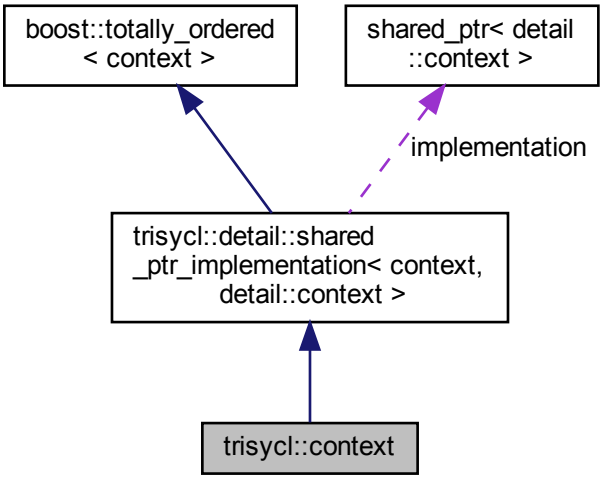
Todo The implementation is quite minimal for now.

Definition at line 46 of file context.hpp.

Inheritance diagram for trisycl::context:



Collaboration diagram for trisycl::context:



Public Member Functions

- context (async_handler asyncHandler)

  Constructs a context object for SYCL host using an async_handler for handling asynchronous errors.
- context (cl_context clContext, async_handler asyncHandler=nullptr)

  Make a SYCL context from an OpenCL context.
- context (const boost::compute::context &c, async_handler asyncHandler=nullptr)

  Build a SYCL context from a Boost.Compute context.
- context (const device_selector &deviceSelector, async_handler asyncHandler=nullptr)

  Constructs a context object using a device_selector object.
- context (const device &dev, async_handler asyncHandler=nullptr)

  Constructs a context object using a device object.
- context (const platform &plt, async_handler asyncHandler=nullptr)

  Constructs a context object using a platform object.
- context (const vector_class< device > &deviceList, async_handler asyncHandler=nullptr)
- context ()

  Default constructor that chooses the context according the heuristics of the default selector.
- cl_context get () const

  Return the underlying `cl_context` object, after retaining the `cl_context`.
- boost::compute::context & get_boost_compute () const

  Return the underlying `boost::compute::context` of the `trisycl::context`.
- boost::compute::command_queue & get_boost_queue () const

  Return the internal queue that is associated to the context and used by triSYCL to move data between some different contests for example.
- bool is_host () const

  Specifies whether the context is in SYCL Host Execution Mode.
- platform get_platform () const

  Returns the SYCL platform that the context is initialized for.

- vector_class< device > get_devices () const

  Returns the set of devices that are part of this context.

- template<info::context param>
  auto get_info () const

  Queries OpenCL information for the under-lying cl context.

Private Types

- using implementation_t = detail::shared_ptr_implementation< context, detail::context >

Private Member Functions

- context (vector_class< device > devList, async_handler asyncHandler, const device_selector &deviceSelector)

Additional Inherited Members

9.10.2.3.1 Member Typedef Documentation

9.10.2.3.1.1 implementation_t `using trisycl::context::implementation_t = detail::shared_ptr_implementation<cont detail::context> [private]`

Definition at line 53 of file context.hpp.

9.10.2.3.2 Constructor & Destructor Documentation

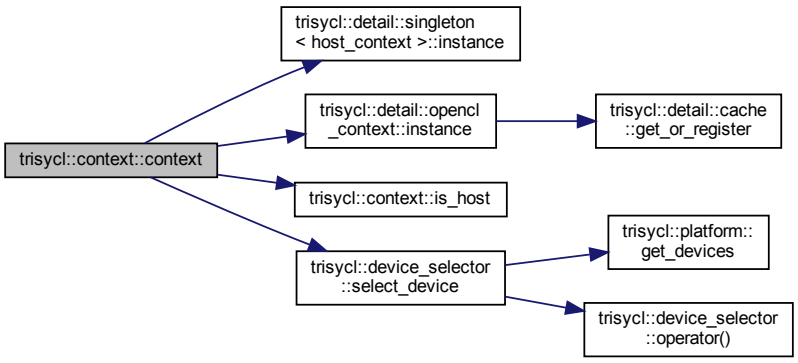
9.10.2.3.2.1 context() [1/9] `trisycl::context::context ( vector_class< device > devList, async_handler asyncHandler, const device_selector & deviceSelector ) [inline], [private]`

Definition at line 56 of file context.hpp.

```
00057                                                {
00058     if (devList.empty())
00059       devList = {deviceSelector.select_device()};
00060     if (devList[0].is_host())
00061       implementation = detail::host_context::instance();
00062 #ifdef TRISYCL_OPENCL
00063     else
00064       implementation = detail::opencl_context::instance(
00065         boost::compute::context { devList[0].get_boost_compute () }
00066       );
00067 #endif
00068   }
```

References trisycl::detail::shared_ptr_implementation< context, detail::context >::implementation, trisycl::detail::singleton< host_cor trisycl::detail::opencl_context::instance(), is_host(), and trisycl::device_selector::select_device().

Here is the call graph for this function:



9.10.2.3.2.2 context() [2/9] trisycl::context::context (async_handler *asyncHandler*) [inline], [explicit]

Constructs a context object for SYCL host using an async_handler for handling asynchronous errors.

Note that the default case asyncHandler = nullptr is handled by the default constructor.

Definition at line 81 of file context.hpp.

```
00082     : context { device{}, asyncHandler } {}
```

9.10.2.3.2.3 context() [3/9] trisycl::context::context (cl_context *clContext*, async_handler *asyncHandler* = *nullptr*) [inline]

Make a SYCL context from an OpenCL context.

The constructor executes a retain on the `cl_context`.

Return synchronous errors via the SYCL exception class and asynchronous errors are handled via the `async_handler`, if provided.

Definition at line 93 of file context.hpp.

```
00094     : context { boost::compute::context { clContext }, asyncHandler } {}
```

9.10.2.3.2.4 context() [4/9] trisycl::context::context (const boost::compute::context & *c*, async_handler *asyncHandler* = *nullptr*) [inline]

Build a SYCL context from a Boost.Compute context.

Definition at line 98 of file context.hpp.

```
00100     : implementation_t { detail::opencl_context::instance(c) } {}
```

9.10.2.3.2.5 context() [5/9] trisycl::context::context (const device_selector & deviceSelector, async_handler asyncHandler = nullptr) [inline]

Constructs a context object using a device_selector object.

The context is constructed with a single device retrieved from the device_selector object provided.

Return synchronous errors via the SYCL exception class and asynchronous errors are handled via the async_↩ handler, if provided.

Definition at line 111 of file context.hpp.

```
00112     : context (vector_class<device>{}, asyncHandler, deviceSelector) {}
```

9.10.2.3.2.6 context() [6/9] trisycl::context::context (const device & dev, async_handler asyncHandler = nullptr) [inline]

Constructs a context object using a device object.

Return synchronous errors via the SYCL exception class and asynchronous errors are handled via the async_↩ handler, if provided.

Definition at line 119 of file context.hpp.

```
00120     : context { vector_class<device>{dev}, asyncHandler } {}
```

9.10.2.3.2.7 context() [7/9] trisycl::context::context (const platform & plt, async_handler asyncHandler = nullptr) [inline]

Constructs a context object using a platform object.

Return synchronous errors via the SYCL exception class and asynchronous errors are handled via the async_↩ handler, if provided.

Definition at line 127 of file context.hpp.

```
00128     : context { plt.get_devices(), asyncHandler } {}
```

9.10.2.3.2.8 context() [8/9] trisycl::context::context (const vector_class< device > & deviceList, async_handler asyncHandler = nullptr)

9.10.2.3.2.9 context() [9/9] trisycl::context::context ( ) [inline]

Default constructor that chooses the context according the heuristics of the default selector.

Return synchronous errors via the SYCL exception class.

Get the default constructors back.

Definition at line 148 of file context.hpp.

```
00148 : context { async_handler {} } {}
```

9.10.2.3.3 Member Function Documentation

9.10.2.3.3.1 get() `cl_context trisycl::context::get ( ) const [inline]`

Return the underlying `cl_context` object, after retaining the `cl_context`.

Retains a reference to the returned `cl_context` object.

Caller should release it when finished.

Definition at line 158 of file context.hpp.

```
00158                                {
00159     return implementation->get();
00160   }
```

9.10.2.3.3.2 get_boost_compute() `boost::compute::context & trisycl::context::get_boost_compute ( ) const [inline]`

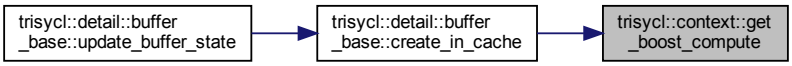
Return the underlying `boost::compute::context` of the `trisycl::context`.

Definition at line 165 of file context.hpp.

```
00165                                                {
00166     return implementation->get_boost_compute();
00167   }
```

Referenced by trisycl::detail::buffer_base::create_in_cache().

Here is the caller graph for this function:



9.10.2.3.3.3 get_boost_queue() `boost::compute::command_queue & trisycl::context::get_boost_↩ queue ( ) const [inline]`

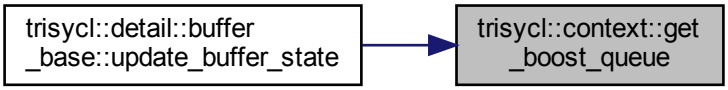
Return the internal queue that is associated to the context and used by triSYCL to move data between some different contests for example.

Definition at line 173 of file context.hpp.

```
00173                                                    {
00174     return implementation->get_boost_queue();
00175   }
```

Referenced by trisycl::detail::buffer_base::update_buffer_state().

Here is the caller graph for this function:



9.10.2.3.3.4 get_devices() vector_class< device > trisycl::context::get_devices ( ) const [inline]

Returns the set of devices that are part of this context.

Todo To be implemented

Definition at line 198 of file context.hpp.

```
00198                                          {
00199     return implementation->get_devices();
00200   }
```

9.10.2.3.3.5 get_info() template<info::context param>

```
auto trisycl::context::get_info ( ) const  [inline]
```

Queries OpenCL information for the under-lying cl context.

Todo To be implemented

9.10.2.3.3.6 get_platform() platform trisycl::context::get_platform ( ) const [inline]

Returns the SYCL platform that the context is initialized for.

Todo To be implemented

Definition at line 189 of file context.hpp.

```
00189                                    {
00190     return implementation->get_platform();
00191   };
```

9.10.2.3.3.7 is_host() `bool trisycl::context::is_host ( ) const [inline]`

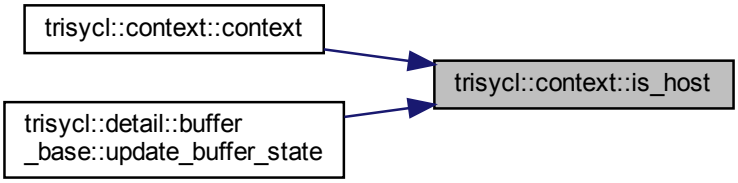
Specifies whether the context is in SYCL Host Execution Mode.

Definition at line 180 of file context.hpp.

```
00180                          {
00181     return implementation->is_host();
00182   }
```

Referenced by context(), and trisycl::detail::buffer_base::update_buffer_state().

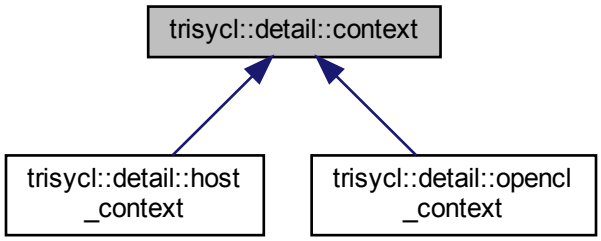
Here is the caller graph for this function:



9.10.2.4 class trisycl::detail::context

Definition at line 22 of file context.hpp.

Inheritance diagram for trisycl::detail::context:



Public Member Functions

- virtual cl_context get () const =0

  Return the underlying `cl_context` of the `trisycl::context`.
- virtual boost::compute::context & get_boost_compute ()=0

  Return the underlying `boost::compute::context` of the `trisycl::context`.

- virtual boost::compute::command_queue & get_boost_queue ()=0

  Return the underlying `boost::compute::command_queue` *associated with the context.*
- virtual bool is_host () const =0

  Returns true is the context is a SYCL host context.
- virtual trisycl::platform get_platform () const =0

  Returns the SYCL platform that the context is initialized for.
- virtual vector_class< trisycl::device > get_devices () const =0

  Returns the set of devices that are part of this context.
- virtual trisycl::cl_uint get_reference_count () const =0

  Returns the reference count of underlying CL context, or 0 on host.
- virtual ~context ()

  Virtual to call the real destructor.

9.10.2.4.1 Constructor & Destructor Documentation

9.10.2.4.1.1 ~context() `virtual trisycl::detail::context::~context ( ) [inline], [virtual]`

Virtual to call the real destructor.

Definition at line 59 of file context.hpp.

```
00059 {}
```

9.10.2.4.2 Member Function Documentation

9.10.2.4.2.1 get() `virtual cl_context trisycl::detail::context::get ( ) const [pure virtual]`

Return the underlying `cl_context` of the `trisycl::context`.

Implemented in trisycl::detail::host_context, and trisycl::detail::opencl_context.

9.10.2.4.2.2 get_boost_compute() `virtual boost::compute::context & trisycl::detail::context::get_boost_compute ( ) [pure virtual]`

Return the underlying `boost::compute::context` of the `trisycl::context`.

Implemented in trisycl::detail::host_context, and trisycl::detail::opencl_context.

9.10.2.4.2.3 get_boost_queue() `virtual boost::compute::command_queue & trisycl::detail::context::get_boost_queue ( ) [pure virtual]`

Return the underlying `boost::compute::command_queue` associated with the context.

Implemented in trisycl::detail::host_context, and trisycl::detail::opencl_context.

9.10.2.4.2.4 get_devices() virtual vector_class< trisycl::device > trisycl::detail::context::get_devices ( ) const [pure virtual]

Returns the set of devices that are part of this context.

Todo virtual cannot be templated template <info::context Param> typename info::param_traits<info::context, Param>::type get_info() const = 0;

Implemented in trisycl::detail::host_context, and trisycl::detail::opencl_context.

9.10.2.4.2.5 get_platform() virtual trisycl::platform trisycl::detail::context::get_platform ( ) const [pure virtual]

Returns the SYCL platform that the context is initialized for.

Implemented in trisycl::detail::host_context, and trisycl::detail::opencl_context.

9.10.2.4.2.6 get_reference_count() virtual trisycl::cl_uint trisycl::detail::context::get_reference_count ( ) const [pure virtual]

Returns the reference count of underlying CL context, or 0 on host.

Implemented in trisycl::detail::host_context, and trisycl::detail::opencl_context.

9.10.2.4.2.7 is_host() virtual bool trisycl::detail::context::is_host ( ) const [pure virtual]

Returns true is the context is a SYCL host context.

Implemented in trisycl::detail::host_context, and trisycl::detail::opencl_context.

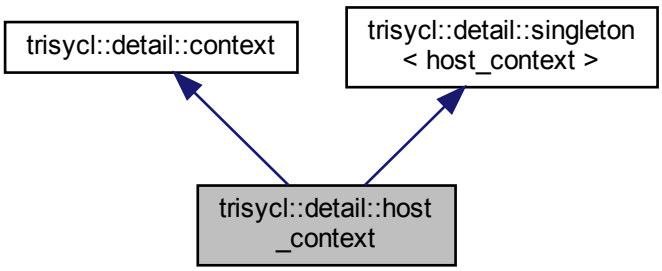
9.10.2.5 class trisycl::detail::host_context

SYCL host context.

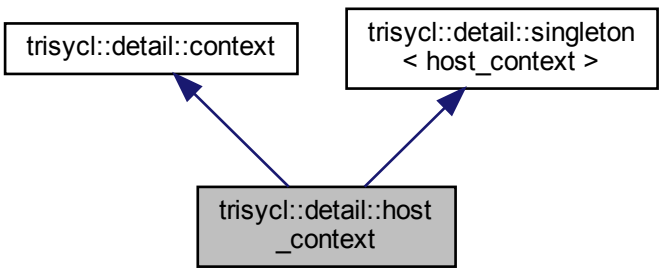
Todo The implementation is quite minimal for now. :-)

Definition at line 31 of file host_context.hpp.

Inheritance diagram for trisycl::detail::host_context:



Collaboration diagram for trisycl::detail::host_context:



Public Member Functions

- cl_context get () const override

  Return the underlying `cl_context` of the `trisycl::context`.
- boost::compute::context & get_boost_compute () override

  Return the SYCL platform that the context is initialized for.
- boost::compute::command_queue & get_boost_queue () override

  Return the internal OpenCL queue that is associated to the host context.

- bool is_host () const override

  Return true since the context is a SYCL host context.
- trisycl::platform get_platform () const override

  Return the platform of the context.
- vector_class< trisycl::device > get_devices () const override

  Returns the set of devices that are part of this context.
- trisycl::cl_uint get_reference_count () const override

  Return 0 since the context is a SYCL host context.

Additional Inherited Members

9.10.2.5.1 Member Function Documentation

9.10.2.5.1.1 get() `cl_context trisycl::detail::host_context::get ( ) const [inline], [override], [virtual]`

Return the underlying `cl_context` of the `trisycl::context`.

This throws an error since there is no OpenCL context associated to the host device.

Implements trisycl::detail::context.

Definition at line 42 of file host_context.hpp.

```
00042                                  {
00043     throw non_cl_error("The host context has no OpenCL context");
00044   }
```

9.10.2.5.1.2 get_boost_compute() `boost::compute::context & trisycl::detail::host_context::get↩_boost_compute ( ) [inline], [override], [virtual]`

Return the SYCL platform that the context is initialized for.

This throws an error since there is no `boost::compute` context associated to the host device.

Implements trisycl::detail::context.

Definition at line 52 of file host_context.hpp.

```
00052                                                  {
00053     throw non_cl_error("The host context has no OpenCL context");
00054   }
```

9.10.2.5.1.3 get_boost_queue() boost::compute::command_queue & trisycl::detail::host_context::get_boost_queue ( ) [inline], [override], [virtual]

Return the internal OpenCL queue that is associated to the host context.

This throws an error since there is no boost::compute::command_queue associated to the host context.

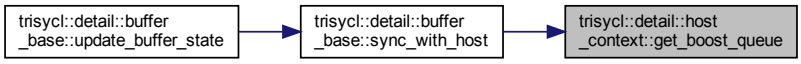
Implements trisycl::detail::context.

Definition at line 63 of file host_context.hpp.

```
00063                                                 {
00064     throw non_cl_error("The host context cannot have an OpenCL queue");
00065   }
```

Referenced by trisycl::detail::buffer_base::sync_with_host().

Here is the caller graph for this function:



9.10.2.5.1.4 get_devices() vector_class< trisycl::device > trisycl::detail::host_context::get_devices ( ) const [inline], [override], [virtual]

Returns the set of devices that are part of this context.

It should only return the host device itself.

Todo To be implemented

Implements trisycl::detail::context.

Definition at line 106 of file host_context.hpp.

```
00106                                       {
00107     // Return just the host device
00108     return { {} };
00109   }
```

9.10.2.5.1.5 get_platform() trisycl::platform trisycl::detail::host_context::get_platform ( ) const [inline], [override], [virtual]

Return the platform of the context.

Return synchronous errors via the SYCL exception class.

Implements trisycl::detail::context.

Definition at line 79 of file host_context.hpp.

```
00079                                           {
00080     // Return the host platform
00081     return {};
00082   }
```

9.10.2.5.1.6 get_reference_count() trisycl::cl_uint trisycl::detail::host_context::get_reference↩ _count ( ) const [inline], [override], [virtual]

Return 0 since the context is a SYCL host context.

Implements trisycl::detail::context.

Definition at line 112 of file host_context.hpp.

```
00112                                                  {
00113     return 0;
00114   }
```

9.10.2.5.1.7 is_host() bool trisycl::detail::host_context::is_host ( ) const [inline], [override], [virtual]

Return true since the context is a SYCL host context.

Implements trisycl::detail::context.

Definition at line 70 of file host_context.hpp.

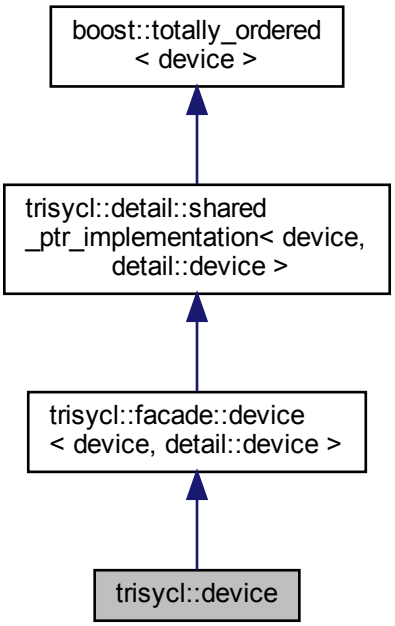
```
00070                                  {
00071     return true;
00072   }
```

9.10.2.6 class trisycl::device

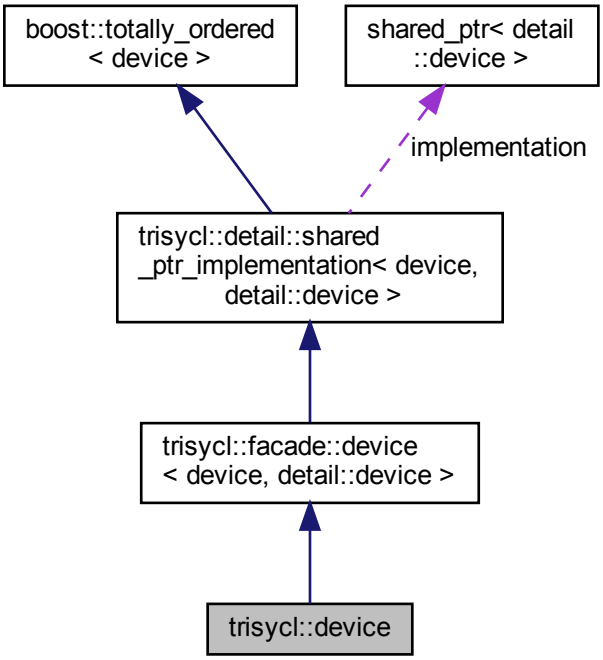
SYCL device.

Definition at line 44 of file device.hpp.

Inheritance diagram for trisycl::device:



Collaboration diagram for trisycl::device:



Public Member Functions

- device ()

  The default constructor uses the SYCL host device.
- device (cl_device_id device_id)

  Construct a device class instance using cl_device_id of the OpenCL device.
- device (const boost::compute::device &d)

  Construct a device class instance using a boost::compute::device.
- device (const device_selector &ds)

  Construct a device class instance using the device selector provided.
- cl_device_id get () const

  Return the cl_device_id of the underlying OpenCL platform.
- boost::compute::device get_boost_compute () const

  Return the underlying Boost.Compute device if it is an OpenCL device.
- bool is_host () const

  Return true if the device is the SYCL host device.
- bool is_cpu () const

  Return true if the device is an OpenCL CPU device.
- bool is_gpu () const

  Return true if the device is an OpenCL GPU device.
- bool is_accelerator () const

  Return true if the device is an OpenCL accelerator device.
- info::device_type type () const

Return the device_type of a device.

- platform get_platform () const

  Return the platform of device.

- template<info::device param>
  info::param_traits< info::device, param >::return_type get_info () const

  Query the device for OpenCL info::device info.

- bool has_extension (const string_class &extension) const

  Test if a specific extension is supported on the device.

- template<info::partition_property prop>
  vector_class< device > create_sub_devices (size_t nbSubDev) const
- template<info::partition_property prop>
  vector_class< device > create_sub_devices (const vector_class< size_t > &counts) const
- template<info::partition_property prop>
  vector_class< device > create_sub_devices (info::partition_affinity_domain affinityDomain) const

Static Public Member Functions

- static vector_class< device > get_devices (info::device_type device_type=info::device_type::all) TRISYCL_WEAK_ATTRIB_SU

  Return a list of all available devices.

Private Types

- using implementation_t = facade::device< device, detail::device >

Additional Inherited Members

9.10.2.6.1 Member Typedef Documentation

9.10.2.6.1.1 implementation_t `using trisycl::device::implementation_t = facade::device<device, detail::device> [private]`

Definition at line 50 of file device.hpp.

9.10.2.6.2 Constructor & Destructor Documentation

9.10.2.6.2.1 device() [1/4] `trisycl::device::device ( ) [inline]`

The default constructor uses the SYCL host device.

Definition at line 58 of file device.hpp.

```
00058 : implementation_t { detail::host_device::instance() } {}
```

9.10.2.6.2.2 device() [2/4] trisycl::device::device (cl_device_id *device_id*) [inline]

Construct a device class instance using cl_device_id of the OpenCL device.

Return synchronous errors via the SYCL exception class.

Retain a reference to the OpenCL device and if this device was an OpenCL subdevice the device should be released by the caller when it is no longer needed.

Definition at line 71 of file device.hpp.

```
00072     : device { boost::compute::device { device_id } } {}
```

9.10.2.6.2.3 device() [3/4] trisycl::device::device (const boost::compute::device & *d*) [inline]

Construct a device class instance using a boost::compute::device.

This is a triSYCL extension for boost::compute interoperation.

Return synchronous errors via the SYCL exception class.

Definition at line 81 of file device.hpp.

```
00082     : implementation_t { detail::opencl_device::instance(d) } {}
```

9.10.2.6.2.4 device() [4/4] trisycl::device::device (const device_selector & *ds*) [inline], [explicit]

Construct a device class instance using the device selector provided.

Return errors via C++ exception class.

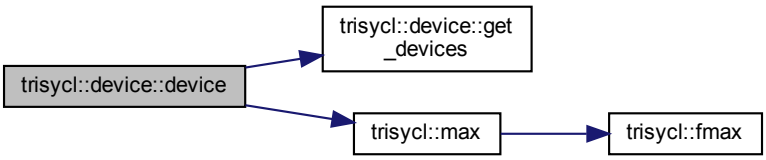
Todo Make it non-explicit in the specification?

Definition at line 93 of file device.hpp.

```
00093                                                {
00094     auto devices = device::get_devices();
00095     if (devices.empty())
00096       // \todo Put a SYCL exception
00097       throw std::domain_error("No device at all!  Internal error...");
00098
00099     /* Find the device with the best score according to the given
00100 device_selector */
00101     auto max = std::max_element(devices.cbegin(), devices.cend(),
00102                                 [&] (const device &d1, const device &d2) {
00103                                   return ds(d1) < ds(d2);
00104                                 });
00105     if (ds(*max) < 0)
00106       // \todo Put a SYCL exception
00107       throw std::domain_error("No device selected because no positive "
00108                               "device_selector score found");
00109
00110     // Create the current device as a shared copy of the selected one
00111     implementation = max->implementation;
00112   }
```

References get_devices(), trisycl::detail::shared_ptr_implementation< device, detail::device >::implementation, and trisycl::max().

Here is the call graph for this function:



9.10.2.6.3 Member Function Documentation

9.10.2.6.3.1 create_sub_devices() [1/3] template<info::partition_property prop>

```
vector_class< device > trisycl::device::create_sub_devices (
            const vector_class< size_t > & counts ) const  [inline]
```

Definition at line 240 of file device.hpp.

```
00241   {
00242     throw trisycl::feature_not_supported("unsupported feature\n");
00243   }
```

9.10.2.6.3.2 create_sub_devices() [2/3] template<info::partition_property prop>

```
vector_class< device > trisycl::device::create_sub_devices (
            info::partition_affinity_domain affinityDomain ) const  [inline]
```

Definition at line 247 of file device.hpp.

```
00248   {
00249     throw trisycl::feature_not_supported("unsupported feature\n");
00250   }
```

9.10.2.6.3.3 create_sub_devices() [3/3] template<info::partition_property prop>

```
vector_class< device > trisycl::device::create_sub_devices (
            size_t nbSubDev ) const  [inline]
```

Definition at line 233 of file device.hpp.

```
00234   {
00235     throw trisycl::feature_not_supported("unsupported feature\n");
00236   }
```

9.10.2.6.3.4 get() `cl_device_id trisycl::device::get ( ) const [inline]`

Return the cl_device_id of the underlying OpenCL platform.

Return synchronous errors via the SYCL exception class.

Retain a reference to the returned cl_device_id object. Caller should release it when finished.

In the case where this is the SYCL host device it will throw an exception.

Definition at line 126 of file device.hpp.

```
00126                                {
00127     return implementation->get();
00128   }
```

References trisycl::detail::shared_ptr_implementation< device, detail::device >::implementation.

9.10.2.6.3.5 get_boost_compute() `boost::compute::device trisycl::device::get_boost_compute ( ) const [inline]`

Return the underlying Boost.Compute device if it is an OpenCL device.

This is a triSYCL extension

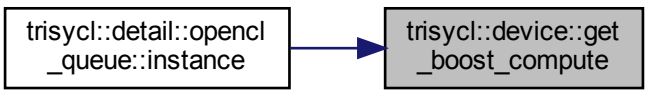
Definition at line 136 of file device.hpp.

```
00136                                                {
00137     return implementation->get_boost_compute();
00138   }
```

References trisycl::detail::shared_ptr_implementation< device, detail::device >::implementation.

Referenced by trisycl::detail::opencl_queue::instance().

Here is the caller graph for this function:



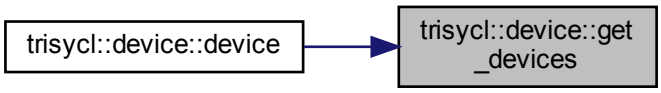
9.10.2.6.3.6 get_devices() static vector_class< device > trisycl::device::get_devices (info::device_type device_type = info::device_type::all) [static]

Return a list of all available devices.

Return synchronous errors via SYCL exception classes.

Referenced by device().

Here is the caller graph for this function:



9.10.2.6.3.7 get_info() template<info::device param> info::param_traits< info::device, param >::return_type trisycl::device::get_info ( ) const [inline]

Query the device for OpenCL info::device info.

Return synchronous errors via the SYCL exception class.

Todo

Query the device for OpenCL info::device info

Return synchronous errors via the SYCL exception class.

Todo

Definition at line 220 of file device.hpp.

```
00220                   {
00221     return std::any_cast<
00222       typename info::param_traits<info::device, param>::return_type
00223       >(implementation->get_info(param));
00224   }
```

References trisycl::detail::shared_ptr_implementation< device, detail::device >::implementation.

9.10.2.6.3.8 get_platform() platform trisycl::device::get_platform ( ) const [inline]

Return the platform of device.

Return synchronous errors via the SYCL exception class.

Definition at line 190 of file device.hpp.

```
00190                                  {
00191     return implementation->get_platform();
00192   }
```

References trisycl::detail::shared_ptr_implementation< device, detail::device >::implementation.

9.10.2.6.3.9 has_extension() bool trisycl::device::has_extension (const string_class & *extension*) const [inline]

Test if a specific extension is supported on the device.

Definition at line 227 of file device.hpp.

```
00227                                                          {
00228     return implementation->has_extension(extension);
00229   }
```

References trisycl::detail::shared_ptr_implementation< device, detail::device >::implementation.

9.10.2.6.3.10 is_accelerator() bool trisycl::device::is_accelerator ( ) const [inline]

Return true if the device is an OpenCL accelerator device.

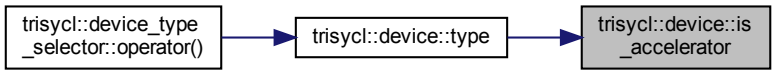
Definition at line 161 of file device.hpp.

```
00161                                  {
00162     return implementation->is_accelerator();
00163   }
```

References trisycl::detail::shared_ptr_implementation< device, detail::device >::implementation.

Referenced by type().

Here is the caller graph for this function:



9.10.2.6.3.11 is_cpu() bool trisycl::device::is_cpu ( ) const [inline]

Return true if the device is an OpenCL CPU device.

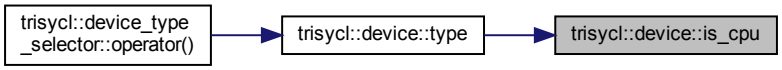
Definition at line 149 of file device.hpp.

```
00149                       {
00150     return implementation->is_cpu();
00151   }
```

References trisycl::detail::shared_ptr_implementation< device, detail::device >::implementation.

Referenced by type().

Here is the caller graph for this function:



9.10.2.6.3.12 is_gpu() bool trisycl::device::is_gpu ( ) const [inline]

Return true if the device is an OpenCL GPU device.

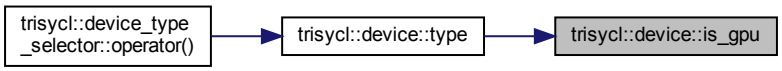
Definition at line 155 of file device.hpp.

```
00155                       {
00156     return implementation->is_gpu();
00157   }
```

References trisycl::detail::shared_ptr_implementation< device, detail::device >::implementation.

Referenced by type().

Here is the caller graph for this function:



9.10.2.6.3.13 is_host() `bool trisycl::device::is_host ( ) const [inline]`

Return true if the device is the SYCL host device.

Definition at line 143 of file device.hpp.

```
00143                          {
00144     return implementation->is_host();
00145   }
```

References trisycl::detail::shared_ptr_implementation< device, detail::device >::implementation.

Referenced by trisycl::device_type_selector::operator()(), and type().

Here is the caller graph for this function:



9.10.2.6.3.14 type() `info::device_type trisycl::device::type ( ) const [inline]`

Return the device_type of a device.

Todo Present in Boost.Compute, to be added to the specification

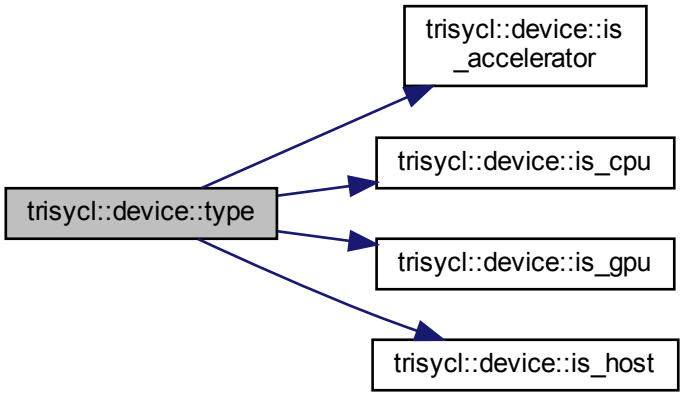
Definition at line 171 of file device.hpp.

```
00171                                  {
00172     if (is_host())
00173       return info::device_type::host;
00174     else if (is_cpu())
00175       return info::device_type::cpu;
00176     else if (is_gpu())
00177       return info::device_type::gpu;
00178     else if (is_accelerator())
00179       return info::device_type::accelerator;
00180     else
00181       // \todo Put a SYCL exception
00182       throw std::domain_error("Unknown trisycl::info::device_type");
00183   }
```

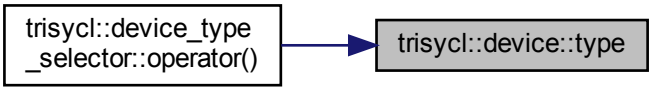
References trisycl::info::accelerator, trisycl::info::cpu, trisycl::info::gpu, trisycl::info::host, is_accelerator(), is_cpu(), is_gpu(), and is_host().

Referenced by trisycl::device_type_selector::operator()().

Here is the call graph for this function:



Here is the caller graph for this function:

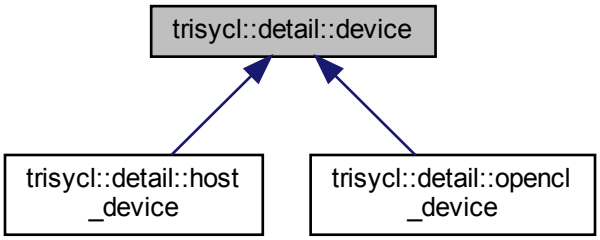


9.10.2.7 class trisycl::detail::device

An abstract class representing various models of SYCL devices.

Definition at line 23 of file device.hpp.

Inheritance diagram for trisycl::detail::device:



Public Member Functions

- virtual cl_device_id get () const =0

  Return the cl_device_id of the underlying OpenCL platform.
- virtual boost::compute::device & get_boost_compute ()=0

  Return the underlying Boost.Compute device, if any.
- virtual bool is_host () const =0

  Return true if the device is a SYCL host device.
- virtual bool is_cpu () const =0

  Return true if the device is an OpenCL CPU device.
- virtual bool is_gpu () const =0

  Return true if the device is an OpenCL GPU device.
- virtual bool is_accelerator () const =0

  Return true if the device is an OpenCL accelerator device.
- virtual trisycl::platform get_platform () const =0

  Return the platform of device.
- virtual std::any get_info (info::device param) const =0

  Query the device for OpenCL info::device info.
- virtual bool has_extension (const string_class &extension) const =0

  Specify whether a specific extension is supported on the device.
- virtual ~device ()

9.10.2.7.1 Constructor & Destructor Documentation

9.10.2.7.1.1 ~device() `virtual trisycl::detail::device::~device ( ) [inline], [virtual]`

Definition at line 65 of file device.hpp.

```
00065 {}
```

9.10.2.7.2 Member Function Documentation

9.10.2.7.2.1 get() `virtual cl_device_id trisycl::detail::device::get ( ) const [pure virtual]`

Return the cl_device_id of the underlying OpenCL platform.

Implemented in trisycl::detail::host_device, and trisycl::detail::opencl_device.

9.10.2.7.2.2 get_boost_compute() `virtual boost::compute::device & trisycl::detail::device::get_boost_compute ( ) [pure virtual]`

Return the underlying Boost.Compute device, if any.

Implemented in trisycl::detail::host_device, and trisycl::detail::opencl_device.

9.10.2.7.2.3 get_info() `virtual std::any trisycl::detail::device::get_info ( info::device param ) const [pure virtual]`

Query the device for OpenCL info::device info.

Implemented in trisycl::detail::host_device, and trisycl::detail::opencl_device.

9.10.2.7.2.4 get_platform() `virtual trisycl::platform trisycl::detail::device::get_platform ( ) const [pure virtual]`

Return the platform of device.

Implemented in trisycl::detail::host_device, and trisycl::detail::opencl_device.

9.10.2.7.2.5 has_extension() `virtual bool trisycl::detail::device::has_extension ( const string_class & extension ) const [pure virtual]`

Specify whether a specific extension is supported on the device.

Implemented in trisycl::detail::host_device, and trisycl::detail::opencl_device.

9.10.2.7.2.6 is_accelerator() `virtual bool trisycl::detail::device::is_accelerator ( ) const [pure virtual]`

Return true if the device is an OpenCL accelerator device.

Implemented in trisycl::detail::host_device, and trisycl::detail::opencl_device.

9.10.2.7.2.7 is_cpu() virtual bool trisycl::detail::device::is_cpu ( ) const [pure virtual]

Return true if the device is an OpenCL CPU device.

Implemented in trisycl::detail::host_device, and trisycl::detail::opencl_device.

9.10.2.7.2.8 is_gpu() virtual bool trisycl::detail::device::is_gpu ( ) const [pure virtual]

Return true if the device is an OpenCL GPU device.

Implemented in trisycl::detail::host_device, and trisycl::detail::opencl_device.

9.10.2.7.2.9 is_host() virtual bool trisycl::detail::device::is_host ( ) const [pure virtual]

Return true if the device is a SYCL host device.

Implemented in trisycl::detail::host_device, and trisycl::detail::opencl_device.

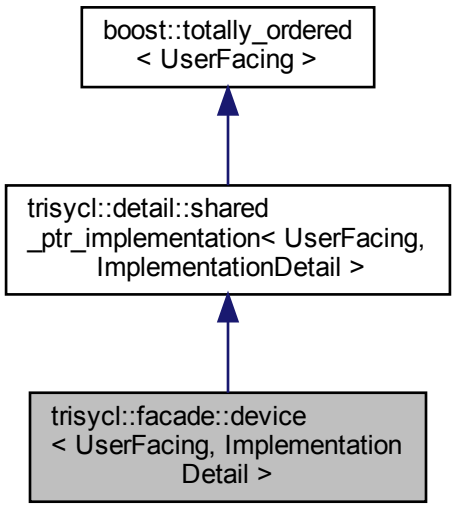
9.10.2.8 class trisycl::facade::device

template<typename UserFacing, typename ImplementationDetail>
class trisycl::facade::device< UserFacing, ImplementationDetail >

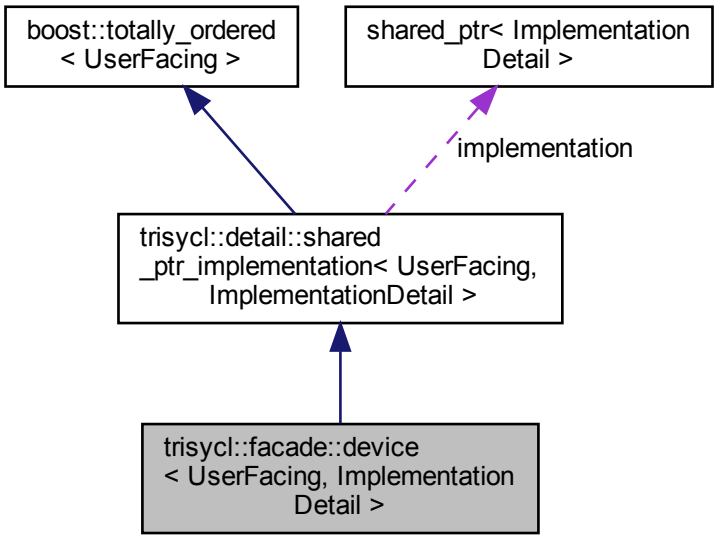
SYCL device.

Definition at line 21 of file device.hpp.

Inheritance diagram for trisycl::facade::device< UserFacing, ImplementationDetail >:



Collaboration diagram for trisycl::facade::device< UserFacing, ImplementationDetail >:



Public Types

- using implementation_t = detail::shared_ptr_implementation< UserFacing, ImplementationDetail >
  The type encapsulating the implementation.
- using detail_t = ImplementationDetail
  The real detail type doing the work behind the scene.

Additional Inherited Members

9.10.2.8.1 Member Typedef Documentation

9.10.2.8.1.1 detail_t

```
template<typename UserFacing , typename ImplementationDetail >
using trisycl::facade::device< UserFacing, ImplementationDetail >::detail_t = Implementation↩
Detail
```

The real detail type doing the work behind the scene.

Definition at line 33 of file device.hpp.

9.10.2.8.1.2 implementation_t

```
template<typename UserFacing , typename ImplementationDetail >
using trisycl::facade::device< UserFacing, ImplementationDetail >::implementation_t = detail::shared_ptr_impl
Facing, ImplementationDetail>
```

The type encapsulating the implementation.

Definition at line 29 of file device.hpp.

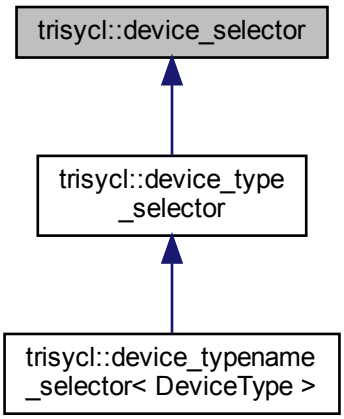
9.10.2.9 class trisycl::device_selector

The SYCL heuristics to select a device.

The device with the highest score is selected

Definition at line 22 of file device_selector.hpp.

Inheritance diagram for trisycl::device_selector:



Public Member Functions

- device select_device () const

  Returns a selected device using the functor operator defined in sub-classes operator()(const device &dev)
- virtual int operator() (const device &dev) const =0

  This pure virtual operator allows the customization of device selection.
- virtual ~device_selector ()

  Virtual destructor so the final destructor can be called if any.

Private Member Functions

- device select_device (vector_class< platform > platforms) const

9.10.2.9.1 Constructor & Destructor Documentation

9.10.2.9.1.1 ~device_selector() `virtual trisycl::device_selector::~device_selector ( ) [inline], [virtual]`

Virtual destructor so the final destructor can be called if any.

Definition at line 46 of file device_selector.hpp.

```
00046 {}
```

9.10.2.9.2 Member Function Documentation

9.10.2.9.2.1 operator()() `virtual int trisycl::device_selector::operator() ( const device & dev ) const [pure virtual]`

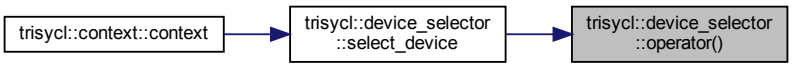
This pure virtual operator allows the customization of device selection.

It defines the behavior of the device_selector functor called by the SYCL runtime on device selection. It returns a "score" for each device in the system and the highest rated device will be used by the SYCL runtime.

Implemented in trisycl::device_type_selector.

Referenced by select_device().

Here is the caller graph for this function:



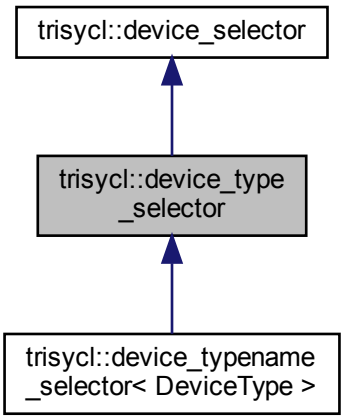
9.10.2.10 class trisycl::device_type_selector

A device selector by device_type.

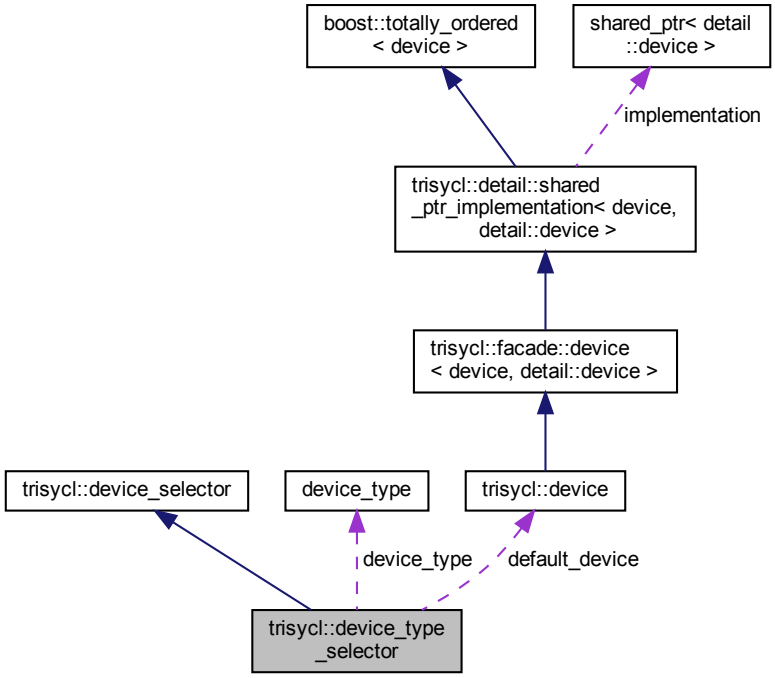
Todo To be added to the specification

Definition at line 54 of file device_selector_tail.hpp.

Inheritance diagram for trisycl::device_type_selector:



Collaboration diagram for trisycl::device_type_selector:



Public Member Functions

- device_type_selector (info::device_type device_type)
- int operator() (const device &dev) const override

  This pure virtual operator allows the customization of device selection.

Private Attributes

- info::device_type device_type

  The device_type to select.
- device default_device

  Cache the default device to select with the default device selector.

9.10.2.10.1 Constructor & Destructor Documentation

9.10.2.10.1.1 device_type_selector() `trisycl::device_type_selector::device_type_selector ( info::device_type device_type ) [inline]`

Definition at line 70 of file device_selector_tail.hpp.

```
00071     :  device_type { device_type } {
00072     // The default device selection heuristic
00073 #ifdef TRISYCL_OPENCL
00074     if (device_type == info::device_type::automatic) {
00075       // Ask Boost.Compute for the default OpenCL device
```

```
00076       try {
00077         default_device = boost::compute::system::default_device();
00078       }
00079       catch (...)  {
00080         /* If there is no OpenCL device, just keep the
00081 default-constructed device, which is the host device */
00082       }
00083     }
00084 #endif
00085   }
```

References trisycl::info::automatic, default_device, and device_type.

9.10.2.10.2 Member Function Documentation

9.10.2.10.2.1 operator()() `int trisycl::device_type_selector::operator() ( const device & dev ) const [inline], [override], [virtual]`

This pure virtual operator allows the customization of device selection.

It defines the behavior of the device_selector functor called by the SYCL runtime on device selection. It returns a "score" for each device in the system and the highest rated device will be used by the SYCL runtime.

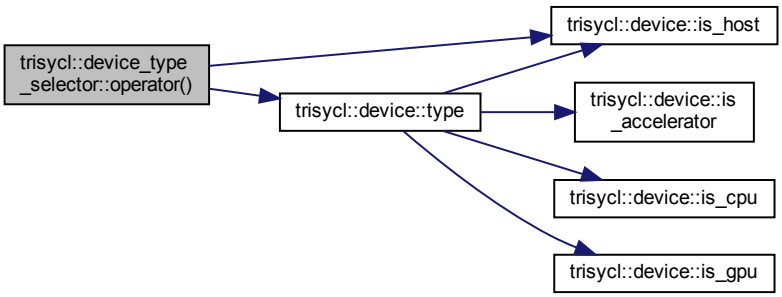
Implements trisycl::device_selector.

Definition at line 89 of file device_selector_tail.hpp.

```
00089                                                 {
00090     if (device_type == info::device_type::all)
00091       // All devices fit all
00092       return 1;
00093
00094     if (device_type == info::device_type::automatic)
00095       // Only select the default device
00096       return dev == default_device ?  1 :  -1;
00097
00098     if (device_type == info::device_type::opencl)
00099       // For now, any non host device is an OpenCL device
00100       return dev.is_host() ?  -1 :  1;
00101
00102     return dev.type() == device_type ?  1 :  -1;
00103   }
```

References trisycl::info::all, trisycl::info::automatic, default_device, device_type, trisycl::device::is_host(), trisycl::info::opencl, and trisycl::device::type().

Here is the call graph for this function:



9.10.2.10.3 Member Data Documentation

9.10.2.10.3.1 default_device `device trisycl::device_type_selector::default_device [private]`

Cache the default device to select with the default device selector.

This is the host device at construction time and remains as is if there is no openCL device

Definition at line 66 of file device_selector_tail.hpp.

Referenced by device_type_selector(), and operator()().

9.10.2.10.3.2 device_type `info::device_type trisycl::device_type_selector::device_type [private]`

The device_type to select.

Definition at line 59 of file device_selector_tail.hpp.

Referenced by device_type_selector(), and operator()().

9.10.2.11 class trisycl::device_typename_selector

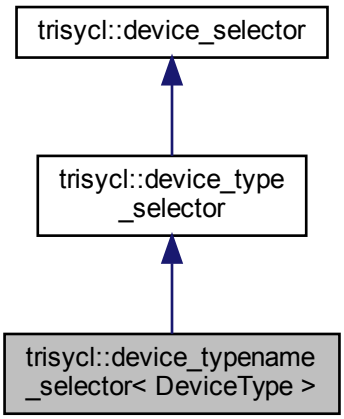
template<info::device_type DeviceType>
class trisycl::device_typename_selector< DeviceType >

Select a device by template device_type parameter.

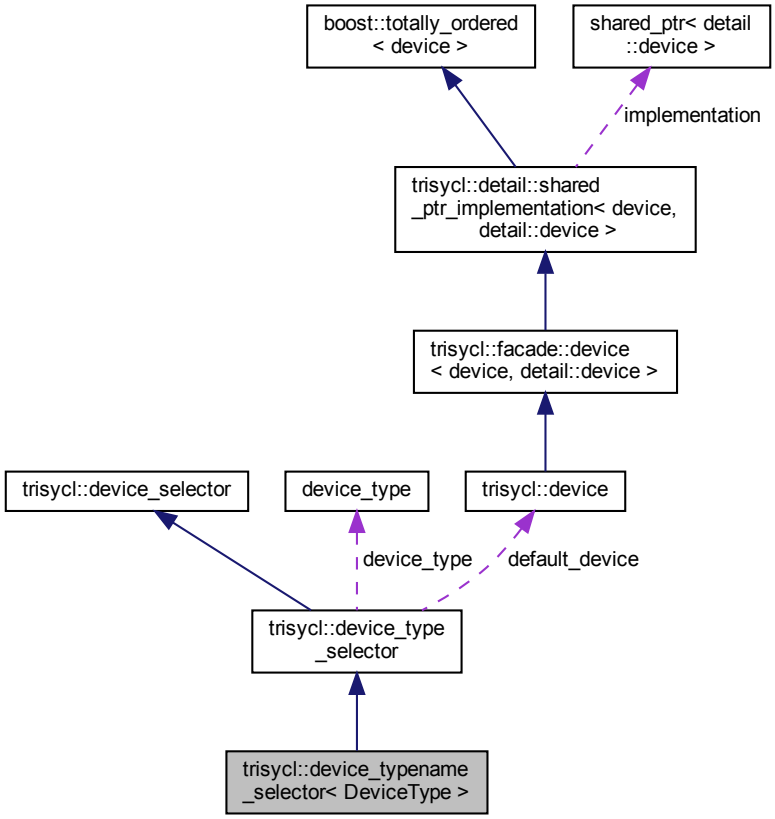
Todo To be added to the specification

Definition at line 113 of file device_selector_tail.hpp.

Inheritance diagram for trisycl::device_typename_selector< DeviceType >:



Collaboration diagram for trisycl::device_typename_selector< DeviceType >:



Public Member Functions

- device_typename_selector ()

9.10.2.11.1 Constructor & Destructor Documentation

9.10.2.11.1.1 device_typename_selector() `template<info::device_type DeviceType>`

`trisycl::device_typename_selector< DeviceType >::device_typename_selector ( ) [inline]`

Definition at line 117 of file device_selector_tail.hpp.

```
00117 : device_type_selector { DeviceType } {}
```

9.10.2.12 class trisycl::handler

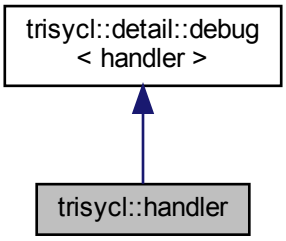
Command group handler class.

A command group handler object can only be constructed by the SYCL runtime.

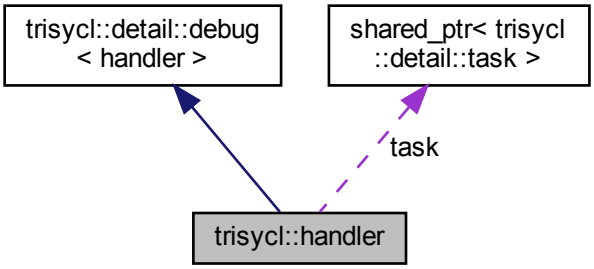
All of the accessors defined in the command group scope take as a parameter an instance of the command group handler and all the kernel invocation functions are methods of this class.

Definition at line 46 of file handler.hpp.

Inheritance diagram for trisycl::handler:



Collaboration diagram for trisycl::handler:



Public Member Functions

- handler (const std::shared_ptr< detail::queue > &q)
- template<typename DataType , int Dimensions, access::mode Mode, access::target Target = access::target::global_buffer>
  void set_arg (int arg_index, accessor< DataType, Dimensions, Mode, Target > &&acc_obj)

  Set accessor kernel arg for an OpenCL kernel which is used through the SYCL/OpenCL interop interface.
- template<typename T , typename = std::enable_if_t<is_wrapper<T>::value>>
  void set_arg (int arg_index, T &&scalar_value)

  Set kernel args for an OpenCL kernel which is used through the SYCL/OpenCL interoperability interface with a wrapper type.
- template<typename T >
  std::enable_if_t<!is_wrapper< T >::value > set_arg (int arg_index, T &&scalar_value)

  Set kernel args for an OpenCL kernel which is used through the SYCL/OpenCL interoperability interface without a wrapper type.

- template<typename... Ts>
  void set_args (Ts &&... args)

  Set all kernel args for an OpenCL kernel which is used through the SYCL/OpenCL interop interface.
- template<typename KernelName = std::nullptr_t, typename ParallelForFunctor >
  void single_task (ParallelForFunctor &&f)

  Kernel invocation method of a kernel defined as a lambda or functor.
- template<typename KernelName = std::nullptr_t, int Dims, typename ParallelForFunctor >
  requires (!std::derived_from<ParallelForFunctor, kernel>)
  void parallel_for (const range< Dims > &global_size, ParallelForFunctor f)

  SYCL parallel_for launches a data parallel computation with parallelism specified at launch time by a range<>
- template<typename KernelName = std::nullptr_t, int Dims, typename ParallelForFunctor >
  void parallel_for (const std::size_t(&global_size)[Dims], ParallelForFunctor f)

  SYCL parallel_for launches a data parallel computation with parallelism specified at launch time with a range defined with a { dim1, dim2, dim3... } syntax.
- template<typename KernelName = std::nullptr_t, typename ParallelForFunctor >
  requires (!std::derived_from<ParallelForFunctor, kernel>)
  void parallel_for (std::size_t global_size, ParallelForFunctor f)

  SYCL parallel_for launches a data parallel computation with parallelism specified at launch time with a range defined with a single dimension as an integer.
- template<typename KernelName = std::nullptr_t, int Dims, typename ParallelForFunctor >
  void parallel_for (range< Dims > global_size, id< Dims > offset, ParallelForFunctor f)

  Kernel invocation method of a kernel defined as a lambda or functor, for the specified range and offset and given an id or item for indexing in the indexing space defined by range.
- template<typename KernelName = std::nullptr_t, int Dimensions, typename ParallelForFunctor >
  void parallel_for (nd_range< Dimensions > r, ParallelForFunctor f)

  Kernel invocation method of a kernel defined as a lambda or functor, for the specified nd_range and given an nd_item for indexing in the indexing space defined by the nd_range.
- template<typename KernelName = std::nullptr_t, int Dimensions = 1, typename ParallelForFunctor >
  void parallel_for_work_group (nd_range< Dimensions > r, ParallelForFunctor f)

  Hierarchical kernel invocation method of a kernel defined as a lambda encoding the body of each work-group to launch.
- template<typename KernelName = std::nullptr_t, int Dimensions = 1, typename ParallelForFunctor >
  void parallel_for_work_group (range< Dimensions > r1, range< Dimensions > r2, ParallelForFunctor f)

  Hierarchical kernel invocation method of a kernel defined as a lambda encoding the body of each work-group to launch.
- void single_task (kernel sycl_kernel)

  Kernel invocation method of a kernel defined as pointer to a kernel object, described in detail in 3.5.3.
- template<int Dimensions = 1, typename SYCLrange >
  requires (std::derived_from<SYCLrange, range<Dimensions>> || std::integral<SYCLrange>)
  void parallel_for (SYCLrange num_work_items, kernel sycl_kernel)

  Kernel invocation method of a kernel defined as a kernel object, for the specified range and given an id or item for indexing in the indexing space defined by range, described in detail in 5.4.
- template<int Dimensions = 1>
  void parallel_for (nd_range< Dimensions >, kernel syclKernel)

  Kernel invocation method of a kernel defined as pointer to a kernel object, for the specified nd_range and given an nd_item for indexing in the indexing space defined by the nd_range, described in detail in 3.5.3.

Public Attributes

- std::shared_ptr< detail::task > task

  Attach the task and accessors to it.

Private Member Functions

- template<std::size_t... Is, typename... Ts>
  void dispatch_set_arg (std::index_sequence< Is... >, Ts &&... args)

  Helper to individually call set_arg() for each argument.
- template<typename KernelName , typename Kernel >
  void schedule_kernel (Kernel k)

  Schedule the kernel.
- template<typename KernelName , typename Kernel , int N>
  void schedule_parallel_for_kernel (Kernel k, const range< N > &num_work_items)

  Schedule a parallel for kernel.

Additional Inherited Members

9.10.2.12.1 Constructor & Destructor Documentation

9.10.2.12.1.1 handler() `trisycl::handler::handler (const std::shared_ptr< detail::queue > & q ) [inline]`

Definition at line 64 of file handler.hpp.

```
00064                                                {
00065     // Create a new task for this command_group
00066     task = std::make_shared<detail::task>(q);
00067   }
```

References task.

9.10.2.12.2 Member Function Documentation

9.10.2.12.2.1 dispatch_set_arg() `template<std::size_t... Is, typename... Ts> void trisycl::handler::dispatch_set_arg (std::index_sequence< Is... > , Ts &&... args ) [inline], [private]`

Helper to individually call set_arg() for each argument.

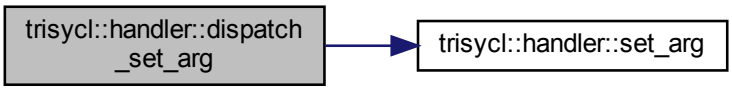
Definition at line 132 of file handler.hpp.

```
00132                                                                {
00133     // Use an intermediate tuple to ease individual argument access
00134     auto &&t = std::make_tuple(std::forward<Ts>(args)...);
00135     // Dispatch individual set_arg() for each argument
00136     auto just_to_evaluate = {
00137       0 /*< At least 1 element to deal with empty set_args() */,
00138       ( set_arg(Is, std::forward<Ts>(std::get<Is>(t))), 0)...
00139     };
00140     // Remove the warning about unused variable
00141     static_cast<void>(just_to_evaluate);
00142   }
```

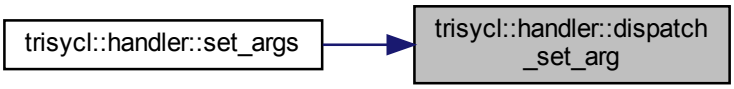
References set_arg().

Referenced by set_args().

Here is the call graph for this function:



Here is the caller graph for this function:



9.10.2.12.2.2 parallel_for() [1/7]

```
template<typename KernelName = std::nullptr_t, int Dims, typename
ParallelForFunctor >
requires (!std::derived_from<ParallelForFunctor, kernel>)
void trisycl::handler::parallel_for (
            const range< Dims > & global_size,
            ParallelForFunctor f )  [inline]
```

SYCL parallel_for launches a data parallel computation with parallelism specified at launch time by a range<>

Kernel invocation method of a kernel defined as a lambda or functor, for the specified range and given an id or item for indexing in the indexing space defined by range.

If it is a lambda function or the if the functor type is globally visible there is no need for the developer to provide a kernel name type (typename KernelName) for it.

Parameters

global_size	is the full size of the range<>

Template Parameters

Dims	dimensionality of the iteration space
KernelName	is a class type that defines the name to be used for the underlying kernel

Parameters

f	is the kernel functor to execute

Template Parameters

ParallelForFunctor	is the type of the kernel functor

Definition at line 264 of file handler.hpp.

```
00264                                                                                {
00265     if constexpr (detail::use_native_work_item) {
00266       // Use a normal parallel for
00267       schedule_parallel_for_kernel<KernelName>(
00268           [=] { detail::parallel_for(global_size, f); }, global_size);
00269     } else
00270       // Launch a single-task kernel containing the loop nests
00271       schedule_kernel<KernelName>(
00272           [=] { detail::parallel_for(global_size, f); });
00273   }
```

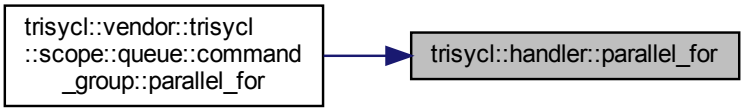
References trisycl::detail::parallel_for(), and trisycl::detail::use_native_work_item.

Referenced by trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >::parallel_for().

Here is the call graph for this function:



Here is the caller graph for this function:



9.10.2.12.2.3 parallel_for() [2/7]

```
template<typename KernelName = std::nullptr_t, int Dims, typename
ParallelForFunctor >
void trisycl::handler::parallel_for (
            const std::size_t(&) global_size[Dims],
            ParallelForFunctor f )  [inline]
```

SYCL parallel_for launches a data parallel computation with parallelism specified at launch time with a range defined with a { dim1, dim2, dim3... } syntax.

Kernel invocation method of a kernel defined as a lambda or functor, for the specified range and given an id or item for indexing in the indexing space defined by range.

If it is a lambda function or the if the functor type is globally visible there is no need for the developer to provide a kernel name type (typename KernelName) for it.

Parameters

global_size	is the full size of the range<> defined as a C array to allow type deduction from a list-initialization with { args... }

Template Parameters

Dims	dimensionality of the iteration space
KernelName	is a class type that defines the name to be used for the underlying kernel

Parameters

f	is the kernel functor to execute

Template Parameters

ParallelForFunctor	is the type of the kernel functor

Definition at line 302 of file handler.hpp.

```
00303                                          {
00304     parallel_for<KernelName>(range<Dims> { global_size }, f);
00305   }
```

9.10.2.12.2.4 parallel_for() [3/7]

```
template<typename KernelName = std::nullptr_t, int Dimensions,
typename ParallelForFunctor >
void trisycl::handler::parallel_for (
            nd_range< Dimensions > r,
            ParallelForFunctor f )  [inline]
```

Kernel invocation method of a kernel defined as a lambda or functor, for the specified nd_range and given an nd_item for indexing in the indexing space defined by the nd_range.

If it is a lambda function or the if the functor type is globally visible there is no need for the developer to provide a kernel name type (typename KernelName) for it, as described in detail in 3.5.3

Parameters

r	defines the iteration space with the work-group layout and offset
Dimensions	dimensionality of the iteration space
f	is the kernel functor to execute
ParallelForFunctor	is the kernel functor type
KernelName	is a class type that defines the name to be used for the underlying kernel

Definition at line 388 of file handler.hpp.

```
00389                                          {
00390     schedule_kernel<KernelName>([=] { detail::parallel_for(r, f); });
00391   }
```

References trisycl::detail::parallel_for().

Here is the call graph for this function:



9.10.2.12.2.5 parallel_for() [4/7] template<int Dimensions = 1>

```
void trisycl::handler::parallel_for (
             nd_range< Dimensions > ,
             kernel syclKernel )  [inline]
```

Kernel invocation method of a kernel defined as pointer to a kernel object, for the specified nd_range and given an nd_item for indexing in the indexing space defined by the nd_range, described in detail in 3.5.3.

Todo Add in the spec a version taking a kernel and a functor, to have host fall-back

Todo To be implemented

Definition at line 519 of file handler.hpp.

```
00519                                                  {
00520     TRISYCL_UNIMPL;
00521   }
```

References TRISYCL_UNIMPL.

9.10.2.12.2.6 parallel_for() [5/7] template<typename KernelName = std::nullptr_t, int Dims, typename ParallelForFunctor >

```
void trisycl::handler::parallel_for (
             range< Dims > global_size,
             id< Dims > offset,
             ParallelForFunctor f )  [inline]
```

Kernel invocation method of a kernel defined as a lambda or functor, for the specified range and offset and given an id or item for indexing in the indexing space defined by range.

If it is a lambda function or the if the functor type is globally visible there is no need for the developer to provide a kernel name type (typename KernelName) for it, as described in detail in 3.5.3

Parameters

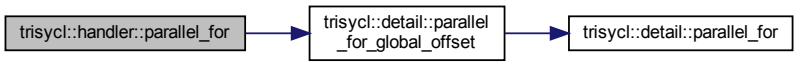
global_size	is the global size of the range<>
offset	is the offset to be add to the id<> during iteration
f	is the kernel functor to execute
ParallelForFunctor	is the kernel functor type
KernelName	is a class type that defines the name to be used for the underlying kernel

Definition at line 359 of file handler.hpp.

```
00360                                       {
00361     schedule_kernel<KernelName>(
00362         [=] { detail::parallel_for_global_offset(global_size, offset, f); });
00363   }
```

References offset, and trisycl::detail::parallel_for_global_offset().

Here is the call graph for this function:



9.10.2.12.2.7 parallel_for() [6/7]

```
template<typename KernelName = std::nullptr_t, typename ParallelForFunctor >
requires (!std::derived_from<ParallelForFunctor, kernel>)
void trisycl::handler::parallel_for (
             std::size_t global_size,
             ParallelForFunctor f )  [inline]
```

SYCL parallel_for launches a data parallel computation with parallelism specified at launch time with a range defined with a single dimension as an integer.

Kernel invocation method of a kernel defined as a lambda or functor, for the specified range and given an id or item for indexing in the indexing space defined by range.

If it is a lambda function or the if the functor type is globally visible there is no need for the developer to provide a kernel name type (typename KernelName) for it.

Parameters

global_size	is the full size of the range<1> defined as an integer

Template Parameters

KernelName	is a class type that defines the name to be used for the underlying kernel

Parameters

f	is the kernel functor to execute

Template Parameters

ParallelForFunctor	is the type of the kernel functor

Definition at line 333 of file handler.hpp.

```
00334                                          {
00335     parallel_for<KernelName>(range { global_size }, f);
00336   }
```

9.10.2.12.2.8 parallel_for() [7/7]

```
template<int Dimensions = 1, typename SYCLrange >
requires (std::derived_from<SYCLrange, range<Dimensions>> || std::integral<SYCLrange>)
void trisycl::handler::parallel_for (
             SYCLrange num_work_items,
             kernel sycl_kernel )  [inline]
```

Kernel invocation method of a kernel defined as a kernel object, for the specified range and given an id or item for indexing in the indexing space defined by range, described in detail in 5.4.

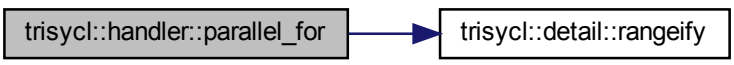
Todo Add in the spec a version taking a kernel and a functor, to have host fall-back

Definition at line 489 of file handler.hpp.

```
00489                                                          {
00490     /* For now just use the usual host task system to schedule
00491 manually the OpenCL kernels instead of using OpenCL event-based
00492 scheduling
00493
00494 todo Move the tracing inside the kernel implementation
00495
00496 todo Simplify this 2 step ugly interface
00497 */
00498     task->set_kernel(sycl_kernel.implementation);
00499     /* Use an intermediate variable to capture task by copy because
00500 otherwise "this" is captured by reference and havoc with task
00501 just accessing the dead "this".  Nasty bug to find...  */
00502     task->schedule(detail::trace_kernel<kernel>([=, t = task] {
00503       sycl_kernel.implementation->parallel_for(t, t->get_queue(),
00504                                                detail::rangeify(num_work_items));
00505     }));
00506   }
```

References trisycl::detail::shared_ptr_implementation< Parent, Implementation >::implementation, trisycl::detail::rangeify(), and task.

Here is the call graph for this function:



9.10.2.12.2.9 parallel_for_work_group() [1/2]

```
template<typename KernelName = std::nullptr_t, int Dimensions = 1, typename ParallelForFunctor >
void trisycl::handler::parallel_for_work_group (
            nd_range< Dimensions > r,
            ParallelForFunctor f )  [inline]
```

Hierarchical kernel invocation method of a kernel defined as a lambda encoding the body of each work-group to launch.

May contain multiple kernel built-in parallel_for_work_item functions representing the execution on each work-item.

Launch num_work_groups work-groups of runtime-defined size. Described in detail in 3.5.3.

Parameters

r	defines the iteration space with the work-group layout and offset
Dimensions	dimensionality of the iteration space
f	is the kernel functor to execute
ParallelForFunctor	is the kernel functor type
KernelName	is a class type that defines the name to be used for the underlying kernel

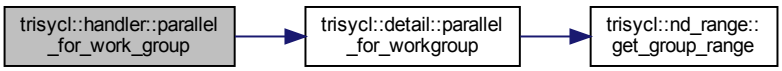
Definition at line 418 of file handler.hpp.

```
00419                                                    {
00420     schedule_kernel<KernelName>([=] {
00421         detail::parallel_for_workgroup(r, f);
00422       });
00423   }
```

References trisycl::detail::parallel_for_workgroup().

Referenced by parallel_for_work_group().

Here is the call graph for this function:



Here is the caller graph for this function:



9.10.2.12.2.10 parallel_for_work_group() [2/2] template<typename KernelName = std::nullptr_t, int Dimensions = 1, typename ParallelForFunctor >

```
void trisycl::handler::parallel_for_work_group (
             range< Dimensions > r1,
             range< Dimensions > r2,
             ParallelForFunctor f )  [inline]
```

Hierarchical kernel invocation method of a kernel defined as a lambda encoding the body of each work-group to launch.

May contain multiple kernel built-in parallel_for_work_item functions representing the execution on each work-item.

Launch num_work_groups work-groups of runtime-defined size. Described in detail in 3.5.3.

Parameters

r	defines the iteration space with the work-group layout and offset
Dimensions	dimensionality of the iteration space
f	is the kernel functor to execute
ParallelForFunctor	is the kernel functor type
KernelName	is a class type that defines the name to be used for the underlying kernel

Definition at line 450 of file handler.hpp.

```
00451                                                  {
00452     parallel_for_work_group(nd_range<Dimensions> { r1, r2 }, f);
00453   }
```

References parallel_for_work_group().

Here is the call graph for this function:



9.10.2.12.2.11 schedule_kernel() template<typename KernelName , typename Kernel >

```
void trisycl::handler::schedule_kernel (
             Kernel k )  [inline], [private]
```

Schedule the kernel.

Add a traced version of the kernel in host mode or add the kernel in an instantiating function for later extraction by the compiler

Definition at line 171 of file handler.hpp.

```
00171                                  {
00172     /* Explicitly capture task by copy instead of having this captured
00173 by reference and task by reference by side effect */
00174     task->schedule(detail::trace_kernel<KernelName>([=, t = task] () mutable {
00175           if (t->owner_queue->is_host())
00176             k();
00177           else {
```

```
00178             TRISYCL_DUMP_T("schedule_kernel &k = " « (void *) &k);
00179
00180             // Structure the kernel code so it can be outlined and called
00181             detail::launch_device_kernel<KernelName>(*t, k);
00182             // \todo for now only deal with 1 physical work-item only
00183             t->get_kernel().single_task(t, t->get_queue());
00184           }
00185         }));
00186   }
```

References task, and TRISYCL_DUMP_T.

9.10.2.12.2.12 schedule_parallel_for_kernel()

```
template<typename KernelName , typename Kernel , int N>
void trisycl::handler::schedule_parallel_for_kernel (
             Kernel k,
             const range< N > & num_work_items )  [inline], [private]
```

Schedule a parallel for kernel.

Todo Add host fall-back execution for parallel_for_kernel

Add a traced version of the kernel in host mode or add the kernel in an instantiating function for later extraction by the compiler and then execute it using parallel_for

Definition at line 199 of file handler.hpp.

```
00199                                                                          {
00200     task->schedule(detail::trace_kernel<KernelName>([=, t = task] () mutable {
00201           // if (t->owner_queue->is_host())
00202              // k();
00203           // else {
00204             TRISYCL_DUMP_T("schedule_parallel_for_kernel &k = " « (void *) &k);
00205
00206             // Structure the kernel code so it can be outlined and called
00207             detail::launch_device_kernel<KernelName>(*t, k);
00208
00209             t->get_kernel().parallel_for(t, t->get_queue(), num_work_items);
00210       // }
00211     }));
00212   }
```

References task, and TRISYCL_DUMP_T.

9.10.2.12.2.13 set_arg() [1/3]

```
template<typename DataType , int Dimensions, access::mode Mode,
access::target Target = access::target::global_buffer>
void trisycl::handler::set_arg (
             int arg_index,
             accessor< DataType, Dimensions, Mode, Target > && acc_obj )  [inline]
```

Set accessor kernel arg for an OpenCL kernel which is used through the SYCL/OpenCL interop interface.

The index value specifies which parameter of the OpenCL kernel is being set and the accessor object, which OpenCL buffer or image is going to be given as kernel argument.

Todo Update the specification to use a ref && to the accessor instead?

Todo It is not that clean to have set_arg() associated to a command handler. Rethink the specification?

Todo It seems more logical to have these methods on kernel instead

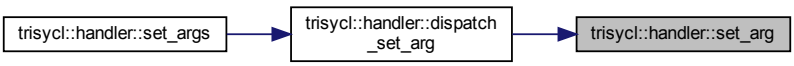
Definition at line 89 of file handler.hpp.

```
00090                                                                    {
00091     /* Think about setting the kernel argument before actually calling
00092 the kernel.
00093
00094 Explicitly capture task by copy instead of having this captured
00095 by reference and task by reference by side effect */
00096     task->add_prelude([=, task = task] {
00097         task->set_arg(arg_index, acc_obj.implementation->get_cl_buffer());
00098       });
00099   }
```

References task.

Referenced by dispatch_set_arg().

Here is the caller graph for this function:



9.10.2.12.2.14 set_arg() [2/3] template<typename T , typename = std::enable_if_t<is_wrapper<↩ T>::value>>

```
void trisycl::handler::set_arg (
             int arg_index,
             T && scalar_value )  [inline]
```

Set kernel args for an OpenCL kernel which is used through the SYCL/OpenCL interoperability interface with a wrapper type.

Definition at line 106 of file handler.hpp.

```
00106                                                {
00107     /* Explicitly capture task by copy instead of having this captured
00108 by reference and task by reference by side effect */
00109     task->add_prelude([=, task = task] {
00110         task->set_arg(arg_index, scalar_value.unwrap());
00111       });
00112   }
```

References task.

9.10.2.12.2.15 set_arg() [3/3] template<typename T >

```
std::enable_if_t<!is_wrapper< T >::value > trisycl::handler::set_arg (
             int arg_index,
             T && scalar_value )  [inline]
```

Set kernel args for an OpenCL kernel which is used through the SYCL/OpenCL interoperability interface without a wrapper type.

Definition at line 120 of file handler.hpp.

```
00120                                                {
00121     /* Explicitly capture task by copy instead of having this captured
00122 by reference and task by reference by side effect */
00123     task->add_prelude([=, task = task] {
00124         task->set_arg(arg_index, scalar_value);
00125       });
00126   }
```

References task.

9.10.2.12.2.16 set_args() template<typename... Ts>

```
void trisycl::handler::set_args (
            Ts &&... args ) [inline]
```

Set all kernel args for an OpenCL kernel which is used through the SYCL/OpenCL interop interface.

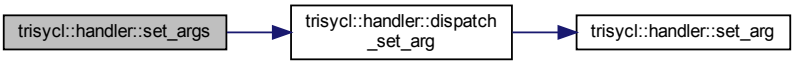
Todo Update the specification to add this function according to https://cvs.khronos.org/bugzilla/show_bug.cgi?id=15978 proposal

Definition at line 153 of file handler.hpp.

```
00153                                {
00154     /* Construct a set of increasing argument index to be able to call
00155 the real set_arg */
00156     dispatch_set_arg(std::index_sequence_for<Ts...>{},
00157                      std::forward<Ts>(args)...);
00158   }
```

References dispatch_set_arg().

Here is the call graph for this function:



9.10.2.12.2.17 single_task() [1/2] void trisycl::handler::single_task (

```
            kernel sycl_kernel ) [inline]
```

Kernel invocation method of a kernel defined as pointer to a kernel object, described in detail in 3.5.3.

Todo Add in the spec a version taking a kernel and a functor, to have host fall-back

Definition at line 462 of file handler.hpp.

```
00462                                    {
00463     /* For now just use the usual host task system to schedule
00464 manually the OpenCL kernels instead of using OpenCL event-based
00465 scheduling
00466
00467 \todo Move the tracing inside the kernel implementation
00468
00469 \todo Simplify this 2 step ugly interface
00470 */
00471     task->set_kernel(sycl_kernel.implementation);
00472     task->schedule(detail::trace_kernel<kernel>([=, t = task] {
00473           sycl_kernel.implementation->single_task(t, t->get_queue());
00474         }));
00475   }
```

References trisycl::detail::shared_ptr_implementation< Parent, Implementation >::implementation, and task.

9.10.2.12.2.18 single_task() [2/2] template<typename KernelName = std::nullptr_t, typename Parallel↩ ForFunctor >

```
void trisycl::handler::single_task (
            ParallelForFunctor && f ) [inline]
```

Kernel invocation method of a kernel defined as a lambda or functor.

If it is a lambda function or the functor type is globally visible there is no need for the developer to provide a kernel name type (typename KernelName) for it, as described in 3.5.3

SYCL single_task launches a computation without parallelism at launch time.

Parameters

F	specify the kernel to be launched as a single_task
KernelName	is a class type that defines the name to be used for the underlying kernel

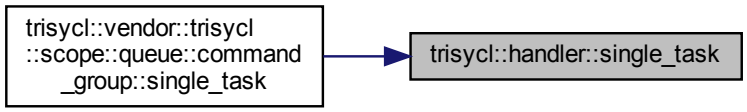
Definition at line 231 of file handler.hpp.

```
00231                                          {
00232     TRISYCL_DUMP_T("single_task &f = " « (void *) &f);
00233
00234     schedule_kernel<KernelName>(std::forward<ParallelForFunctor>(f));
00235   }
```

References TRISYCL_DUMP_T.

Referenced by trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >::single_task().

Here is the caller graph for this function:



9.10.2.12.3 Member Data Documentation

9.10.2.12.3.1 task `std::shared_ptr<detail::task> trisycl::handler::task`

Attach the task and accessors to it.

Definition at line 52 of file handler.hpp.

Referenced by trisycl::detail::add_buffer_to_task(), handler(), parallel_for(), schedule_kernel(), schedule_parallel_for_kernel(), set_arg(), and single_task().

9.10.2.13 class trisycl::kernel

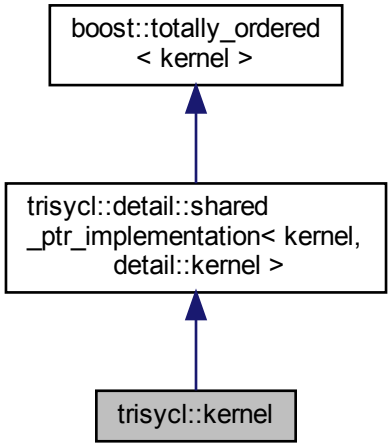
SYCL kernel.

Todo To be implemented

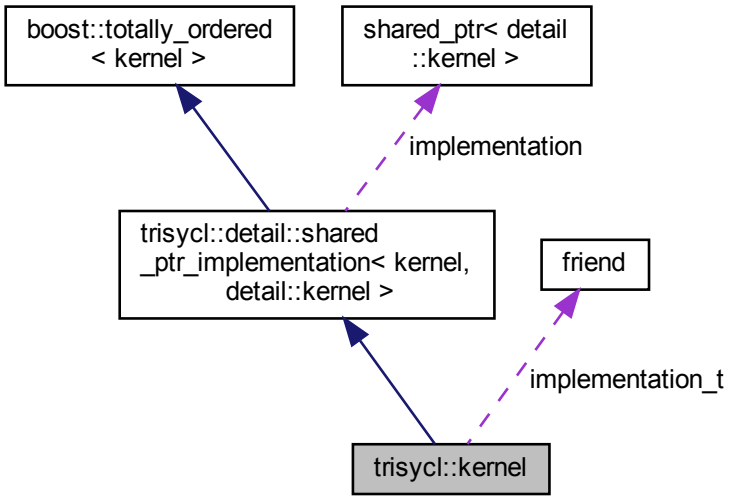
Todo Check specification

Definition at line 37 of file kernel.hpp.

Inheritance diagram for trisycl::kernel:



Collaboration diagram for trisycl::kernel:



Public Member Functions

- kernel ()=delete

The default object is not valid because there is no program or.

- kernel (cl_kernel k)

  Constructor for SYCL kernel class given an OpenCL kernel object with set arguments, valid for enqueuing.
- kernel (const boost::compute::kernel &k)

  Construct a kernel class instance using a boost::compute::kernel.
- cl_kernel get () const

  Return the OpenCL kernel object for this kernel.

Private Types

- using implementation_t = typename kernel::shared_ptr_implementation

Private Attributes

- friend implementation_t

Friends

- class handler

Additional Inherited Members

9.10.2.13.1 Member Typedef Documentation

9.10.2.13.1.1 implementation_t `using trisycl::kernel::implementation_t = typename kernel::shared_ptr_implementa [private]`

Definition at line 43 of file kernel.hpp.

9.10.2.13.2 Constructor & Destructor Documentation

9.10.2.13.2.1 kernel() [1/3] `trisycl::kernel::kernel ( ) [delete]`

The default object is not valid because there is no program or.
`cl_kernel`

associated with it

9.10.2.13.2.2 kernel() [2/3] `trisycl::kernel::kernel ( cl_kernel k ) [inline]`

Constructor for SYCL kernel class given an OpenCL kernel object with set arguments, valid for enqueuing.

Retains a reference to the `cl_kernel` object. The Caller should release the passed cl_kernel object when it is no longer needed.

Definition at line 68 of file kernel.hpp.
```
00068 :  kernel { boost::compute::kernel { k } } {}
```

9.10.2.13.2.3 kernel() [3/3] trisycl::kernel::kernel (const boost::compute::kernel & k) [inline]

Construct a kernel class instance using a boost::compute::kernel.

This is a triSYCL extension for boost::compute interoperation.

Return synchronous errors via the SYCL exception class.

Definition at line 77 of file kernel.hpp.

```
00078     : implementation_t { detail::opencl_kernel::instance(k) } {}
```

9.10.2.13.3 Member Function Documentation

9.10.2.13.3.1 get() cl_kernel trisycl::kernel::get ( ) const [inline]

Return the OpenCL kernel object for this kernel.

Retains a reference to the returned cl_kernel object. Caller should release it when finished.

Definition at line 86 of file kernel.hpp.

```
00086                          {
00087     return implementation->get();
00088   }
```

References trisycl::detail::shared_ptr_implementation< kernel, detail::kernel >::implementation.

9.10.2.13.4 Friends And Related Function Documentation

9.10.2.13.4.1 handler friend class handler [friend]

Definition at line 46 of file kernel.hpp.

9.10.2.13.5 Member Data Documentation

9.10.2.13.5.1 implementation_t friend trisycl::kernel::implementation_t [private]

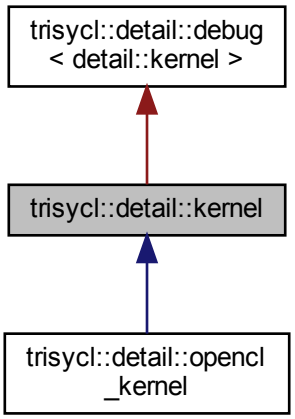
Definition at line 49 of file kernel.hpp.

9.10.2.14 class trisycl::detail::kernel

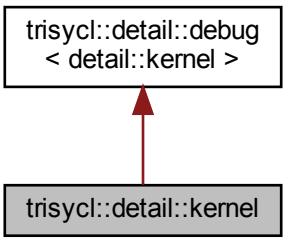
Abstract SYCL kernel.

Definition at line 29 of file kernel.hpp.

Inheritance diagram for trisycl::detail::kernel:



Collaboration diagram for trisycl::detail::kernel:



Public Member Functions

- virtual cl_kernel get () const =0

  Return the OpenCL kernel object for this kernel.
- virtual boost::compute::kernel & get_boost_compute ()=0

  Return the Boost.Compute OpenCL kernel object for this kernel.
- virtual void single_task (std::shared_ptr< detail::task > task, std::shared_ptr< detail::queue > q)=0

  Launch a single task of the kernel.
- TRISYCL_ParallelForKernel_RANGE (1) TRISYCL_ParallelForKernel_RANGE(2) TRISYCL_ParallelFor↩ Kernel_RANGE(3) virtual ~kernel()

  Return the context that this kernel is defined for.

Additional Inherited Members

9.10.2.14.1 Member Function Documentation

9.10.2.14.1.1 get() `virtual cl_kernel trisycl::detail::kernel::get ( ) const [pure virtual]`

Return the OpenCL kernel object for this kernel.

Retains a reference to the returned cl_kernel object. Caller should release it when finished.

Implemented in trisycl::detail::opencl_kernel.

9.10.2.14.1.2 get_boost_compute() `virtual boost::compute::kernel & trisycl::detail::kernel↩ ::get_boost_compute ( ) [pure virtual]`

Return the Boost.Compute OpenCL kernel object for this kernel.

This is an extension.

Implemented in trisycl::detail::opencl_kernel.

Referenced by trisycl::detail::task::set_arg().

Here is the caller graph for this function:



9.10.2.14.1.3 single_task() `virtual void trisycl::detail::kernel::single_task ( std::shared_ptr< detail::task > task, std::shared_ptr< detail::queue > q ) [pure virtual]`

Launch a single task of the kernel.

Implemented in trisycl::detail::opencl_kernel.

9.10.2.14.1.4 TRISYCL_ParallelForKernel_RANGE() `trisycl::detail::kernel::TRISYCL_ParallelFor↩Kernel_RANGE ( 1 ) [inline]`

Return the context that this kernel is defined for.

Return the program that this kernel is part of

Definition at line 65 of file kernel.hpp.

```
00078                       {}
```

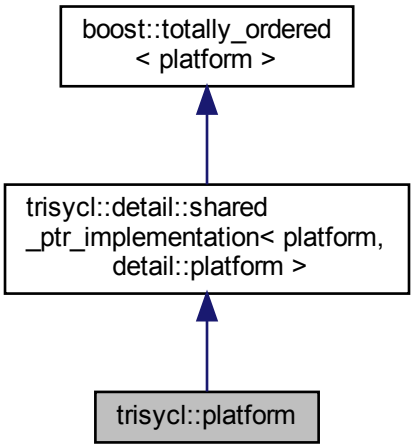
9.10.2.15 class trisycl::platform

Abstract the OpenCL platform.

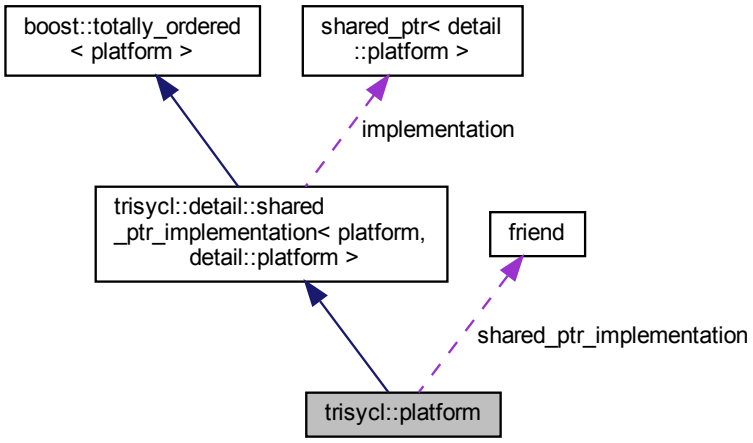
Todo triSYCL Implementation

Definition at line 40 of file platform.hpp.

Inheritance diagram for trisycl::platform:



Collaboration diagram for trisycl::platform:



Public Member Functions

- platform ()

  Default constructor for platform which is the host platform.
- platform (cl_platform_id platform_id)

  Construct a platform class instance using cl_platform_id of the OpenCL device.
- platform (const boost::compute::platform &p)

  Construct a platform class instance using a boost::compute::platform.
- platform (const device_selector &dev_selector)

  Construct a platform object from the device selected by a device selector of the user's choice.
- cl_platform_id get () const

  Returns the cl_platform_id of the underlying OpenCL platform.
- const boost::compute::platform get_boost_compute () const

  Return the underlying Boost.Compute platform if it is an OpenCL platform.
- template<info::platform param>
  auto get_info () const

  Get the OpenCL information about the requested parameter.
- bool has_extension (const string_class &extension) const

  Test if an extension is available on the platform.
- bool is_host () const

  Test if this platform is a host platform.
- vector_class< device > get_devices (info::device_type device_type=info::device_type::all) const

  Get all the available devices for this platform.

Static Public Member Functions

- static vector_class< platform > get_platforms ()

  Get the list of all the platforms available to the application.

Private Attributes

- friend shared_ptr_implementation

Additional Inherited Members

9.10.2.15.1 Constructor & Destructor Documentation

9.10.2.15.1.1 platform() [1/4] trisycl::platform::platform ( ) [inline]

Default constructor for platform which is the host platform.

Returns errors via the SYCL exception class.

Definition at line 57 of file platform.hpp.

```
00057             :
00058     shared_ptr_implementation { detail::host_platform::instance() } {}
```

9.10.2.15.1.2 platform() [2/4] trisycl::platform::platform (cl_platform_id *platform_id*) [inline], [explicit]

Construct a platform class instance using cl_platform_id of the OpenCL device.

Return synchronous errors via the SYCL exception class.

Retain a reference to the OpenCL platform.

Definition at line 69 of file platform.hpp.

```
00070     :  platform { boost::compute::platform { platform_id } } {}
```

9.10.2.15.1.3 platform() [3/4] trisycl::platform::platform (const boost::compute::platform & *p*) [inline]

Construct a platform class instance using a boost::compute::platform.

This is a triSYCL extension for boost::compute interoperation.

Return synchronous errors via the SYCL exception class.

Definition at line 79 of file platform.hpp.

```
00080     :  shared_ptr_implementation { detail::opencl_platform::instance(p) } {}
```

9.10.2.15.1.4 platform() [4/4] trisycl::platform::platform (const device_selector & *dev_selector*) [explicit]

Construct a platform object from the device selected by a device selector of the user's choice.

Returns errors via the SYCL exception class.

9.10.2.15.2 Member Function Documentation

9.10.2.15.2.1 get() `cl_platform_id trisycl::platform::get ( ) const [inline]`

Returns the cl_platform_id of the underlying OpenCL platform.

If the platform is not a valid OpenCL platform, for example if it is the SYCL host, an exception is thrown

Todo Define a SYCL exception for this

Definition at line 99 of file platform.hpp.

```
00099                                {
00100     return implementation->get();
00101   }
```

References trisycl::detail::shared_ptr_implementation< platform, detail::platform >::implementation.

9.10.2.15.2.2 get_boost_compute() `const boost::compute::platform trisycl::platform::get_boost_compute ( ) const [inline]`

Return the underlying Boost.Compute platform if it is an OpenCL platform.

This is a triSYCL extension

Definition at line 109 of file platform.hpp.

```
00109                                                {
00110     return implementation->get_boost_compute();
00111   }
```

References trisycl::detail::shared_ptr_implementation< platform, detail::platform >::implementation.

9.10.2.15.2.3 get_devices() `vector_class< device > trisycl::platform::get_devices ( info::device_type device_type = info::device_type::all ) const`

Get all the available devices for this platform.

Parameters

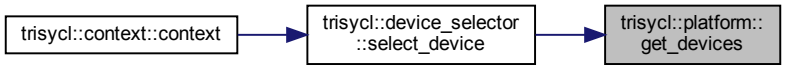
in	*device_type*	is the device type to filter the selection or `info::device_type::all` by default to return all the devices

Returns

the device list

Referenced by trisycl::device_selector::select_device().

Here is the caller graph for this function:



9.10.2.15.2.4 get_info() template<info::platform param>

`auto trisycl::platform::get_info ( ) const [inline]`

Get the OpenCL information about the requested parameter.

Todo Add to the specification

9.10.2.15.2.5 get_platforms() static vector_class< platform > trisycl::platform::get_platforms

`( ) [inline], [static]`

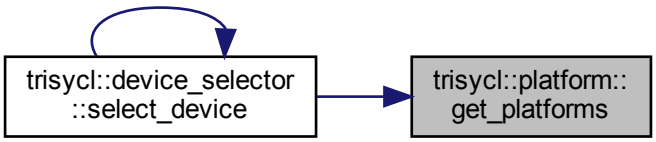
Get the list of all the platforms available to the application.

Definition at line 116 of file platform.hpp.

```
00116                                                {
00117     // Start with the default platform
00118     vector_class<platform> platforms { {} };
00119
00120 #ifdef TRISYCL_OPENCL
00121     // Then add all the OpenCL platforms
00122     for (const auto &d :  boost::compute::system::platforms())
00123       platforms.emplace_back(d);
00124 #endif
00125
00126     return platforms;
00127   }
```

Referenced by trisycl::device_selector::select_device().

Here is the caller graph for this function:



9.10.2.15.2.6 has_extension() bool trisycl::platform::has_extension (const string_class & extension) const [inline]

Test if an extension is available on the platform.

Definition at line 138 of file platform.hpp.

```
00138                                                          {
00139     return implementation->has_extension(extension);
00140   }
```

References trisycl::detail::shared_ptr_implementation< platform, detail::platform >::implementation.

9.10.2.15.2.7 is_host() bool trisycl::platform::is_host ( ) const [inline]

Test if this platform is a host platform.

Definition at line 144 of file platform.hpp.

```
00144                        {
00145     return implementation->is_host();
00146   }
```

References trisycl::detail::shared_ptr_implementation< platform, detail::platform >::implementation.

9.10.2.15.3 Member Data Documentation

9.10.2.15.3.1 shared_ptr_implementation friend trisycl::platform::shared_ptr_implementation [private]

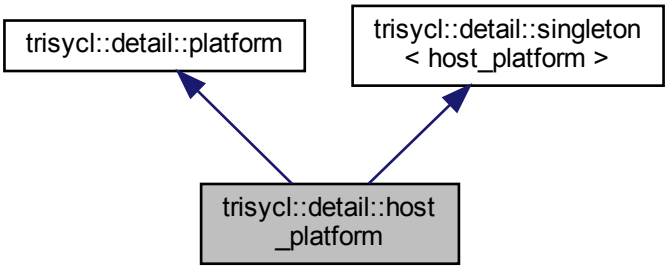
Definition at line 46 of file platform.hpp.

9.10.2.16 class trisycl::detail::host_platform

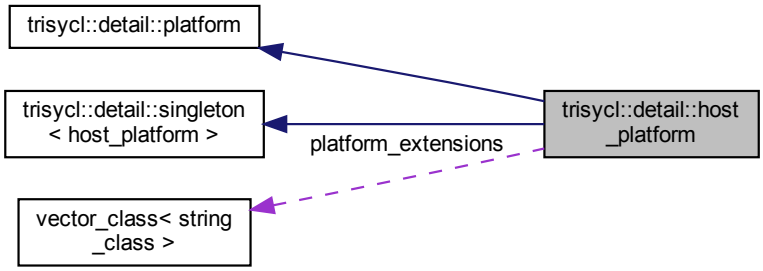
SYCL host platform.

Definition at line 35 of file host_platform.hpp.

Inheritance diagram for trisycl::detail::host_platform:



Collaboration diagram for trisycl::detail::host_platform:



Public Member Functions

- cl_platform_id get () const override

  Return the cl_platform_id of the underlying OpenCL platform.
- boost::compute::platform & get_boost_compute () const override

  Return the underlying Boost.Compute platform.
- bool is_host () const override

  Return true since this platform is the SYCL host platform.
- string_class get_info_string (info::platform param) const override

  Returning the information parameters for the host platform implementation.
- vector_class< string_class > get_extension_strings (void) const override
- bool has_extension (const string_class &extension) const override

  Specify whether a specific extension is supported on the platform.
- vector_class< trisycl::device > get_devices (const device_selector &device_selector) const override

  Get all the available devices for the host platform.

Private Attributes

- vector_class< string_class > platform_extensions { "Xilinx_blocking_pipes" }

Additional Inherited Members

9.10.2.16.1 Member Function Documentation

9.10.2.16.1.1 get() `cl_platform_id trisycl::detail::host_platform::get ( ) const [inline], [override], [virtual]`

Return the cl_platform_id of the underlying OpenCL platform.

This throws an error since there is no OpenCL platform associated to the host platform.

Implements trisycl::detail::platform.

Definition at line 51 of file host_platform.hpp.

```
00051                                  {
00052     throw non_cl_error("The host platform has no OpenCL platform");
00053   }
```

9.10.2.16.1.2 get_boost_compute() boost::compute::platform & trisycl::detail::host_platform↩ ::get_boost_compute ( ) const [inline], [override], [virtual]

Return the underlying Boost.Compute platform.

This throws an error since there is no Boost Compute platform associated to the host platform.

Implements trisycl::detail::platform.

Definition at line 61 of file host_platform.hpp.

```
00061                                                          {
00062     throw
00063       non_cl_error("The host device has no underlying Boost Compute platform");
00064   }
```

9.10.2.16.1.3 get_extension_strings() vector_class< string_class > trisycl::detail::host_↩ platform::get_extension_strings (

void) const [inline], [override], [virtual]

Implements trisycl::detail::platform.

Definition at line 101 of file host_platform.hpp.

```
00101                                                                  {
00102     return platform_extensions;
00103   }
```

References platform_extensions.

9.10.2.16.1.4 get_info_string() string_class trisycl::detail::host_platform::get_info_string (

info::platform *param*) const [inline], [override], [virtual]

Returning the information parameters for the host platform implementation.

Implements trisycl::detail::platform.

Definition at line 77 of file host_platform.hpp.

```
00077                                                                  {
00078     switch (param) {
00079     case info::platform::profile:
00080       /* Well...  Is the host platform really a full profile whereas it
00081 is not really OpenCL? */
00082       return "FULL_PROFILE";
00083
00084     case info::platform::version:
00085       // \todo I guess it should include the software version too...
00086       return "2.2";
00087
00088     case info::platform::name:
00089       return "triSYCL host platform";
00090
00091     case info::platform::vendor:
00092       return "triSYCL Open Source project";
00093
00094     default:
00095       // \todo Define some SYCL exception type for this type of errors
00096       throw std::invalid_argument {
00097         "Unknown parameter value for SYCL platform information" };
00098     }
00099   }
```

9.10.2.16.1.5 has_extension() bool trisycl::detail::host_platform::has_extension (const string_class & *extension*) const [inline], [override], [virtual]

Specify whether a specific extension is supported on the platform.

Todo To be implemented

Implements trisycl::detail::platform.

Definition at line 110 of file host_platform.hpp.

```
00110                                                                    {
00111     return std::find(platform_extensions.begin(), platform_extensions.end(), extension) !=
00112            platform_extensions.end();
00113   }
```

References platform_extensions.

9.10.2.16.1.6 is_host() bool trisycl::detail::host_platform::is_host ( ) const [inline], [override], [virtual]

Return true since this platform is the SYCL host platform.

Implements trisycl::detail::platform.

Definition at line 69 of file host_platform.hpp.

```
00069                                {
00070     return true;
00071   }
```

9.10.2.16.2 Member Data Documentation

9.10.2.16.2.1 platform_extensions vector_class<string_class> trisycl::detail::host_platform::platform_extensions { "Xilinx_blocking_pipes" } [private]

Definition at line 41 of file host_platform.hpp.

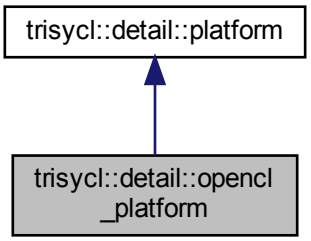
Referenced by get_extension_strings(), and has_extension().

9.10.2.17 class trisycl::detail::opencl_platform

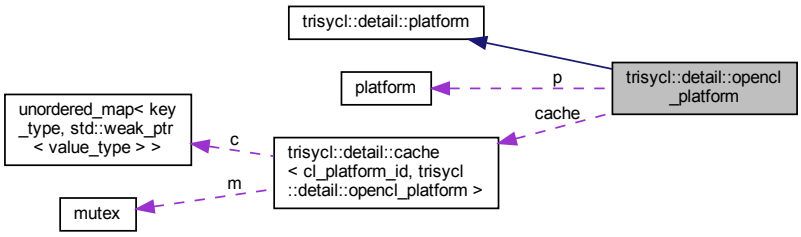
SYCL OpenCL platform.

Definition at line 34 of file opencl_platform.hpp.

Inheritance diagram for trisycl::detail::opencl_platform:



Collaboration diagram for trisycl::detail::opencl_platform:



Public Member Functions

- cl_platform_id get () const override

  Return the cl_platform_id of the underlying OpenCL platform.
- const boost::compute::platform & get_boost_compute () const override

  Return the underlying Boost.Compute platform.
- bool is_host () const override

  Return false since an OpenCL platform is not the SYCL host platform.
- string_class get_info_string (info::platform param) const override

  Returning the information string parameters for the OpenCL platform.
- vector_class< string_class > get_extension_strings () const override
- bool has_extension (const string_class &extension) const override

  Specify whether a specific extension is supported on the platform.
- vector_class< trisycl::device > get_devices (const device_selector &device_selector) const override

  Get all the available devices for this OpenCL platform.
- ~opencl_platform () override

  Unregister from the cache on destruction.

Static Public Member Functions

- static std::shared_ptr< opencl_platform > instance (const boost::compute::platform &p)

Private Member Functions

- opencl_platform (const boost::compute::platform &p)

  Only the instance factory can built it.

Private Attributes

- boost::compute::platform p

  Use the Boost Compute abstraction of the OpenCL platform.

Static Private Attributes

- static detail::cache< cl_platform_id, detail::opencl_platform > cache

  A cache to always return the same live platform for a given OpenCL platform.

9.10.2.17.1 Constructor & Destructor Documentation

9.10.2.17.1.1 opencl_platform() `trisycl::detail::opencl_platform::opencl_platform ( const boost::compute::platform & p ) [inline], [private]`

Only the instance factory can built it.

Definition at line 107 of file opencl_platform.hpp.

```
00107 : p { p } {}
```

9.10.2.17.1.2 ~opencl_platform() `trisycl::detail::opencl_platform::~opencl_platform ( ) [inline], [override]`

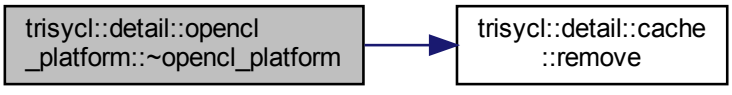
Unregister from the cache on destruction.

Definition at line 112 of file opencl_platform.hpp.

```
00112                                {
00113     cache.remove(p.id());
00114   }
```

References p, and trisycl::detail::cache< Key, Value >::remove().

Here is the call graph for this function:



9.10.2.17.2 Member Function Documentation

9.10.2.17.2.1 get() cl_platform_id trisycl::detail::opencl_platform::get ( ) const [inline], [override], [virtual]

Return the cl_platform_id of the underlying OpenCL platform.

Implements trisycl::detail::platform.

Definition at line 49 of file opencl_platform.hpp.

```
00049                                        {
00050     return p.id();
00051   }
```

References p.

9.10.2.17.2.2 get_boost_compute() const boost::compute::platform & trisycl::detail::opencl_↩ platform::get_boost_compute ( ) const [inline], [override], [virtual]

Return the underlying Boost.Compute platform.

Implements trisycl::detail::platform.

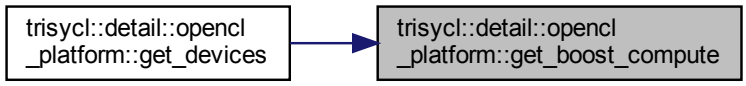
Definition at line 55 of file opencl_platform.hpp.

```
00055                                                    {
00056     return p;
00057   }
```

References p.

Referenced by get_devices().

Here is the caller graph for this function:



9.10.2.17.2.3 get_extension_strings() vector_class< string_class > trisycl::detail::opencl_↩ platform::get_extension_strings (void) const [inline], [override], [virtual]

Implements trisycl::detail::platform.

Definition at line 74 of file opencl_platform.hpp.

```
00074                                                        {
00075     return p.extensions();
00076   }
```

References p.

9.10.2.17.2.4 get_info_string()

```
string_class trisycl::detail::opencl_platform::get_info_string (
        info::platform param ) const  [inline], [override], [virtual]
```

Returning the information string parameters for the OpenCL platform.

Implements trisycl::detail::platform.

Definition at line 67 of file opencl_platform.hpp.

```
00067                                                                {
00068     /* Use the fact that the triSYCL info values are the same as the
00069 OpenCL ones used in Boost.Compute to just cast the enum class
00070 to the int value */
00071     return p.get_info<std::string>(static_cast<cl_platform_info>(param));
00072   }
```

References p.

9.10.2.17.2.5 has_extension()

```
bool trisycl::detail::opencl_platform::has_extension (
        const string_class & extension ) const  [inline], [override], [virtual]
```

Specify whether a specific extension is supported on the platform.

Implements trisycl::detail::platform.

Definition at line 80 of file opencl_platform.hpp.

```
00080                                                                {
00081     return p.supports_extension(extension);
00082   }
```

References p.

9.10.2.17.2.6 instance()

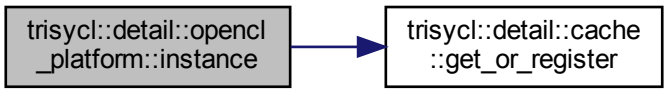
```
static std::shared_ptr< opencl_platform > trisycl::detail::opencl_↩
platform::instance (
        const boost::compute::platform & p )  [inline], [static]
```

Definition at line 87 of file opencl_platform.hpp.

```
00087                                                    {
00088     return cache.get_or_register(p.id(),
00089                                  [&] { return new opencl_platform { p }; });
00090   }
```

References trisycl::detail::cache< Key, Value >::get_or_register(), and p.

Here is the call graph for this function:



9.10.2.17.2.7 is_host() bool trisycl::detail::opencl_platform::is_host ( ) const [inline], [override], [virtual]

Return false since an OpenCL platform is not the SYCL host platform.

Implements trisycl::detail::platform.

Definition at line 61 of file opencl_platform.hpp.

```
00061                                {
00062     return false;
00063   }
```

9.10.2.17.3 Member Data Documentation

9.10.2.17.3.1 cache detail::cache<cl_platform_id, detail::opencl_platform> trisycl::detail::opencl_platform::cache [static], [private]

A cache to always return the same live platform for a given OpenCL platform.

C++11 guaranties the static construction is thread-safe

Definition at line 44 of file opencl_platform.hpp.

9.10.2.17.3.2 p boost::compute::platform trisycl::detail::opencl_platform::p [private]

Use the Boost Compute abstraction of the OpenCL platform.

Definition at line 37 of file opencl_platform.hpp.

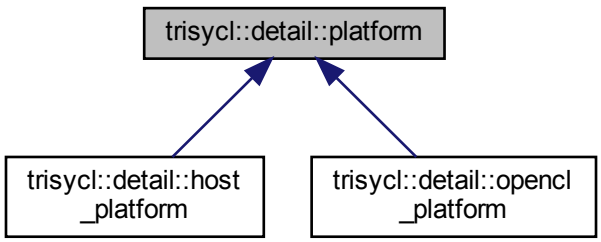
Referenced by get(), get_boost_compute(), get_extension_strings(), get_info_string(), has_extension(), and instance().

9.10.2.18 class trisycl::detail::platform

An abstract class representing various models of SYCL platforms.

Definition at line 28 of file platform.hpp.

Inheritance diagram for trisycl::detail::platform:



Public Member Functions

- virtual cl_platform_id get () const =0

  Return the cl_platform_id of the underlying OpenCL platform.
- virtual const boost::compute::platform & get_boost_compute () const =0

  Return the underlying Boost.Compute platform, if any.
- virtual bool is_host () const =0

  Return true if the platform is a SYCL host platform.
- virtual string_class get_info_string (info::platform param) const =0

  Query the platform for OpenCL string info::platform info.
- virtual vector_class< string_class > get_extension_strings () const =0
- virtual bool has_extension (const string_class &extension) const =0

  Specify whether a specific extension is supported on the platform.
- virtual vector_class< device > get_devices (const device_selector &device_selector) const =0

  Get all the available devices for this platform.
- virtual ~platform ()

9.10.2.18.1 Constructor & Destructor Documentation

9.10.2.18.1.1 ~platform() `virtual trisycl::detail::platform::~platform ( ) [inline], [virtual]`

Definition at line 64 of file platform.hpp.

```
00064 {}
```

9.10.2.18.2 Member Function Documentation

9.10.2.18.2.1 get() `virtual cl_platform_id trisycl::detail::platform::get ( ) const [pure virtual]`

Return the cl_platform_id of the underlying OpenCL platform.

Implemented in trisycl::detail::host_platform, and trisycl::detail::opencl_platform.

9.10.2.18.2.2 get_boost_compute() `virtual const boost::compute::platform & trisycl::detail::platform::get_boost_compute ( ) const [pure virtual]`

Return the underlying Boost.Compute platform, if any.

Implemented in trisycl::detail::host_platform, and trisycl::detail::opencl_platform.

9.10.2.18.2.3 get_devices() `virtual vector_class< device > trisycl::detail::platform::get_devices (`

```
        const device_selector & device_selector ) const [pure virtual]
```

Get all the available devices for this platform.

Parameters

in	*device_selector*	is used to filter the selection.

Returns

the device list

Implemented in trisycl::detail::host_platform, and trisycl::detail::opencl_platform.

9.10.2.18.2.4 get_extension_strings() `virtual vector_class< string_class > trisycl::detail::platform::get_extension_strings ( ) const [pure virtual]`

Implemented in trisycl::detail::opencl_platform, and trisycl::detail::host_platform.

9.10.2.18.2.5 get_info_string() `virtual string_class trisycl::detail::platform::get_info_string ( info::platform param ) const [pure virtual]`

Query the platform for OpenCL string info::platform info.

Implemented in trisycl::detail::host_platform, and trisycl::detail::opencl_platform.

9.10.2.18.2.6 has_extension() `virtual bool trisycl::detail::platform::has_extension ( const string_class & extension ) const [pure virtual]`

Specify whether a specific extension is supported on the platform.

Implemented in trisycl::detail::host_platform, and trisycl::detail::opencl_platform.

9.10.2.18.2.7 is_host() `virtual bool trisycl::detail::platform::is_host ( ) const [pure virtual]`

Return true if the platform is a SYCL host platform.

Implemented in trisycl::detail::host_platform, and trisycl::detail::opencl_platform.

9.10.2.19 class trisycl::queue

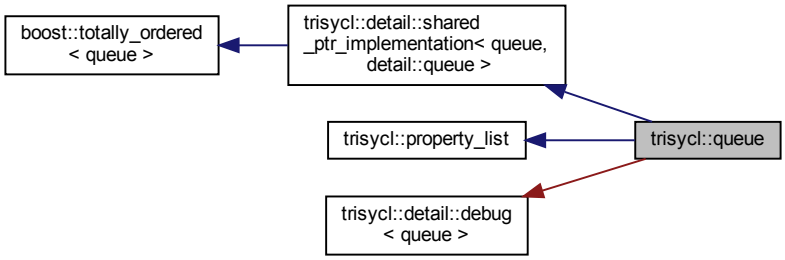
SYCL queue, similar to the OpenCL queue concept.

Todo The implementation is quite minimal for now. :-)

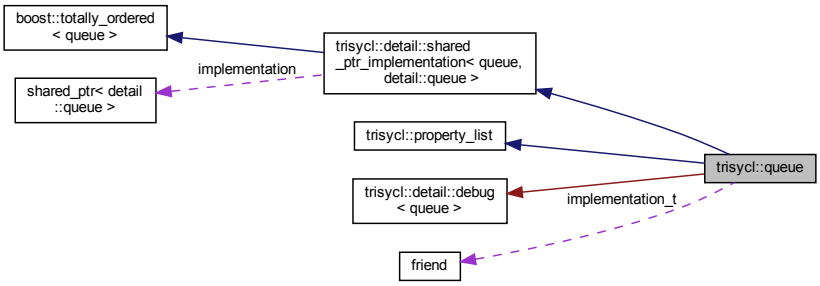
Todo All the queue methods should return a queue& instead of void to it is possible to chain opoerations

Definition at line 52 of file queue.hpp.

Inheritance diagram for trisycl::queue:



Collaboration diagram for trisycl::queue:



Public Member Functions

- queue (const property_list &propList={})

  Default constructor for queue from the default queue selector.
- queue (const async_handler &asyncHandler, const property_list &propList={})

  This constructor creates a SYCL queue from an OpenCL queue.
- queue (const device_selector &deviceSelector, const property_list &propList={})

  Creates a queue for the device provided by the device selector.

- queue (const device_selector &deviceSelector, const async_handler &asyncHandler, const property_list &propList={})

  Creates a queue for the device provided by the device selector.
- queue (const device &d, const async_handler &asyncHandler, const property_list &propList={})

  A queue is created for a SYCL device.
- queue (const device &d, const property_list &propList={})

  A queue is created for a SYCL device.
- queue (const context &syclContext, const device_selector &deviceSelector, const async_handler &async↩ Handler, const property_list &propList={})

  This constructor chooses a device based on the provided device_selector, which needs to be in the given context.
- queue (const context &syclContext, const device_selector &deviceSelector, const property_list &propList={})

  This constructor chooses a device based on the provided device_selector, which needs to be in the given context.
- queue (const cl_command_queue &q, const context &syclContext, const async_handler &ah={})

  This constructor creates a SYCL queue from an OpenCL queue.
- queue (const boost::compute::command_queue &q, async_handler ah=nullptr)

  Construct a queue instance using a boost::compute::command_queue.
- cl_command_queue get () const

  Return the underlying OpenCL command queue after doing a retain.
- boost::compute::command_queue get_boost_compute () const

  Return the underlying Boost.Compute command queue if it is an OpenCL queue.
- context get_context () const

  Return the SYCL queue's context.
- device get_device () const

  Return the SYCL device the queue is associated with.
- bool is_host () const

  Return whether the queue is executing on a SYCL host device.
- void wait ()

  Performs a blocking wait for the completion all enqueued tasks in the queue.
- void wait_and_throw ()

  Perform a blocking wait for the completion all enqueued tasks in the queue.
- void throw_asynchronous ()

  Checks to see if any asynchronous errors have been produced by the queue and if so reports them by passing them to the async_handler passed to the queue on construction.
- template<info::queue param>
  auto get_info () const

  Queries the platform for cl_command_queue info.
- template<typename Handler_Functor >
  event submit (Handler_Functor cgf)

  Submit a command group functor to the queue, in order to be scheduled for execution on the device.
- template<typename Handler_Functor >
  event submit (Handler_Functor cgf, queue &secondaryQueue)

  Submit a command group functor to the queue, in order to be scheduled for execution on the device.
- template<typename propertyT >
  bool has_property () const

  Check if the queue was constructed with the specified property.
- template<typename propertyT >
  propertyT get_property () const

  Return a copy of the property that the queue was constructed with.

Private Types

- using implementation_t = typename queue::shared_ptr_implementation

Private Attributes

- friend implementation_t

Additional Inherited Members

9.10.2.19.1 Member Typedef Documentation

9.10.2.19.1.1 implementation_t `using trisycl::queue::implementation_t = typename queue::shared_ptr_implementati`
`[private]`

Definition at line 61 of file queue.hpp.

9.10.2.19.2 Constructor & Destructor Documentation

9.10.2.19.2.1 queue() [1/10] `trisycl::queue::queue (`
`const property_list & propList = {} ) [explicit]`

Default constructor for queue from the default queue selector.

Returns errors via the SYCL exception class.

9.10.2.19.2.2 queue() [2/10] `trisycl::queue::queue (`
`const async_handler & asyncHandler,`
`const property_list & propList = {} )`

This constructor creates a SYCL queue from an OpenCL queue.

At construction it does a retain on the queue memory object.

Retain a reference to the cl_command_queue object. Caller should release the passed cl_command_queue object when it is no longer needed.

Return synchronous errors regarding the creation of the queue and report asynchronous errors via the async_↩ handler callback function in conjunction with the synchronization and throw methods.

Note that the default case asyncHandler = nullptr is handled by the default constructor.

9.10.2.19.2.3 queue() [3/10] `trisycl::queue::queue (`
`const device_selector & deviceSelector,`
`const property_list & propList = {} ) [inline]`

Creates a queue for the device provided by the device selector.

If no device is selected, an error is reported.

Return synchronous errors regarding the creation of the queue and report asynchronous errors via the `async_↩ handler` callback function if and only if there is an `async_handler` provided.

Definition at line 106 of file queue.hpp.

```
00107                                       {})
00108     // Just create the queue from the selected device
00109     :  queue { device { deviceSelector }, nullptr, propList } { }
```

9.10.2.19.2.4 queue() [4/10] trisycl::queue::queue (

```
const device_selector & deviceSelector,
const async_handler & asyncHandler,
const property_list & propList = {} )  [inline]
```

Creates a queue for the device provided by the device selector.

Return synchronous errors regarding the creation of the queue and report asynchronous errors via the `async_handler` callback function if and only if there is an `async_handler` provided.

Definition at line 117 of file queue.hpp.

```
00119                                       {})
00120     // Just create the queue from the selected device
00121     :  queue { device { deviceSelector }, asyncHandler, propList } { }
```

9.10.2.19.2.5 queue() [5/10] trisycl::queue::queue (

```
const device & d,
const async_handler & asyncHandler,
const property_list & propList = {} )  [inline]
```

A queue is created for a SYCL device.

Return asynchronous errors via the `async_handler` callback function.

Definition at line 128 of file queue.hpp.

```
00130                                       {}) :  implementation_t {
00131 #ifdef TRISYCL_OPENCL
00132     d.is_host()
00133       ?  std::shared_ptr<detail::queue>{ new detail::host_queue }
00134       :  detail::opencl_queue::instance(d)
00135 #else
00136     new detail::host_queue
00137 #endif
00138   }, property_list { propList } { }
```

9.10.2.19.2.6 queue() [6/10] trisycl::queue::queue (

```
const device & d,
const property_list & propList = {} )  [inline]
```

A queue is created for a SYCL device.

Definition at line 143 of file queue.hpp.

```
00144                                       {})
00145     :  queue { d, {}, propList } {}
```

9.10.2.19.2.7 queue() [7/10] trisycl::queue::queue (

```
const context & syclContext,
const device_selector & deviceSelector,
const async_handler & asyncHandler,
const property_list & propList = {} )  [inline]
```

This constructor chooses a device based on the provided device_selector, which needs to be in the given context.

If no device is selected, an error is reported.

Return synchronous errors regarding the creation of the queue.

If and only if there is an asyncHandler provided, it reports asynchronous errors via the async_handler callback function in conjunction with the synchronization and throw methods.

Definition at line 158 of file queue.hpp.

```
00161                                  {})
00162     :  property_list { propList } {
00163     device d = deviceSelector.select_device();
00164     vector_class<device> ctx_devs = syclContext.get_devices();
00165     if (std::find(ctx_devs.begin(), ctx_devs.end(), d) == ctx_devs.end())
00166       throw trisycl::invalid_object_error("Device doesn't belong to context\n");
00167     implementation =
00168 #ifdef TRISYCL_OPENCL
00169       d.is_host() ?  std::shared_ptr<detail::queue>{ new detail::host_queue }
00170     :  detail::opencl_queue::instance(d);
00171 #else
00172     std::shared_ptr<detail::queue>{ new detail::host_queue };
00173 #endif
00174   }
```

9.10.2.19.2.8 queue() [8/10] trisycl::queue::queue (

```
const context & syclContext,
const device_selector & deviceSelector,
const property_list & propList = {} )  [inline]
```

This constructor chooses a device based on the provided device_selector, which needs to be in the given context.

If no device is selected, an error is reported.

Return synchronous errors regarding the creation of the queue.

Definition at line 183 of file queue.hpp.

```
00185                                  {})
00186     :  queue { syclContext, deviceSelector, {}, propList } {}
```

9.10.2.19.2.9 queue() [9/10] trisycl::queue::queue (

```
const cl_command_queue & q,
const context & syclContext,
const async_handler & ah = {} )  [inline]
```

This constructor creates a SYCL queue from an OpenCL queue.

At construction it does a retain on the queue memory object.

Return synchronous errors regarding the creation of the queue. If and only if there is an async_handler provided, it reports asynchronous errors via the async_handler callback function in conjunction with the synchronization and throw methods.

Definition at line 198 of file queue.hpp.

```
00199                               {})
00200     :  queue { boost::compute::command_queue { q }, ah } {}
```

9.10.2.19.2.10 queue() [10/10] trisycl::queue::queue (const boost::compute::command_queue & q, async_handler ah = nullptr) [inline]

Construct a queue instance using a boost::compute::command_queue.

This is a triSYCL extension for boost::compute interoperation.

Return synchronous errors via the SYCL exception class.

Todo Deal with handler

Definition at line 211 of file queue.hpp.

```
00212     :  implementation_t { detail::opencl_queue::instance(q) }, property_list {} {}
```

9.10.2.19.3 Member Function Documentation

9.10.2.19.3.1 get() cl_command_queue trisycl::queue::get ( ) const [inline]

Return the underlying OpenCL command queue after doing a retain.

This memory object is expected to be released by the developer.

Retain a reference to the returned cl_command_queue object.

Caller should release it when finished.

If the queue is a SYCL host queue then an exception is thrown.

Definition at line 227 of file queue.hpp.

```
00227                                {
00228     return implementation->get();
00229   }
```

References trisycl::detail::shared_ptr_implementation< queue, detail::queue >::implementation.

9.10.2.19.3.2 get_boost_compute() boost::compute::command_queue trisycl::queue::get_boost_↩ compute ( ) const [inline]

Return the underlying Boost.Compute command queue if it is an OpenCL queue.

This is a triSYCL extension

Definition at line 237 of file queue.hpp.

```
00237                                                {
00238     return implementation->get_boost_compute();
00239   }
```

References trisycl::detail::shared_ptr_implementation< queue, detail::queue >::implementation.

9.10.2.19.3.3 get_context() context trisycl::queue::get_context ( ) const [inline]

Return the SYCL queue's context.

Report errors using SYCL exception classes.

Definition at line 247 of file queue.hpp.

```
00247                                {
00248     return implementation->get_context();
00249   }
```

References trisycl::detail::shared_ptr_implementation< queue, detail::queue >::implementation.

9.10.2.19.3.4 get_device() device trisycl::queue::get_device ( ) const [inline]

Return the SYCL device the queue is associated with.

Report errors using SYCL exception classes.

Definition at line 256 of file queue.hpp.

```
00256                               {
00257     return implementation->get_device();
00258   }
```

References trisycl::detail::shared_ptr_implementation< queue, detail::queue >::implementation.

9.10.2.19.3.5 get_info() template<info::queue param> auto trisycl::queue::get_info ( ) const [inline]

Queries the platform for cl_command_queue info.

9.10.2.19.3.6 get_property() template<typename propertyT > propertyT trisycl::queue::get_property ( ) const [inline]

Return a copy of the property that the queue was constructed with.

Definition at line 355 of file queue.hpp.

```
00355                                    {
00356     return property_list::get_property<propertyT>();
00357   }
```

9.10.2.19.3.7 has_property() template<typename propertyT > bool trisycl::queue::has_property ( ) const [inline]

Check if the queue was constructed with the specified property.

Definition at line 347 of file queue.hpp.

```
00347                                   {
00348     return property_list::has_property<propertyT>();
00349   }
```

9.10.2.19.3.8 is_host() `bool trisycl::queue::is_host ( ) const [inline]`

Return whether the queue is executing on a SYCL host device.

Definition at line 262 of file queue.hpp.

```
00262                          {
00263     return implementation->is_host();
00264   }
```

References trisycl::detail::shared_ptr_implementation< queue, detail::queue >::implementation.

9.10.2.19.3.9 submit() [1/2] `template<typename Handler_Functor >`

```
event trisycl::queue::submit (
             Handler_Functor cgf ) [inline]
```

Submit a command group functor to the queue, in order to be scheduled for execution on the device.

Use an explicit functor parameter taking a handler& so we can use "auto" in submit() lambda parameter.

Todo Add in the spec an implicit conversion of event to queue& so it is possible to chain operations on the queue

Definition at line 320 of file queue.hpp.

```
00320                                  {
00321     handler command_group_handler { implementation };
00322     cgf(command_group_handler);
00323     return {};
00324   }
```

References trisycl::detail::shared_ptr_implementation< queue, detail::queue >::implementation.

Referenced by submit().

Here is the caller graph for this function:



9.10.2.19.3.10 submit() [2/2] template<typename Handler_Functor >

```
event trisycl::queue::submit (
            Handler_Functor cgf,
            queue & secondaryQueue )  [inline]
```

Submit a command group functor to the queue, in order to be scheduled for execution on the device.

On kernel error, this command group functor, then it is scheduled for execution on the secondary queue.

Return a command group functor event, which is corresponds to the queue the command group functor is being enqueued on.

Definition at line 337 of file queue.hpp.

```
00337                                                    {
00338     TRISYCL_UNIMPL;
00339     // Since it is not implemented, always submit on the main queue
00340     return submit(cgf);
00341   }
```

References submit(), and TRISYCL_UNIMPL.

Here is the call graph for this function:



9.10.2.19.3.11 throw_asynchronous() void trisycl::queue::throw_asynchronous ( ) [inline]

Checks to see if any asynchronous errors have been produced by the queue and if so reports them by passing them to the async_handler passed to the queue on construction.

If no async_handler was provided then asynchronous exceptions will be lost.

Definition at line 301 of file queue.hpp.

```
00301                                {
00302     TRISYCL_UNIMPL;
00303   }
```

References TRISYCL_UNIMPL.

9.10.2.19.3.12 wait() `void trisycl::queue::wait ( ) [inline]`

Performs a blocking wait for the completion all enqueued tasks in the queue.

Synchronous errors will be reported through SYCL exceptions.

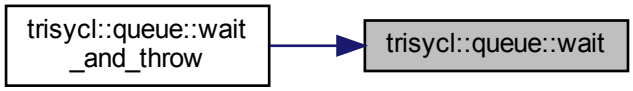
Definition at line 272 of file queue.hpp.

```
00272             {
00273     implementation->wait_for_kernel_execution();
00274   }
```

References trisycl::detail::shared_ptr_implementation< queue, detail::queue >::implementation.

Referenced by wait_and_throw().

Here is the caller graph for this function:



9.10.2.19.3.13 wait_and_throw() `void trisycl::queue::wait_and_throw ( ) [inline]`

Perform a blocking wait for the completion all enqueued tasks in the queue.

Synchronous errors will be reported via SYCL exceptions.

Asynchronous errors will be passed to the async_handler passed to the queue on construction.

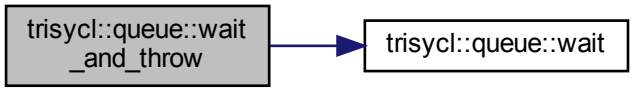
If no async_handler was provided then asynchronous exceptions will be lost.

Definition at line 287 of file queue.hpp.

```
00287                       {
00288     // \todo Implement the throw part of wait_and_throw
00289     wait();
00290     TRISYCL_UNIMPL;
00291   }
```

References TRISYCL_UNIMPL, and wait().

Here is the call graph for this function:



9.10.2.19.4 Member Data Documentation

9.10.2.19.4.1 implementation_t `friend trisycl::queue::implementation_t [private]`

Definition at line 67 of file queue.hpp.

9.10.3 Macro Definition Documentation

9.10.3.1 PLATFORM_GET_INFO_STRING

```
#define PLATFORM_GET_INFO_STRING(
              X )
```

`#include <include/triSYCL/platform.hpp>`

Value:

```
template<>                              \
inline auto platform::get_info<info::platform::X>()const { \
  return implementation->get_info_string(info::platform::X);    \
}
```

Definition at line 166 of file platform.hpp.

9.10.4 Typedef Documentation

9.10.4.1 cpu_selector

```
using trisycl::cpu_selector = typedef device_typename_selector<info::device_type::cpu>
```

`#include <include/triSYCL/device_selector/detail/device_selector_tail.hpp>`

Select devices according to device type info::device::device_type::cpu from all the available devices and heuristics.

If no OpenCL CPU device is found the selector fails.

Definition at line 155 of file device_selector_tail.hpp.

9.10.4.2 default_selector

```
using trisycl::default_selector = typedef device_typename_selector<info::device_type::automatic>
```

```
#include <include/triSYCL/device_selector/detail/device_selector_tail.hpp>
```

Devices selected by heuristics of the system.

If no OpenCL device is found then it defaults to the SYCL host device.

To influence the default device selection, use the Boost.Compute environment variables:

- BOOST_COMPUTE_DEFAULT_DEVICE
- BOOST_COMPUTE_DEFAULT_DEVICE_TYPE
- BOOST_COMPUTE_DEFAULT_PLATFORM
- BOOST_COMPUTE_DEFAULT_VENDOR

Definition at line 137 of file device_selector_tail.hpp.

9.10.4.3 device_exec_capabilities

```
using trisycl::info::device_exec_capabilities = typedef unsigned int
```

```
#include <include/triSYCL/info/device.hpp>
```

Definition at line 180 of file device.hpp.

9.10.4.4 device_fp_config

```
using trisycl::info::device_fp_config = typedef unsigned int
```

```
#include <include/triSYCL/info/device.hpp>
```

Definition at line 179 of file device.hpp.

9.10.4.5 device_queue_properties

```
using trisycl::info::device_queue_properties = typedef unsigned int
```

```
#include <include/triSYCL/info/device.hpp>
```

Definition at line 181 of file device.hpp.

9.10.4.6 gpu_selector

```
using trisycl::gpu_selector = typedef device_typename_selector<info::device_type::gpu>
```

```
#include <include/triSYCL/device_selector/detail/device_selector_tail.hpp>
```

Select devices according to device type info::device::device_type::gpu from all the available OpenCL devices.

If no OpenCL GPU device is found the selector fails.

Select the best GPU, if any.

Definition at line 147 of file device_selector_tail.hpp.

9.10.4.7 host_selector

```
using trisycl::host_selector = typedef device_typename_selector<info::device_type::host>
```

```
#include <include/triSYCL/device_selector/detail/device_selector_tail.hpp>
```

Selects the SYCL host CPU device that does not require an OpenCL runtime.

Definition at line 161 of file device_selector_tail.hpp.

9.10.5 Enumeration Type Documentation

9.10.5.1 context

```
enum class trisycl::info::context :  int  [strong]
```

```
#include <include/triSYCL/info/context.hpp>
```

Context information descriptors.

Todo Should be unsigned int to be consistent with others?

Enumerator

reference_count	
platform	
devices	

Definition at line 25 of file context.hpp.

```
00025                   :  int {
```

```
00026   reference_count,
00027   platform,
00028   devices
00029 };
```

9.10.5.2 device

```
enum class trisycl::info::device :  int  [strong]
```

```
#include <include/triSYCL/info/device.hpp>
```

Device information descriptors.

From specs/latex/headers/deviceInfo.h in the specification

Todo Should be unsigned int?

Enumerator

device_type	
vendor_id	
max_compute_units	
max_work_item_dimensions	
max_work_item_sizes	
max_work_group_size	
preferred_vector_width_char	
preferred_vector_width_short	
preferred_vector_width_int	
preferred_vector_width_long	
preferred_vector_width_float	
preferred_vector_width_double	
preferred_vector_width_half	
native_vector_width_char	
native_vector_width_short	
native_vector_width_int	
native_vector_width_long	
native_vector_width_float	
native_vector_width_double	
native_vector_width_half	
max_clock_frequency	
address_bits	
max_mem_alloc_size	
image_support	
max_read_image_args	
max_write_image_args	
image2d_max_width	
image2d_max_height	
image3d_max_width	
image3d_max_height	

Enumerator

image3d_max_depth	
image_max_buffer_size	
image_max_array_size	
max_samplers	
max_parameter_size	
mem_base_addr_align	
half_fp_config	
single_fp_config	
double_fp_config	
global_mem_cache_type	
global_mem_cache_line_size	
global_mem_cache_size	
global_mem_size	
max_constant_buffer_size	
max_constant_args	
local_mem_type	
local_mem_size	
error_correction_support	
host_unified_memory	
profiling_timer_resolution	
is_endian_little	
is_available	
is_compiler_available	
is_linker_available	
execution_capabilities	
queue_profiling	
built_in_kernels	
platform	
name	
vendor	
driver_version	
profile	
version	
opencl_c_version	
extensions	
printf_buffer_size	
preferred_interop_user_sync	
parent_device	
partition_max_sub_devices	
partition_properties	
partition_affinity_domains	
partition_type_property	
partition_type_affinity_domain	
reference_count	

Definition at line 56 of file device.hpp.

```
00056                  :  int {
00057   device_type,
```

```
00058   vendor_id,
00059   max_compute_units,
00060   max_work_item_dimensions,
00061   max_work_item_sizes,
00062   max_work_group_size,
00063   preferred_vector_width_char,
00064   preferred_vector_width_short,
00065   preferred_vector_width_int,
00066   preferred_vector_width_long,
00067   preferred_vector_width_float,
00068   preferred_vector_width_double,
00069   preferred_vector_width_half,
00070   native_vector_width_char,
00071   native_vector_width_short,
00072   native_vector_width_int,
00073   native_vector_width_long,
00074   native_vector_width_float,
00075   native_vector_width_double,
00076   native_vector_width_half,
00077   max_clock_frequency,
00078   address_bits,
00079   max_mem_alloc_size,
00080   image_support,
00081   max_read_image_args,
00082   max_write_image_args,
00083   image2d_max_width,
00084   image2d_max_height,
00085   image3d_max_width,
00086   image3d_max_height,
00087   image3d_max_depth,
00088   image_max_buffer_size,
00089   image_max_array_size,
00090   max_samplers,
00091   max_parameter_size,
00092   mem_base_addr_align,
00093   half_fp_config,
00094   single_fp_config,
00095   double_fp_config,
00096   global_mem_cache_type,
00097   global_mem_cache_line_size,
00098   global_mem_cache_size,
00099   global_mem_size,
00100   max_constant_buffer_size,
00101   max_constant_args,
00102   local_mem_type,
00103   local_mem_size,
00104   error_correction_support,
00105   host_unified_memory,
00106   profiling_timer_resolution,
00107   is_endian_little,
00108   is_available,
00109   is_compiler_available,
00110   is_linker_available,
00111   execution_capabilities,
00112   queue_profiling,
00113   built_in_kernels,
00114   platform,
00115   name,
00116   vendor,
00117   driver_version,
00118   profile,
00119   version,
00120   opencl_c_version,
00121   extensions,
00122   printf_buffer_size,
00123   preferred_interop_user_sync,
00124   parent_device,
00125   partition_max_sub_devices,
00126   partition_properties,
00127   partition_affinity_domains,
00128   partition_type_property,
00129   partition_type_affinity_domain,
00130   reference_count
00131 };
```

9.10.5.3 device_type

```
enum class trisycl::info::device_type :  unsigned int  [strong]
```

```
#include <include/triSYCL/info/device.hpp>
```

Type of devices.

To be used either to define a device type or to select more broadly a kind of device

Todo To be moved in the specification from platform to device

Todo Add opencl to the specification

Todo there is no accelerator_selector and custom_accelerator

Enumerator

cpu	
gpu	
accelerator	
custom	
automatic	
host	
opencl	
all	

Definition at line 38 of file device.hpp.

```
00038                         :  unsigned int {
00039   cpu,
00040   gpu,
00041   accelerator,
00042   custom,
00043   automatic,
00044   host,
00045   opencl,
00046   all
00047 };
```

9.10.5.4 execution_capability

```
enum class trisycl::info::execution_capability :  unsigned int  [strong]
```

```
#include <include/triSYCL/info/device.hpp>
```

Enumerator

exec_kernel	
exec_native_kernel	

Definition at line 173 of file device.hpp.

```
00173                                 :  unsigned int {
00174   exec_kernel,
00175   exec_native_kernel
00176 };
```

9.10.5.5 fp_config

```
enum class trisycl::info::fp_config :  int  [strong]
```

```
#include <include/triSYCL/info/device.hpp>
```

Enumerator

denorm	
inf_nan	
round_to_nearest	
round_to_zero	
round_to_inf	
fma	
correctly_rounded_divide_sqrt	
soft_float	

Definition at line 156 of file device.hpp.

```
00156                       :  int {
00157   denorm,
00158   inf_nan,
00159   round_to_nearest,
00160   round_to_zero,
00161   round_to_inf,
00162   fma,
00163   correctly_rounded_divide_sqrt,
00164   soft_float
00165 };
```

9.10.5.6 global_mem_cache_type

```
enum class trisycl::info::global_mem_cache_type :  int  [strong]
```

```
#include <include/triSYCL/info/device.hpp>
```

Enumerator

none	
read_only	
write_only	

Definition at line 167 of file device.hpp.

```
00167                                    :  int {
00168   none,
00169   read_only,
00170   write_only
00171 };
```

9.10.5.7 local_mem_type

```
enum class trisycl::info::local_mem_type :  int  [strong]
```

```
#include <include/triSYCL/info/device.hpp>
```

Enumerator

none	
local	
global	

Definition at line 150 of file device.hpp.

```
00150                           : int {
00151   none,
00152   local,
00153   global
00154 };
```

9.10.5.8 partition_affinity_domain

```
enum class trisycl::info::partition_affinity_domain : int [strong]
```

```
#include <include/triSYCL/info/device.hpp>
```

Enumerator

not_applicable	
numa	
L4_cache	
L3_cache	
L2_cache	
L1_cache	
next_partitionable	

Definition at line 140 of file device.hpp.

```
00140                                     : int {
00141   not_applicable,
00142   numa,
00143   L4_cache,
00144   L3_cache,
00145   L2_cache,
00146   L1_cache,
00147   next_partitionable
00148 };
```

9.10.5.9 partition_property

```
enum class trisycl::info::partition_property : int [strong]
```

```
#include <include/triSYCL/info/device.hpp>
```

Enumerator

no_partition	
partition_equally	
partition_by_counts	
partition_by_affinity_domain	

Definition at line 133 of file device.hpp.

```
00133                                : int {
00134   no_partition,
00135   partition_equally,
00136   partition_by_counts,
00137   partition_by_affinity_domain,
00138 };
```

9.10.5.10 platform

```
enum class trisycl::info::platform : unsigned int [strong]
```

```
#include <include/triSYCL/info/platform.hpp>
```

Platform information descriptors.

A SYCL platform can be queried for all of the following information using the get_info function.

In this implementation, the values are mapped to OpenCL values to avoid further remapping later when OpenCL is used

Enumerator

TRISYCL_OPENCL_ONLY	Returns the profile name (as a string_class) supported by the implementation. Can be either FULL PROFILE or EMBEDDED PROFILE.
TRISYCL_OPENCL_ONLY	Returns the OpenCL software driver version string in the form major number.minor number (as a string_class)
TRISYCL_OPENCL_ONLY	Returns the name of the platform (as a string_class)
TRISYCL_OPENCL_ONLY	Returns the string provided by the platform vendor (as a string_class)
TRISYCL_OPENCL_ONLY	Returns a space-separated list of extension names supported by the platform (as a string_class)

Definition at line 30 of file platform.hpp.

```
00030                        : unsigned int {
00031 /** Returns the profile name (as a string_class) supported by the
00032 implementation.
00033
00034 Can be either FULL PROFILE or EMBEDDED PROFILE.
00035 */
00036   profile TRISYCL_OPENCL_ONLY(= CL_PLATFORM_PROFILE),
00037
00038 /** Returns the OpenCL software driver version string in the form major
00039 number.minor number (as a string_class)
00040 */
00041   version TRISYCL_OPENCL_ONLY(= CL_PLATFORM_VERSION),
00042
00043 /** Returns the name of the platform (as a string_class)
00044 */
00045   name TRISYCL_OPENCL_ONLY(= CL_PLATFORM_NAME),
00046
00047 /** Returns the string provided by the platform vendor (as a string_class)
00048 */
00049   vendor TRISYCL_OPENCL_ONLY(= CL_PLATFORM_VENDOR),
00050
00051 /** Returns a space-separated list of extension names supported by the
00052 platform (as a string_class)
00053 */
00054   extensions TRISYCL_OPENCL_ONLY(= CL_PLATFORM_EXTENSIONS),
00055
00056 #if CL_SYCL_LANGUAGE_VERSION >= 220 && defined(CL_VERSION_2_1)
00057 /** Returns the resolution of the host timer in nanoseconds as used by
00058 clGetDeviceAndHostTimer
00059 */
00060   host_timer_resolution
00061     TRISYCL_OPENCL_ONLY(= CL_PLATFORM_HOST_TIMER_RESOLUTION)
```

```
00062 #endif
00063 };
```

9.10.6 Function Documentation

9.10.6.1 context::get_info< info::context::devices >()

```
template<>
auto trisycl::context::get_info< info::context::devices > ( ) const  [inline]
```

#include <include/triSYCL/context.hpp>

Definition at line 223 of file context.hpp.

```
00223                                                  {
00224   return get_devices();
00225 }
```

9.10.6.2 context::get_info< info::context::platform >()

```
template<>
auto trisycl::context::get_info< info::context::platform > ( ) const  [inline]
```

#include <include/triSYCL/context.hpp>

Definition at line 218 of file context.hpp.

```
00218                                                   {
00219   return get_platform();
00220 }
```

9.10.6.3 context::get_info< info::context::reference_count >()

```
template<>
auto trisycl::context::get_info< info::context::reference_count > ( ) const  [inline]
```

#include <include/triSYCL/context.hpp>

Definition at line 213 of file context.hpp.

```
00213                                                           {
00214   return implementation->get_reference_count();
00215 }
```

9.10.6.4 get_devices() [1/2]

```
vector_class<::trisycl::device > trisycl::detail::host_platform::get_devices (
            const device_selector & device_selector ) const  [inline], [override], [virtual]
```

#include <include/triSYCL/platform/detail/host_platform.hpp>

Get all the available devices for the host platform.

Get all the available devices for this platform.

Parameters

in	*device_type*	is the device type to filter the selection or `info::device_type::all` by default to return all the devices

Returns

the device list

If `get_devices` is called with the host platform and the right device type, returns the host_device.

Implements trisycl::detail::platform.

Definition at line 29 of file host_platform_tail.hpp.

```
00029                                                                          {
00030 /** If \c get_devices is called with the host platform
00031 and the right device type, returns the host_device.
00032 */
00033   if (device_selector(::trisycl::device {}) > 0)
00034     // Return 1 default device, i.e.  the host device
00035     return { {} };
00036   else
00037     // No matching device
00038     return {};
00039 }
```

9.10.6.5 get_devices() [2/2]

```
vector_class<::trisycl::device > trisycl::detail::opencl_platform::get_devices (
            const device_selector & device_selector ) const  [inline], [override], [virtual]
```

`#include <include/triSYCL/platform/detail/opencl_platform.hpp>`

Get all the available devices for this OpenCL platform.

Returns a vector class containing all SYCL devices associated with this OpenCL platform.

Parameters

in	*device_type*	is the device type to filter the selection or `info::device_type::all` by default to return all the devices

Returns

the device list

Implements trisycl::detail::platform.

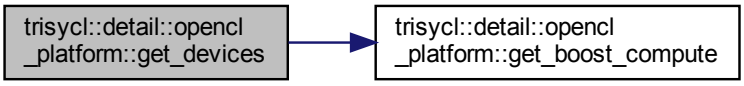
Definition at line 30 of file opencl_platform_tail.hpp.

```
00030                                                                          {
00031   vector_class<::trisycl::device> devices;
00032   // Add the desired OpenCL devices
00033   for (const auto &d :  get_boost_compute().devices()) {
00034     // Get the SYCL device from the Boost Compute device
00035     ::trisycl::device sycl_dev { d };
00036     /* Return the devices with the good criterion according to the selector.
```

```
00037 By calling devices on the \c boost::compute::platform we know that
00038 we iterate only over the device belonging to the current platform,
00039 */
00040     if (device_selector(sycl_dev) > 0)
00041       devices.push_back(sycl_dev);
00042   }
00043
00044   return devices;
00045 }
```

References trisycl::detail::opencl_platform::get_boost_compute().

Here is the call graph for this function:



9.10.6.6 platform::get_info< info::platform::extensions >()

```
template<>
auto trisycl::platform::get_info< info::platform::extensions > ( ) const  [inline]
```

```
#include <include/triSYCL/platform.hpp>
```

Definition at line 162 of file platform.hpp.

```
00162                                                    {
00163   return implementation->get_extension_strings();
00164 }
```

9.10.6.7 queue::get_info< info::queue::context >()

```
template<>
auto trisycl::queue::get_info< info::queue::context > ( ) const  [inline]
```

```
#include <include/triSYCL/queue.hpp>
```

Definition at line 366 of file queue.hpp.

```
00366                                                {
00367   return implementation->get_context();
00368 }
```

9.10.6.8 queue::get_info< info::queue::device >()

```
template<>
auto trisycl::queue::get_info< info::queue::device > ( ) const  [inline]
```

```
#include <include/triSYCL/queue.hpp>
```

Definition at line 361 of file queue.hpp.

```
00361                                                {
00362   return implementation->get_device();
00363 }
```

9.10.6.9 queue::get_info< info::queue::reference_count >()

```
template<>
auto trisycl::queue::get_info< info::queue::reference_count > ( ) const  [inline]
```

```
#include <include/triSYCL/queue.hpp>
```

Definition at line 371 of file queue.hpp.

```
00371                                                        {
00372   return trisycl::cl_uint {0};
00373 }
```

9.10.6.10 select_device() [1/2]

```
device trisycl::device_selector::select_device ( ) const  [inline]
```

```
#include <include/triSYCL/device_selector.hpp>
```

Returns a selected device using the functor operator defined in sub-classes operator()(const device &dev)

Todo Remove this from specification

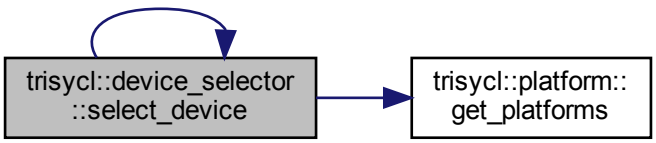
Definition at line 46 of file device_selector_tail.hpp.

```
00046                                                {
00047   return select_device(platform::get_platforms());
00048 }
```

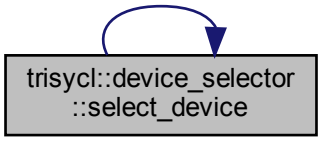
References trisycl::platform::get_platforms(), and trisycl::device_selector::select_device().

Referenced by trisycl::device_selector::select_device().

Here is the call graph for this function:



Here is the caller graph for this function:



9.10.6.11 select_device() [2/2]

```
device trisycl::device_selector::select_device (
            vector_class< platform > platforms ) const  [inline], [private]
```

`#include <include/triSYCL/device_selector.hpp>`

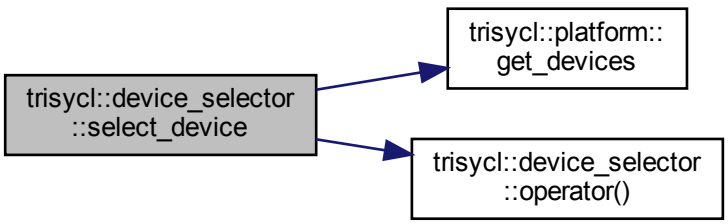
Definition at line 25 of file device_selector_tail.hpp.

```
00025                                                                          {
00026   int best_id = -1;
00027   int best_score = -1;
00028   device best_device;
00029
00030   for (auto &platform :  platforms) {
00031     for (auto &dev :  platform.get_devices()) {
00032       int score = operator()(dev);
00033       if (score > best_score) {
00034         best_id = 1;
00035         best_device = dev;
00036         best_score = score;
00037       }
00038     }
00039   }
00040   if (best_id == -1)
00041     throw ::trisycl::runtime_error("Could not find device");
00042
00043   return best_device;
00044 }
```

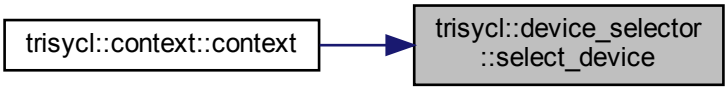
References trisycl::platform::get_devices(), and trisycl::device_selector::operator()().

Referenced by trisycl::context::context().

Here is the call graph for this function:



Here is the caller graph for this function:



9.10.6.12 TRISYCL_INFO_PARAM_TRAITS() [1/7]

```
trisycl::info::TRISYCL_INFO_PARAM_TRAITS (
            info::device::image2d_max_height ,
            size_t  )
```

```
#include <include/triSYCL/info/device.hpp>
```

9.10.6.13 TRISYCL_INFO_PARAM_TRAITS() [2/7]

```
trisycl::info::TRISYCL_INFO_PARAM_TRAITS (
            info::device::image2d_max_width ,
            size_t  )
```

```
#include <include/triSYCL/info/device.hpp>
```

Query the return type for get_info() on context stuff.

Todo To be implemented, return always void.

9.10.6.14 TRISYCL_INFO_PARAM_TRAITS() [3/7]

```
trisycl::info::TRISYCL_INFO_PARAM_TRAITS (
            info::device::image3d_max_depth ,
            size_t  )
```

```
#include <include/triSYCL/info/device.hpp>
```

9.10.6.15 TRISYCL_INFO_PARAM_TRAITS() [4/7]

```
trisycl::info::TRISYCL_INFO_PARAM_TRAITS (
            info::device::image3d_max_height ,
            size_t  )
```

```
#include <include/triSYCL/info/device.hpp>
```

9.10.6.16 TRISYCL_INFO_PARAM_TRAITS() [5/7]

```
trisycl::info::TRISYCL_INFO_PARAM_TRAITS (
            info::device::image3d_max_width ,
            size_t  )
```

```
#include <include/triSYCL/info/device.hpp>
```

9.10.6.17 TRISYCL_INFO_PARAM_TRAITS() [6/7]

```
trisycl::info::TRISYCL_INFO_PARAM_TRAITS (
            info::device::image_max_array_size ,
            size_t  )
```

```
#include <include/triSYCL/info/device.hpp>
```

9.10.6.18 TRISYCL_INFO_PARAM_TRAITS() [7/7]

```
trisycl::info::TRISYCL_INFO_PARAM_TRAITS (
            info::device::image_max_buffer_size ,
            size_t  )
```

```
#include <include/triSYCL/info/device.hpp>
```

9.10.7 Variable Documentation

9.10.7.1 TRISYCL_WEAK_ATTRIB_SUFFIX

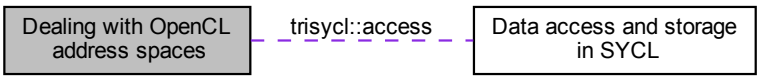
```
TRISYCL_WEAK_ATTRIB_PREFIX detail::cache< cl_command_queue, detail::opencl_queue > opencl_queue::cache
trisycl::detail::TRISYCL_WEAK_ATTRIB_SUFFIX
```

```
#include <include/triSYCL/context/detail/opencl_context.hpp>
```

Definition at line 146 of file opencl_context.hpp.

9.11 Dealing with OpenCL address spaces

Collaboration diagram for Dealing with OpenCL address spaces:



Namespaces

- namespace trisycl::access

  Describe the type of access by kernels.

Classes

- struct trisycl::detail::ocl_type< T, AS >

  Generate a type with some real OpenCL 2 attribute if we are on an OpenCL device. More...
- struct trisycl::detail::ocl_type< T, access::address_space::constant_space >

  Add an attribute for __constant address space. More...
- struct trisycl::detail::ocl_type< T, access::address_space::global_space >

  Add an attribute for __global address space. More...
- struct trisycl::detail::ocl_type< T, access::address_space::local_space >

  Add an attribute for __local address space. More...
- struct trisycl::detail::ocl_type< T, access::address_space::private_space >

  Add an attribute for __private address space. More...
- struct trisycl::detail::address_space_array< T, AS >

  Implementation of an array variable with an OpenCL address space. More...
- struct trisycl::detail::address_space_fundamental< T, AS >

  Implementation of a fundamental type with an OpenCL address space. More...
- struct trisycl::detail::address_space_object< T, AS >

  Implementation of an object type with an OpenCL address space. More...
- struct trisycl::detail::address_space_ptr< T, AS >

  Implementation for an OpenCL address space pointer. More...
- struct trisycl::detail::address_space_base< T, AS >

  Implementation of the base infrastructure to wrap something in an OpenCL address space. More...
- struct trisycl::detail::address_space_variable< T, AS >

  Implementation of a variable with an OpenCL address space. More...

Typedefs

- template<typename T >
  using trisycl::constant = detail::addr_space< T, access::address_space::constant_space >

  Declare a variable to be in the OpenCL constant address space.
- template<typename T >
  using trisycl::constant_ptr = constant< T * >

  Declare a variable to be in the OpenCL constant address space.
- template<typename T >
  using trisycl::generic = detail::addr_space< T, access::address_space::generic_space >

  Declare a variable to be in the OpenCL 2 generic address space.
- template<typename T >
  using trisycl::global = detail::addr_space< T, access::address_space::global_space >

  Declare a variable to be in the OpenCL global address space.
- template<typename T >
  using trisycl::global_ptr = global< T * >

  Declare a variable to be in the OpenCL global address space.
- template<typename T >
  using trisycl::local = detail::addr_space< T, access::address_space::local_space >

  Declare a variable to be in the OpenCL local address space.
- template<typename T >
  using trisycl::local_ptr = local< T * >

  Declare a variable to be in the OpenCL local address space.
- template<typename T >
  using trisycl::priv = detail::addr_space< T, access::address_space::private_space >

  Declare a variable to be in the OpenCL private address space.
- template<typename T >
  using trisycl::private_ptr = priv< T * >

  Declare a variable to be in the OpenCL private address space.
- template<typename Pointer , access::address_space AS>
  using trisycl::multi_ptr = detail::address_space_ptr< Pointer, AS >

  A pointer that can be statically associated to any address-space.
- template<typename T , access::address_space AS>
  using trisycl::detail::addr_space = typename std::conditional< std::is_pointer< T >::value, address_space_ptr< T, AS >, typename std::conditional< std::is_class< T >::value, address_space_object< T, AS >, typename std::conditional< std::is_array< T >::value, address_space_array< T, AS >, address_space_fundamental< T, AS > >::type >::type >::type

  Dispatch the address space implementation according to the requested type.

Functions

- template<typename T , access::address_space AS>
  multi_ptr< T, AS > trisycl::make_ptr (multi_ptr< T, AS > pointer)

  Construct a trisycl::multi_ptr<> with the right type.
- template<typename T , access::address_space AS>
  multi_ptr< T, AS > trisycl::make_ptr (T *pointer)

  Construct a trisycl::multi_ptr<> with the right type.

9.11.1 Detailed Description

9.11.2 Class Documentation

9.11.2.1 struct trisycl::detail::ocl_type

template<typename T, access::address_space AS>
struct trisycl::detail::ocl_type< T, AS >

Generate a type with some real OpenCL 2 attribute if we are on an OpenCL device.

In the general case, do not add any OpenCL address space qualifier

Definition at line 63 of file address_space.hpp.

Public Types

- using type = T

9.11.2.1.1 Member Typedef Documentation

9.11.2.1.1.1 type template<typename T , access::address_space AS>

```
using trisycl::detail::ocl_type< T, AS >::type = T
```

Definition at line 64 of file address_space.hpp.

9.11.2.2 struct trisycl::detail::ocl_type< T, access::address_space::constant_space >

template<typename T>
struct trisycl::detail::ocl_type< T, access::address_space::constant_space >

Add an attribute for __constant address space.

Definition at line 69 of file address_space.hpp.

Public Types

- using type = TRISYCL_CONSTANT_AS T

9.11.2.2.1 Member Typedef Documentation

9.11.2.2.1.1 type template<typename T >

```
using trisycl::detail::ocl_type< T, access::address_space::constant_space >::type = TRISYCL_CONSTANT_AS
T
```

Definition at line 70 of file address_space.hpp.

9.11.2.3 struct trisycl::detail::ocl_type< T, access::address_space::global_space >

template<typename T>
struct trisycl::detail::ocl_type< T, access::address_space::global_space >

Add an attribute for __global address space.

Definition at line 83 of file address_space.hpp.

Public Types

- using type = TRISYCL_GLOBAL_AS T

9.11.2.3.1 Member Typedef Documentation

9.11.2.3.1.1 type template<typename T >

```
using trisycl::detail::ocl_type< T, access::address_space::global_space >::type = TRISYCL_GLOBAL_AS
T
```

Definition at line 84 of file address_space.hpp.

9.11.2.4 struct trisycl::detail::ocl_type< T, access::address_space::local_space >

template<typename T>
struct trisycl::detail::ocl_type< T, access::address_space::local_space >

Add an attribute for __local address space.

Definition at line 89 of file address_space.hpp.

Public Types

- using type = TRISYCL_LOCAL_AS T

9.11.2.4.1 Member Typedef Documentation

9.11.2.4.1.1 type template<typename T >

```
using trisycl::detail::ocl_type< T, access::address_space::local_space >::type = TRISYCL_LOCAL_AS
T
```

Definition at line 90 of file address_space.hpp.

9.11.2.5 struct trisycl::detail::ocl_type< T, access::address_space::private_space >

template<typename T>
struct trisycl::detail::ocl_type< T, access::address_space::private_space >

Add an attribute for __private address space.

Definition at line 95 of file address_space.hpp.

Public Types

- using type = TRISYCL_PRIVATE_AS T

9.11.2.5.1 Member Typedef Documentation

9.11.2.5.1.1 type template<typename T >

```
using trisycl::detail::ocl_type< T, access::address_space::private_space >::type = TRISYCL_PRIVATE_AS
T
```

Definition at line 96 of file address_space.hpp.

9.11.2.6 struct trisycl::detail::address_space_array

template<typename T, access::address_space AS>
struct trisycl::detail::address_space_array< T, AS >

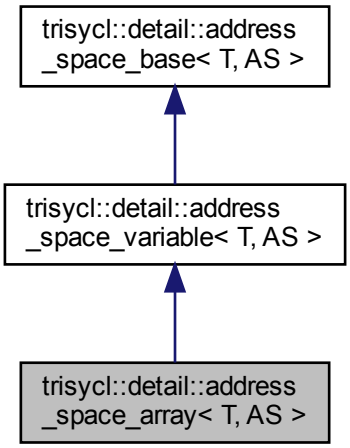
Implementation of an array variable with an OpenCL address space.

Parameters

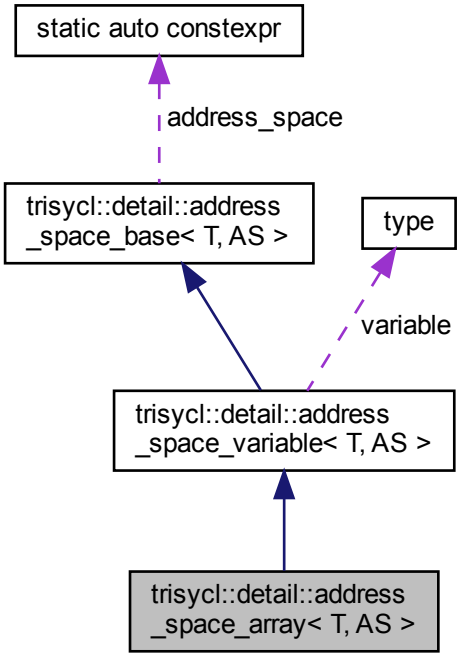
T	is the type of the basic object to be created
AS	is the address space to place the object into

Definition at line 317 of file address_space.hpp.

Inheritance diagram for trisycl::detail::address_space_array< T, AS >:



Collaboration diagram for trisycl::detail::address_space_array< T, AS >:



Public Types

- using super = address_space_variable< T, AS >

Keep track of the base class as a short-cut.

Public Member Functions

- address_space_array (const T &array)
  Allow to create an address space array from an array.
- address_space_array (std::initializer_list< std::remove_extent_t< T > > list)
  Allow to create an address space array from an initializer list.

Additional Inherited Members

9.11.2.6.1 Member Typedef Documentation

9.11.2.6.1.1 super `template<typename T , access::address_space AS>`

```
using trisycl::detail::address_space_array< T, AS >::super = address_space_variable<T, AS>
```

Keep track of the base class as a short-cut.

Definition at line 319 of file address_space.hpp.

9.11.2.6.2 Constructor & Destructor Documentation

9.11.2.6.2.1 address_space_array() [1/2] `template<typename T , access::address_space AS>`

```
trisycl::detail::address_space_array< T, AS >::address_space_array (
            const T & array ) [inline]
```

Allow to create an address space array from an array.

Definition at line 327 of file address_space.hpp.

```
00327                                       {
00328     std::copy(std::begin(array), std::end(array), std::begin(super::variable));
00329   };
```

References trisycl::detail::address_space_variable< T, AS >::variable.

9.11.2.6.2.2 address_space_array() [2/2] `template<typename T , access::address_space AS>`

```
trisycl::detail::address_space_array< T, AS >::address_space_array (
            std::initializer_list< std::remove_extent_t< T > > list ) [inline]
```

Allow to create an address space array from an initializer list.

Todo Extend to more than 1 dimension

Definition at line 336 of file address_space.hpp.

```
00336                                                                       {
00337     std::copy(std::begin(list), std::end(list), std::begin(super::variable));
00338   };
```

References trisycl::detail::address_space_variable< T, AS >::variable.

9.11.2.7 struct trisycl::detail::address_space_fundamental

template<typename T, access::address_space AS>
struct trisycl::detail::address_space_fundamental< T, AS >

Implementation of a fundamental type with an OpenCL address space.

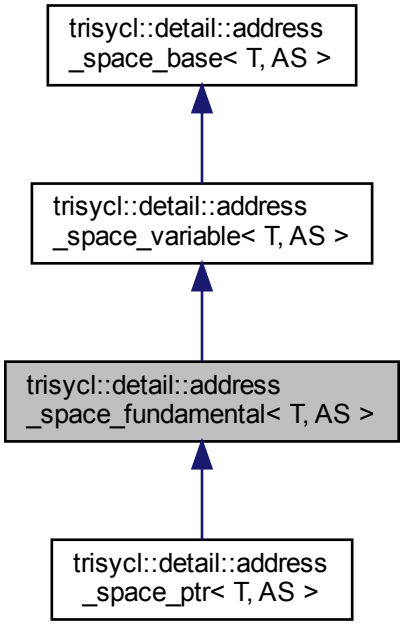
Parameters

T	is the type of the basic object to be created
AS	is the address space to place the object into

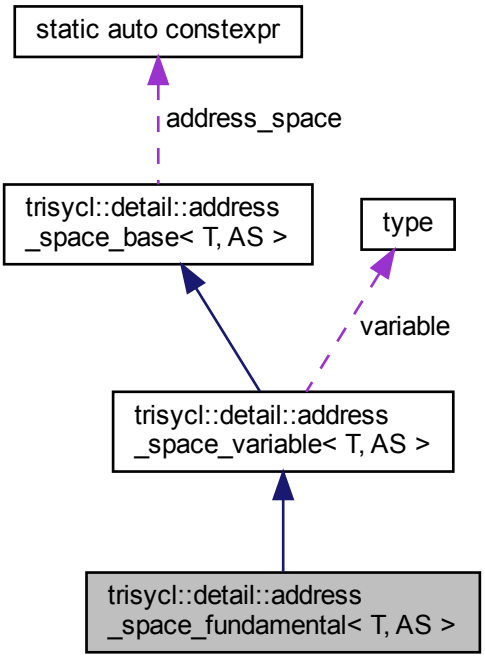
Todo Verify/improve to deal with const/volatile?

Definition at line 225 of file address_space.hpp.

Inheritance diagram for trisycl::detail::address_space_fundamental< T, AS >:



Collaboration diagram for trisycl::detail::address_space_fundamental< T, AS >:



Public Types

- using super = address_space_variable< T, AS >

  Keep track of the base class as a short-cut.

Public Member Functions

- address_space_fundamental ()=default

  Also request for the default constructors that have been disabled by the declaration of another constructor.
- template<typename SomeType , trisycl::access::address_space SomeAS>
  address_space_fundamental (address_space_fundamental< SomeType, SomeAS > &v)

  Allow for example assignment of a global<float> to a priv<double> for example.

Additional Inherited Members

9.11.2.7.1 Member Typedef Documentation

9.11.2.7.1.1 super `template<typename T , access::address_space AS>`

```
using trisycl::detail::address_space_fundamental< T, AS >::super = address_space_variable<T,
AS>
```

Keep track of the base class as a short-cut.

Definition at line 227 of file address_space.hpp.

9.11.2.7.2 Constructor & Destructor Documentation

9.11.2.7.2.1 address_space_fundamental() [1/2] template<typename T , access::address_space AS> trisycl::detail::address_space_fundamental< T, AS >::address_space_fundamental ( ) [default]

Also request for the default constructors that have been disabled by the declaration of another constructor.

This ensures for example that we can write
```
generic<float *> q;
```

without initialization.

9.11.2.7.2.2 address_space_fundamental() [2/2] template<typename T , access::address_space AS> template<typename SomeType , trisycl::access::address_space SomeAS> trisycl::detail::address_space_fundamental< T, AS >::address_space_fundamental (address_space_fundamental< SomeType, SomeAS > & v) [inline]

Allow for example assignment of a global<float> to a priv<double> for example.

Since it needs 2 implicit conversions, it does not work with the conversion operators already define, so add 1 more explicit conversion here so that the remaining implicit conversion can be found by the compiler.

Strangely
```
template <typename SomeType, address_space SomeAS>
address_space_base(addr_space<SomeType, SomeAS>& v)
:  variable(SomeType(v)) { }
```

cannot be used here because SomeType cannot be inferred. So use address_space_base<> instead

Need to think further about it...

Definition at line 265 of file address_space.hpp.
```
00266   {
00267     /* Strangely I cannot have it working in the initializer instead, for
00268 some cases */
00269     super::variable = SomeType(v);
00270   }
```

References trisycl::detail::address_space_variable< T, AS >::variable.

9.11.2.8 struct trisycl::detail::address_space_object

template<typename T, access::address_space AS>
struct trisycl::detail::address_space_object< T, AS >

Implementation of an object type with an OpenCL address space.

Parameters

T	is the type of the basic object to be created
AS	is the address space to place the object into

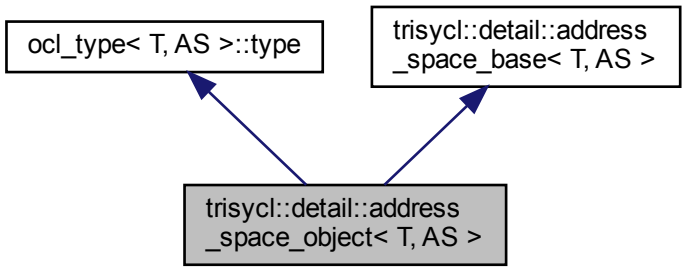
The class implementation is just inheriting of T so that all methods and non-member operators on T work also on address_space_object<T>

Todo Verify/improve to deal with const/volatile?

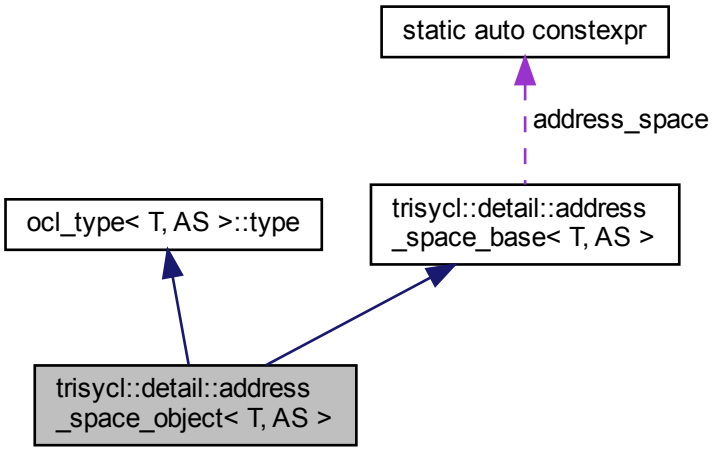
Todo what about T having some final methods?

Definition at line 358 of file address_space.hpp.

Inheritance diagram for trisycl::detail::address_space_object< T, AS >:



Collaboration diagram for trisycl::detail::address_space_object< T, AS >:



Public Types

- using opencl_type = typename ocl_type< T, AS >::type

  Store the base type of the object with OpenCL address space modifier.

Public Member Functions

- address_space_object (T &&v)

  Allow to create an address space version of an object or to convert one.
- operator opencl_type & ()

  Conversion operator to allow a address_space_object<T> to be used as a T so that all the methods of a T and the built-in operators for T can be used on a address_space_object<T> too.

Additional Inherited Members

9.11.2.8.1 Member Typedef Documentation

9.11.2.8.1.1 opencl_type `template<typename T , access::address_space AS>`

```
using trisycl::detail::address_space_object< T, AS >::opencl_type = typename ocl_type<T,
AS>::type
```

Store the base type of the object with OpenCL address space modifier.

Todo Add to the specification

Definition at line 364 of file address_space.hpp.

9.11.2.8.2 Constructor & Destructor Documentation

9.11.2.8.2.1 address_space_object() `template<typename T , access::address_space AS>`

```
trisycl::detail::address_space_object< T, AS >::address_space_object (
            T && v )  [inline]
```

Allow to create an address space version of an object or to convert one.

Definition at line 375 of file address_space.hpp.

```
00375 :  opencl_type(v) { }
```

9.11.2.8.3 Member Function Documentation

9.11.2.8.3.1 operator opencl_type &() `template<typename T , access::address_space AS>`

```
trisycl::detail::address_space_object< T, AS >::operator opencl_type & ( )  [inline]
```

Conversion operator to allow a address_space_object<T> to be used as a T so that all the methods of a T and the built-in operators for T can be used on a address_space_object<T> too.

Use opencl_type so that if we take the address of it, the address space is kept.

Definition at line 383 of file address_space.hpp.

```
00383 { return *this; }
```

9.11.2.9 struct trisycl::detail::address_space_ptr

template<typename T, access::address_space AS>
struct trisycl::detail::address_space_ptr< T, AS >

Implementation for an OpenCL address space pointer.

Parameters

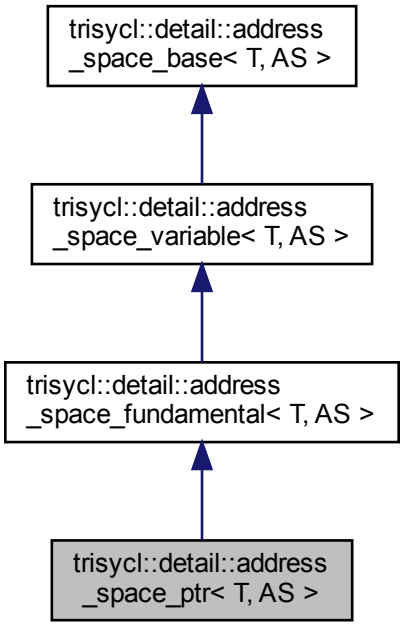
T	is the pointer type

Note that if *T* is not a pointer type, it is an error.

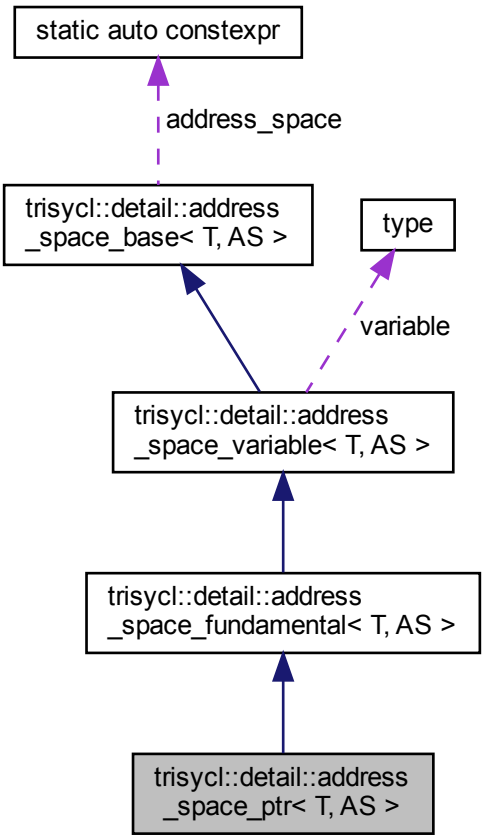
All the address space pointers inherit from it, which makes trivial the implementation of `trisycl::multi_ptr<T, AS>`

Definition at line 285 of file address_space.hpp.

Inheritance diagram for trisycl::detail::address_space_ptr< T, AS >:



Collaboration diagram for trisycl::detail::address_space_ptr< T, AS >:



Public Types

- using super = address_space_fundamental< T, AS >

  Keep track of the base class as a short-cut.
- using pointer_t = typename super::address_space_fundamental::type
- using reference_t = typename std::remove_pointer_t< pointer_t > &

Public Member Functions

- address_space_ptr (address_space_fundamental< typename std::pointer_traits< T >::element_type, AS > *p)

  Allow initialization of a pointer type from the address of an element with the same type and address space.
- address_space_ptr ()=default

  Put back the default constructors canceled by the previous definition.

Additional Inherited Members

9.11.2.9.1 Member Typedef Documentation

9.11.2.9.1.1 pointer_t template<typename T , access::address_space AS>

```
using trisycl::detail::address_space_ptr< T, AS >::pointer_t = typename super::address_space_fundamental::typ
```

Definition at line 296 of file address_space.hpp.

9.11.2.9.1.2 reference_t template<typename T , access::address_space AS>

```
using trisycl::detail::address_space_ptr< T, AS >::reference_t = typename std::remove_pointer↩
_t<pointer_t>&
```

Definition at line 297 of file address_space.hpp.

9.11.2.9.1.3 super template<typename T , access::address_space AS>

```
using trisycl::detail::address_space_ptr< T, AS >::super = address_space_fundamental<T, AS>
```

Keep track of the base class as a short-cut.

Definition at line 291 of file address_space.hpp.

9.11.2.9.2 Constructor & Destructor Documentation

9.11.2.9.2.1 address_space_ptr() [1/2] template<typename T , access::address_space AS>

```
trisycl::detail::address_space_ptr< T, AS >::address_space_ptr (
            address_space_fundamental< typename std::pointer_traits< T >::element_type, AS >
* p )  [inline]
```

Allow initialization of a pointer type from the address of an element with the same type and address space.

Definition at line 302 of file address_space.hpp.

```
00303     :  address_space_fundamental<T, AS> { p->get_address() } {}
```

References p.

9.11.2.9.2.2 address_space_ptr() [2/2] template<typename T , access::address_space AS>

```
trisycl::detail::address_space_ptr< T, AS >::address_space_ptr ( )  [default]
```

Put back the default constructors canceled by the previous definition.

9.11.2.10 struct trisycl::detail::address_space_base

template<typename T, access::address_space AS>
struct trisycl::detail::address_space_base< T, AS >

Implementation of the base infrastructure to wrap something in an OpenCL address space.

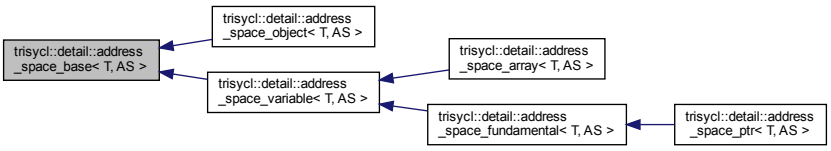
Parameters

T	is the type of the basic stuff to be created
AS	is the address space to place the object into

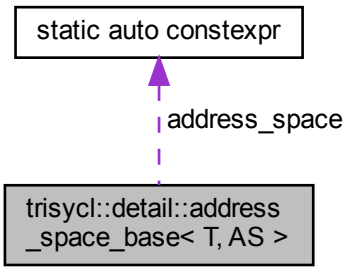
Todo Verify/improve to deal with const/volatile?

Definition at line 143 of file address_space.hpp.

Inheritance diagram for trisycl::detail::address_space_base< T, AS >:



Collaboration diagram for trisycl::detail::address_space_base< T, AS >:



Public Types

- using type = T

  Store the base type of the object.
- using opencl_type = typename ocl_type< T, AS >::type

  Store the base type of the object with OpenCL address space modifier.

Static Public Attributes

- static auto constexpr address_space = AS

  Set the address_space identifier that can be queried to know the pointer type.

9.11.2.10.1 Member Typedef Documentation

9.11.2.10.1.1 opencl_type `template<typename T , access::address_space AS>`

```
using trisycl::detail::address_space_base< T, AS >::opencl_type = typename ocl_type<T, AS>::type
```

Store the base type of the object with OpenCL address space modifier.

Todo Add to the specification

Definition at line 154 of file address_space.hpp.

9.11.2.10.1.2 type `template<typename T , access::address_space AS>`

```
using trisycl::detail::address_space_base< T, AS >::type = T
```

Store the base type of the object.

Todo Add to the specification

Definition at line 148 of file address_space.hpp.

9.11.2.10.2 Member Data Documentation

9.11.2.10.2.1 address_space `template<typename T , access::address_space AS>`

```
auto constexpr trisycl::detail::address_space_base< T, AS >::address_space = AS  [static],
[constexpr]
```

Set the address_space identifier that can be queried to know the pointer type.

Definition at line 158 of file address_space.hpp.

9.11.2.11 struct trisycl::detail::address_space_variable

template<typename T, access::address_space AS>
struct trisycl::detail::address_space_variable< T, AS >

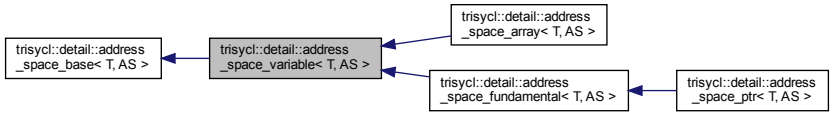
Implementation of a variable with an OpenCL address space.

Parameters

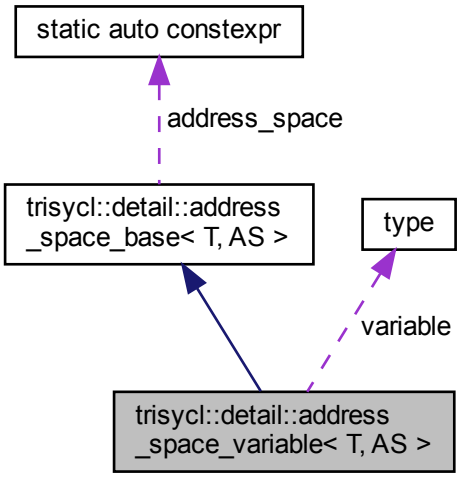
T	is the type of the basic object to be created
AS	is the address space to place the object into

Definition at line 170 of file address_space.hpp.

Inheritance diagram for trisycl::detail::address_space_variable< T, AS >:



Collaboration diagram for trisycl::detail::address_space_variable< T, AS >:



Public Types

- using opencl_type = typename ocl_type< T, AS >::type

  Store the base type of the object with OpenCL address space modifier.
- using super = address_space_base< T, AS >

  Keep track of the base class as a short-cut.

Public Member Functions

- address_space_variable (const T &v)

  Allow to create an address space version of an object or to convert one to be used by the classes inheriting by this one because it is not possible to directly initialize a base class member in C++.
- address_space_variable ()=default

  Put back the default constructors canceled by the previous definition.
- operator opencl_type & ()

  Conversion operator to allow a address_space_object<T> to be used as a T so that all the methods of a T and the built-in operators for T can be used on a address_space_object<T> too.
- opencl_type * get_address ()

  Return the address of the value to implement pointers.

Protected Attributes

- opencl_type variable

Additional Inherited Members

9.11.2.11.1 Member Typedef Documentation

9.11.2.11.1.1 opencl_type

```
template<typename T , access::address_space AS>
using trisycl::detail::address_space_variable< T, AS >::opencl_type = typename ocl_type<T,
AS>::type
```

Store the base type of the object with OpenCL address space modifier.

Todo Add to the specification

Definition at line 175 of file address_space.hpp.

9.11.2.11.1.2 super

```
template<typename T , access::address_space AS>
using trisycl::detail::address_space_variable< T, AS >::super = address_space_base<T, AS>
```

Keep track of the base class as a short-cut.

Definition at line 178 of file address_space.hpp.

9.11.2.11.2 Constructor & Destructor Documentation

9.11.2.11.2.1 address_space_variable() [1/2]

```
template<typename T , access::address_space AS>
trisycl::detail::address_space_variable< T, AS >::address_space_variable (
            const T & v )  [inline]
```

Allow to create an address space version of an object or to convert one to be used by the classes inheriting by this one because it is not possible to directly initialize a base class member in C++.

Definition at line 194 of file address_space.hpp.

```
00194 :  variable(v) { }
```

9.11.2.11.2.2 address_space_variable() [2/2]

```
template<typename T , access::address_space AS>
trisycl::detail::address_space_variable< T, AS >::address_space_variable ( )  [default]
```

Put back the default constructors canceled by the previous definition.

9.11.2.11.3 Member Function Documentation

9.11.2.11.3.1 get_address() template<typename T , access::address_space AS>

opencl_type * trisycl::detail::address_space_variable< T, AS >::get_address ( ) [inline]

Return the address of the value to implement pointers.

Definition at line 211 of file address_space.hpp.

```
00211 { return &variable; }
```

References trisycl::detail::address_space_variable< T, AS >::variable.

9.11.2.11.3.2 operator opencl_type &() template<typename T , access::address_space AS>

trisycl::detail::address_space_variable< T, AS >::operator opencl_type & ( ) [inline]

Conversion operator to allow a address_space_object<T> to be used as a T so that all the methods of a T and the built-in operators for T can be used on a address_space_object<T> too.

Use opencl_type so that if we take the address of it, the address space is kept.

Definition at line 208 of file address_space.hpp.

```
00208 { return variable; }
```

References trisycl::detail::address_space_variable< T, AS >::variable.

9.11.2.11.4 Member Data Documentation

9.11.2.11.4.1 variable template<typename T , access::address_space AS>

opencl_type trisycl::detail::address_space_variable< T, AS >::variable [protected]

Definition at line 187 of file address_space.hpp.

Referenced by trisycl::detail::address_space_array< T, AS >::address_space_array(), trisycl::detail::address_space_fundamental< T
trisycl::detail::address_space_variable< T, AS >::get_address(), and trisycl::detail::address_space_variable< T, AS >::operator ope

9.11.3 Typedef Documentation

9.11.3.1 addr_space

```
template<typename T , access::address_space AS>
using trisycl::detail::addr_space = typedef typename std::conditional<std::is_pointer<T>↩
::value, address_space_ptr<T, AS>, typename std::conditional<std::is_class<T>::value, address_space_object<T
AS>, typename std::conditional<std::is_array<T>::value, address_space_array<T, AS>, address_space_fundamenta
AS> >::type>::type>::type
```

#include <include/triSYCL/address_space/detail/address_space.hpp>

Dispatch the address space implementation according to the requested type.

Parameters

T	is the type of the object to be created
AS	is the address space to place the object into or to point to in the case of a pointer type

Definition at line 122 of file address_space.hpp.

9.11.3.2 constant

```
template<typename T >
using trisycl::constant = typedef detail::addr_space<T, access::address_space::constant_space>
```

```
#include <include/triSYCL/address_space.hpp>
```

Declare a variable to be in the OpenCL constant address space.

Parameters

T	is the type of the object

Definition at line 54 of file address_space.hpp.

9.11.3.3 constant_ptr

```
template<typename T >
using trisycl::constant_ptr = typedef constant<T*>
```

```
#include <include/triSYCL/address_space.hpp>
```

Declare a variable to be in the OpenCL constant address space.

Parameters

T	is the type of the object

Definition at line 62 of file address_space.hpp.

9.11.3.4 generic

```
template<typename T >
using trisycl::generic = typedef detail::addr_space<T, access::address_space::generic_space>
```

```
#include <include/triSYCL/address_space.hpp>
```

Declare a variable to be in the OpenCL 2 generic address space.

Parameters

T	is the type of the object

Definition at line 70 of file address_space.hpp.

9.11.3.5 global

```
template<typename T >
using trisycl::global = typedef detail::addr_space<T, access::address_space::global_space>
```

```
#include <include/triSYCL/address_space.hpp>
```

Declare a variable to be in the OpenCL global address space.

Parameters

T	is the type of the object

Definition at line 78 of file address_space.hpp.

9.11.3.6 global_ptr

```
template<typename T >
using trisycl::global_ptr = typedef global<T*>
```

```
#include <include/triSYCL/address_space.hpp>
```

Declare a variable to be in the OpenCL global address space.

Parameters

T	is the type of the object

Definition at line 87 of file address_space.hpp.

9.11.3.7 local

```
template<typename T >
using trisycl::local = typedef detail::addr_space<T, access::address_space::local_space>
```

```
#include <include/triSYCL/address_space.hpp>
```

Declare a variable to be in the OpenCL local address space.

Parameters

T	is the type of the object

Definition at line 95 of file address_space.hpp.

9.11.3.8 local_ptr

```
template<typename T >
using trisycl::local_ptr = typedef local<T*>
```

```
#include <include/triSYCL/address_space.hpp>
```

Declare a variable to be in the OpenCL local address space.

Parameters

T	is the type of the object

Definition at line 103 of file address_space.hpp.

9.11.3.9 multi_ptr

```
template<typename Pointer , access::address_space AS>
using trisycl::multi_ptr = typedef detail::address_space_ptr<Pointer, AS>
```

```
#include <include/triSYCL/address_space.hpp>
```

A pointer that can be statically associated to any address-space.

Parameters

Pointer	is the pointer type
AS	is the address space to point to

Note that if *Pointer* is not a pointer type, it is an error.

Definition at line 131 of file address_space.hpp.

9.11.3.10 priv

```
template<typename T >
using trisycl::priv = typedef detail::addr_space<T, access::address_space::private_space>
```

```
#include <include/triSYCL/address_space.hpp>
```

Declare a variable to be in the OpenCL private address space.

Parameters

T	is the type of the object

Definition at line 111 of file address_space.hpp.

9.11.3.11 private_ptr

```
template<typename T >
using trisycl::private_ptr = typedef priv<T*>
```

```
#include <include/triSYCL/address_space.hpp>
```

Declare a variable to be in the OpenCL private address space.

Parameters

T	is the type of the object

Definition at line 119 of file address_space.hpp.

9.11.4 Function Documentation

9.11.4.1 make_ptr() [1/2]

```
template<typename T , access::address_space AS>
multi_ptr< T, AS > trisycl::make_ptr (
            multi_ptr< T, AS > pointer )
```

```
#include <include/triSYCL/address_space.hpp>
```

Construct a trisycl::multi_ptr<> with the right type.

Parameters

pointer	is the address with its address space to point to

Definition at line 140 of file address_space.hpp.

```
00140                                                 {
00141   return pointer;
00142 }
```

9.11.4.2 make_ptr() [2/2]

```
template<typename T , access::address_space AS>
multi_ptr< T, AS > trisycl::make_ptr (
             T * pointer )
```

```
#include <include/triSYCL/address_space.hpp>
```

Construct a trisycl::multi_ptr<> with the right type.

Todo Implement the case with a plain pointer

Definition at line 149 of file address_space.hpp.

```
00149                                        {
00150   TRISYCL_UNIMPL;
00151   return pointer;
00152 }
```

References TRISYCL_UNIMPL.

9.12 Helpers to do array and tuple conversion

Classes

- struct trisycl::detail::expand_to_vector< V, Tuple, expansion >

  Allows optional expansion of a 1-element tuple to a V::dimension tuple to replicate scalar values in vector initialization. More...
- struct trisycl::detail::expand_to_vector< V, Tuple, true >

  Specialization in the case we ask for expansion. More...

Functions

- template<typename V , typename Tuple , std::size_t... Is>
  std::array< typename V::element_type, V::dimension > trisycl::detail::tuple_to_array_iterate (Tuple t, std::index_sequence< Is... >)

  Helper to construct an array from initializer elements provided as a tuple.
- template<typename V , typename Tuple >
  auto trisycl::detail::tuple_to_array (Tuple t)

  Construct an array from initializer elements provided as a tuple.
- static auto trisycl::detail::expand_to_vector< V, Tuple, expansion >::expand (Tuple t)
- template<typename Value , std::size_t... Is>
  static auto trisycl::detail::expand_to_vector< V, Tuple, true >::fill_tuple (Value e, std::index_sequence< Is... >)

  Construct a tuple from a value.
- static auto trisycl::detail::expand_to_vector< V, Tuple, true >::expand (Tuple t)

  We expand the 1-element tuple by replicating into a tuple with the size of the vector.
- template<typename V , typename Tuple >
  auto trisycl::detail::expand (Tuple t)

  Create the array data of V from a tuple of initializer.

9.12.1 Detailed Description

9.12.2 Class Documentation

9.12.2.1 struct trisycl::detail::expand_to_vector

template<typename V, typename Tuple, bool expansion = false>
struct trisycl::detail::expand_to_vector< V, Tuple, expansion >

Allows optional expansion of a 1-element tuple to a V::dimension tuple to replicate scalar values in vector initialization.

Definition at line 62 of file array_tuple_helpers.hpp.

Static Public Member Functions

- static auto expand (Tuple t)

9.12.2.2 struct trisycl::detail::expand_to_vector< V, Tuple, true >

template<typename V, typename Tuple>
struct trisycl::detail::expand_to_vector< V, Tuple, true >

Specialization in the case we ask for expansion.

Definition at line 72 of file array_tuple_helpers.hpp.

Static Public Member Functions

- template<typename Value , std::size_t... Is>
  static auto fill_tuple (Value e, std::index_sequence< Is... >)

  Construct a tuple from a value.
- static auto expand (Tuple t)

  We expand the 1-element tuple by replicating into a tuple with the size of the vector.

9.12.3 Function Documentation

9.12.3.1 expand() [1/3]

```
template<typename V , typename Tuple , bool expansion = false>
static auto trisycl::detail::expand_to_vector< V, Tuple, expansion >::expand (
             Tuple t )  [inline], [static]
```

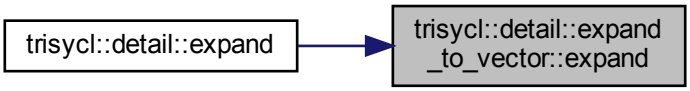
#include <include/triSYCL/detail/array_tuple_helpers.hpp>

Definition at line 68 of file array_tuple_helpers.hpp.

```
00068 { return t; }
```

Referenced by trisycl::detail::expand().

Here is the caller graph for this function:



9.12.3.2 expand() [2/3]

```
template<typename V , typename Tuple >
static auto trisycl::detail::expand_to_vector< V, Tuple, true >::expand (
             Tuple t )  [inline], [static]
```

#include <include/triSYCL/detail/array_tuple_helpers.hpp>

We expand the 1-element tuple by replicating into a tuple with the size of the vector.

Definition at line 104 of file array_tuple_helpers.hpp.

```
00104                                    {
00105     return fill_tuple(std::get<0>(t),
00106                       std::make_index_sequence<V::dimension>{});
00107   }
```

9.12.3.3 expand() [3/3]

```
template<typename V , typename Tuple >
auto trisycl::detail::expand (
             Tuple t )
```

`#include <include/triSYCL/detail/array_tuple_helpers.hpp>`

Create the array data of V from a tuple of initializer.

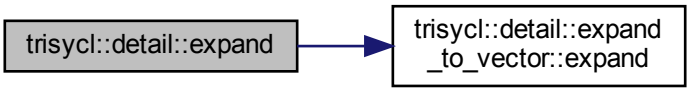
If there is only 1 initializer, this is a scalar initialization of a vector and the value is expanded to all the vector elements first.

Definition at line 116 of file array_tuple_helpers.hpp.

```
00116                     {
00117   return tuple_to_array<V>(expand_to_vector<V,
00118                             decltype(t),
00119                             /* Only ask the expansion to all vector
00120 element if there only a scalar
00121 initializer */
00122                             std::tuple_size_v<Tuple> == 1>{}.expand(t));
00123 }
```

References trisycl::detail::expand_to_vector< V, Tuple, expansion >::expand().

Here is the call graph for this function:



9.12.3.4 fill_tuple()

```
template<typename V , typename Tuple >
template<typename Value , std::size_t...  Is>
static auto trisycl::detail::expand_to_vector< V, Tuple, true >::fill_tuple (
             Value e,
             std::index_sequence< Is...  >  )  [inline], [static]
```

`#include <include/triSYCL/detail/array_tuple_helpers.hpp>`

Construct a tuple from a value.

Parameters

value	is used to initialize each tuple element
size	is the number of elements of the tuple to be generated

The trick is to get the std::index_sequence<> that represent 0, 1,..., dimension-1 as a variadic template pack Is that we can iterate on, in this function.

Definition at line 88 of file array_tuple_helpers.hpp.

```
00088                                                                    {
00089     /* The effect is like a static for-loop with Is counting from 0 to
00090 dimension-1 and thus replicating the pattern to have
00091 make_tuple( (0, e), (1, e), ...  (n - 1, e) )
00092
00093 Since the "," operator is just here to throw away the Is value
00094 (which is needed for the pack expansion...), at the end this is
00095 equivalent to:
00096 make_tuple( e, e, ..., e )
00097 */
00098     return std::make_tuple(((void)Is, e)...);
00099   }
```

9.12.3.5 tuple_to_array()

```
template<typename V , typename Tuple >
auto trisycl::detail::tuple_to_array (
             Tuple t )
```

```
#include <include/triSYCL/detail/array_tuple_helpers.hpp>
```

Construct an array from initializer elements provided as a tuple.

Definition at line 51 of file array_tuple_helpers.hpp.

```
00051                                {
00052   /* Construct an index_sequence with 0, 1, ..., (size of the tuple-1)
00053 so that tuple_to_array_iterate can statically iterate on it */
00054   return tuple_to_array_iterate<V>(
00055       t, std::make_index_sequence<std::tuple_size_v<Tuple» {});
00056 }
```

9.12.3.6 tuple_to_array_iterate()

```
template<typename V , typename Tuple , std::size_t...  Is>
std::array< typename V::element_type, V::dimension > trisycl::detail::tuple_to_array_iterate (
             Tuple t,
             std::index_sequence< Is...  >  )
```

```
#include <include/triSYCL/detail/array_tuple_helpers.hpp>
```

Helper to construct an array from initializer elements provided as a tuple.

The trick is to get the std::index_sequence<> that represent 0, 1,..., dimension-1 as a variadic template pack Is that we can iterate on, in this function.

Definition at line 35 of file array_tuple_helpers.hpp.

```
00035                                                                  {
00036   /* The effect is like a static for-loop with Is counting from 0 to
00037 dimension-1 and thus constructing a uniform initialization { }
00038 construction from each tuple element:
00039 { std::get<0>(t), std::get<1>(t), ..., std::get<dimension-1>(t) }
00040
00041 The static cast is here to avoid the warning when there is a loss
00042 of precision, for example when initializing an int from a float.
00043 */
00044   return { { static_cast<typename V::element_type>(std::get<Is>(t))...  } };
00045 }
```

9.13 Some helpers for the implementation

Concepts

- concept trisycl::detail::is_small_array

  Concept describing a class inheriting from small_array.

Classes

- struct trisycl::detail::container_element_aspect< T >

  A mix-in to add some container element aspects. More...
- struct trisycl::detail::overloaded< Ts >

  Helper type for the visitor to do pattern matching on invocables https://en.cppreference.com/w/cpp/utility/variant/visit. More...
- struct trisycl::detail::small_array_tag

  Tag to define simply inheritance from small_array. More...
- struct trisycl::detail::small_array< BasicType, FinalType, Dims >

  Define a multi-dimensional index, used for example to locate a work item or a buffer element. More...
- struct trisycl::detail::small_array_sycl< BasicType, FinalType, Dims >

  A small array to implement SYCL objects like id, range, item... with 1, 2 or 3 elements (or even more if detail::allow_any_dimension is true), with the implicit constructors. More...

Macros

- #define TRISYCL_BOOST_OPERATOR_VECTOR_OP(op)

  Helper macro to declare a vector operation with the given side-effect operator.
- #define TRISYCL_LOGICAL_OPERATOR_VECTOR_OP(op)

  Helper macro to declare a vector operation returning a new type containing the result of the operator.
- #define TRISYCL_UNARY_OPERATOR_VECTOR_OP(op)

  Helper macro to declare a vector unary operation.
- #define TRISYCL_UNARY_PREFIX_OPERATOR_VECTOR_OP(op)

  Helper macro to declare a vector prefix unary operation.
- #define TRISYCL_UNARY_POSTFIX_OPERATOR_VECTOR_OP(op)

  Helper macro to declare a vector prefix unary operation.
- #define TRISYCL_OPERATOR_BASIC_TYPE_OP(op)

  Helper macro to declare out-of-class operators for b vs a[] operations.
- #define TRISYCL_UNIMPL trisycl::detail::unimplemented(__func__, __FILE__, __LINE__)

Functions

- template<typename Range , typename Id >
  size_t constexpr trisycl::detail::linear_id (Range range, Id id, Id offset={})

  Compute a linearized array access used in the OpenCL 2 world.
- template<class... Ts>
  trisycl::detail::overloaded (Ts...) -> overloaded< Ts... >
- void trisycl::detail::unimplemented (const char *func, const char *file, int line)

  Display an "unimplemented" message.

9.13.1 Detailed Description

9.13.2 Class Documentation

9.13.2.1 struct trisycl::detail::container_element_aspect

template<typename T>
struct trisycl::detail::container_element_aspect< T >

A mix-in to add some container element aspects.

Definition at line 21 of file container_element_aspect.hpp.

Public Types

- using value_type = T
- using pointer = value_type *
- using const_pointer = const value_type *
- using reference = value_type &
- using const_reference = const value_type &

9.13.2.1.1 Member Typedef Documentation

9.13.2.1.1.1 const_pointer template<typename T >

```
using trisycl::detail::container_element_aspect< T >::const_pointer = const value_type*
```

Definition at line 25 of file container_element_aspect.hpp.

9.13.2.1.1.2 const_reference template<typename T >

```
using trisycl::detail::container_element_aspect< T >::const_reference = const value_type&
```

Definition at line 27 of file container_element_aspect.hpp.

9.13.2.1.1.3 pointer template<typename T >

```
using trisycl::detail::container_element_aspect< T >::pointer = value_type*
```

Definition at line 24 of file container_element_aspect.hpp.

9.13.2.1.1.4 reference template<typename T >

```
using trisycl::detail::container_element_aspect< T >::reference = value_type&
```

Definition at line 26 of file container_element_aspect.hpp.

9.13.2.1.1.5 value_type template<typename T >

```
using trisycl::detail::container_element_aspect< T >::value_type = T
```

Definition at line 23 of file container_element_aspect.hpp.

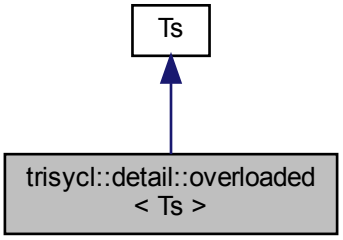
9.13.2.2 struct trisycl::detail::overloaded

template<class... Ts>
struct trisycl::detail::overloaded< Ts >

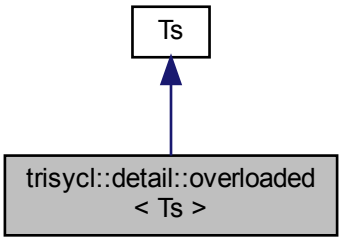
Helper type for the visitor to do pattern matching on invocables https://en.cppreference.com/w/cpp/utility/variant/visit.

Definition at line 18 of file overloaded.hpp.

Inheritance diagram for trisycl::detail::overloaded< Ts >:



Collaboration diagram for trisycl::detail::overloaded< Ts >:

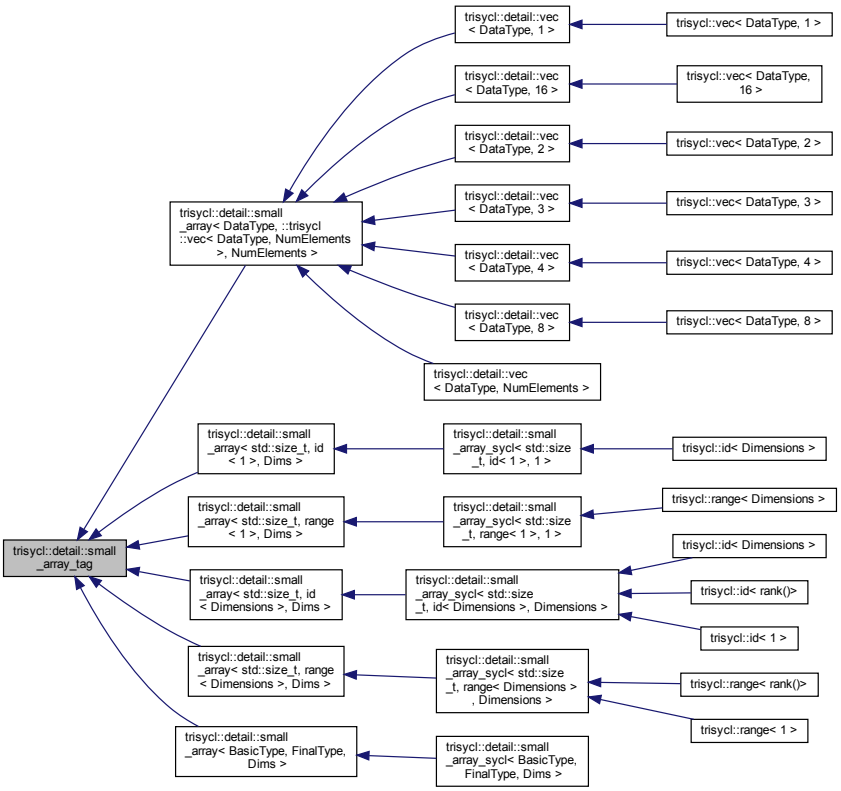


9.13.2.3 struct trisycl::detail::small_array_tag

Tag to define simply inheritance from small_array.

Definition at line 36 of file small_array.hpp.

Inheritance diagram for trisycl::detail::small_array_tag:



9.13.2.4 struct trisycl::detail::small_array

template<typename BasicType, typename FinalType, std::size_t Dims>
struct trisycl::detail::small_array< BasicType, FinalType, Dims >

Define a multi-dimensional index, used for example to locate a work item or a buffer element.

Unfortunately, even if std::array is an aggregate class allowing native list initialization, it is no longer an aggregate if we derive from an aggregate. Thus we have to redeclare the constructors.

Parameters

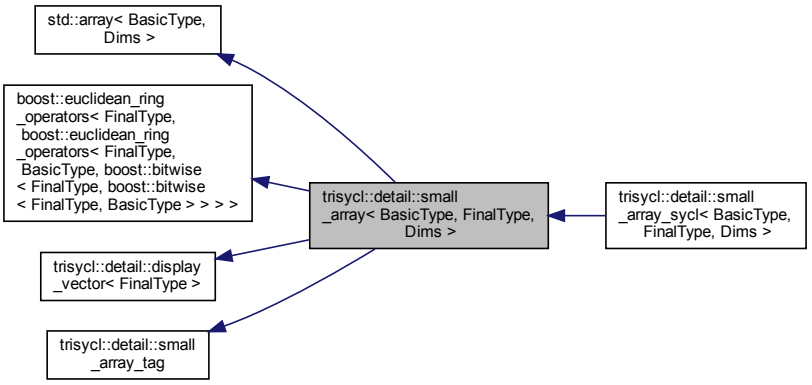
BasicType	is the type element, such as int
Dims	is the dimension number, typically between 1 and 3
FinalType	is the final type, such as range<> or id<>, so that boost::operator can return the right type

std::array<> provides the collection concept, with .size(), == and != too.

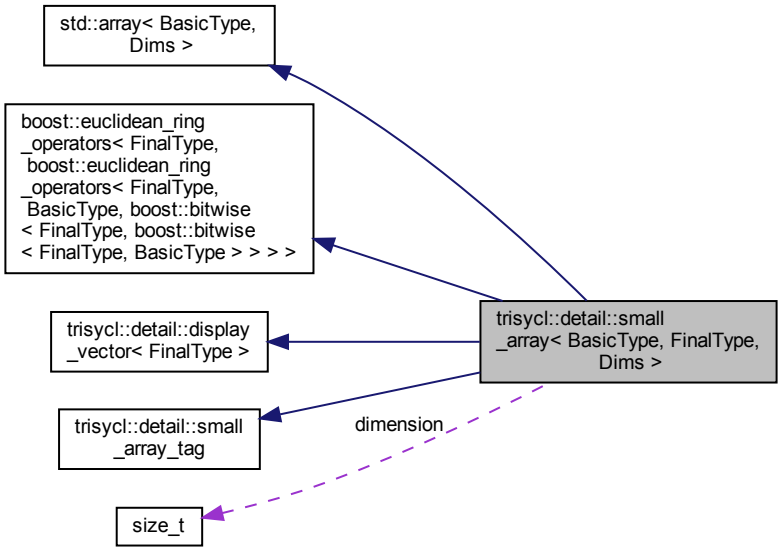
Can't use boost::shiftable, as it doesn't define the U, T overload outside the class, but when that is added, it gets ambiguity problems.

Definition at line 140 of file small_array.hpp.

Inheritance diagram for trisycl::detail::small_array< BasicType, FinalType, Dims >:



Collaboration diagram for trisycl::detail::small_array< BasicType, FinalType, Dims >:



Public Types

- using element_type = BasicType

Public Member Functions

- template<typename SourceType >
  small_array (const SourceType src[Dims])

  A constructor from another array.
- template<typename SourceBasicType , typename SourceFinalType >
  small_array (const small_array< SourceBasicType, SourceFinalType, Dims > &src)

  A constructor from another small_array of the same size.
- template<typename... Types>
  requires (sizeof...(Types) == Dims)
  small_array (const Types &... args)

  Initialize the array from a list of exactly Dims elements.
- template<typename SourceBasicType >
  small_array (const std::array< SourceBasicType, Dims > &src)

  Construct a small_array from a std::array.
- small_array ()=default

  Keep the synthesized constructors.
- auto get (std::size_t index) const

  Return the element of the array.
- operator FinalType ()

  Add + like operations on the id<> and others.

Static Public Attributes

- static constexpr size_t dimension = Dims

9.13.2.4.1 Member Typedef Documentation

9.13.2.4.1.1 element_type

```
template<typename BasicType , typename FinalType , std::size_t Dims>
using trisycl::detail::small_array< BasicType, FinalType, Dims >::element_type = BasicType
```

Definition at line 161 of file small_array.hpp.

9.13.2.4.2 Constructor & Destructor Documentation

9.13.2.4.2.1 small_array() [1/5]

```
template<typename BasicType , typename FinalType , std::size_t
Dims>
template<typename SourceType >
trisycl::detail::small_array< BasicType, FinalType, Dims >::small_array (
            const SourceType src[Dims] )  [inline]
```

A constructor from another array.

Make it explicit to avoid spurious range<> constructions from int * for example

Definition at line 169 of file small_array.hpp.

```
00170                                       {
00171     // (*this)[0] is the first element of the underlying array
00172     std::copy_n(src, Dims, &(*this)[0]);
```

References src.

9.13.2.4.2.2 small_array() [2/5]

```
template<typename BasicType , typename FinalType , std::size_t
Dims>
template<typename SourceBasicType , typename SourceFinalType >
trisycl::detail::small_array< BasicType, FinalType, Dims >::small_array (
            const small_array< SourceBasicType, SourceFinalType, Dims > & src )  [inline]
```

A constructor from another small_array of the same size.

Definition at line 178 of file small_array.hpp.

```
00181                                {
00182     std::copy_n(&src[0], Dims, &(*this)[0]);
```

References src.

9.13.2.4.2.3 small_array() [3/5]

```
template<typename BasicType , typename FinalType , std::size_t
Dims>
template<typename... Types>
requires (sizeof...(Types) == Dims)
trisycl::detail::small_array< BasicType, FinalType, Dims >::small_array (
            const Types &... args )  [inline]
```

Initialize the array from a list of exactly Dims elements.

Strangely, even when using the array constructors, the initialization of the aggregate is not available. So recreate an equivalent here.

Definition at line 192 of file small_array.hpp.

```
00194       :  std::array<BasicType, Dims> {
00195         // Allow a loss of precision in initialization with the static_cast
00196         { static_cast<BasicType>(args)... }
```

9.13.2.4.2.4 small_array() [4/5]

```
template<typename BasicType , typename FinalType , std::size_t
Dims>
template<typename SourceBasicType >
trisycl::detail::small_array< BasicType, FinalType, Dims >::small_array (
            const std::array< SourceBasicType, Dims > & src )  [inline]
```

Construct a small_array from a std::array.

Definition at line 200 of file small_array.hpp.

```
00202     :  std::array<BasicType, Dims>(src) {}
```

9.13.2.4.2.5 small_array() [5/5]

```
template<typename BasicType , typename FinalType , std::size_t
Dims>
trisycl::detail::small_array< BasicType, FinalType, Dims >::small_array ( )  [default]
```

Keep the synthesized constructors.

9.13.2.4.3 Member Function Documentation

9.13.2.4.3.1 get() template<typename BasicType , typename FinalType , std::size_t Dims> auto trisycl::detail::small_array< BasicType, FinalType, Dims >::get (std::size_t *index*) const [inline]

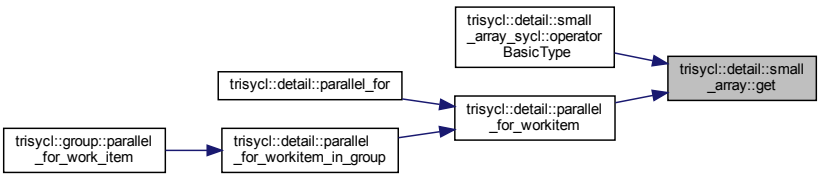
Return the element of the array.

Definition at line 211 of file small_array.hpp.

```
00212                                 {
00213     return (*this)[index];
```

Referenced by trisycl::detail::small_array_sycl< BasicType, FinalType, Dims >::operator BasicType(), and trisycl::detail::parallel_for_workitem().

Here is the caller graph for this function:



9.13.2.4.3.2 operator FinalType() template<typename BasicType , typename FinalType , std::size_t Dims> trisycl::detail::small_array< BasicType, FinalType, Dims >::operator FinalType ( ) [inline]

Add + like operations on the id<> and others.

Add - like operations on the id<> and others Add * like operations on the id<> and others Add / like operations on the id<> and others Add % like operations on the id<> and others Add << like operations on the id<> and others Add >> like operations on the id<> and others Add & like operations on the id<> and others Add ^ like operations on the id<> and others Add | like operations on the id<> and others Add && operations on the id<> and others Add || operations on the id<> and others Add comparison operations on the id<> and others Add shiftable operators on the vector op Implement unary operations Implement prefix unary operations Implement postfix unary operations Since the boost::operator work on the small_array, add an implicit conversion to produce the expected type

Definition at line 287 of file small_array.hpp.

```
00288                           {
00289     return static_cast<FinalType&>(*this);
```

9.13.2.4.4 Member Data Documentation

9.13.2.4.4.1 dimension template<typename BasicType , typename FinalType , std::size_t Dims> constexpr size_t trisycl::detail::small_array< BasicType, FinalType, Dims >::dimension = Dims [static], [constexpr]

Definition at line 159 of file small_array.hpp.

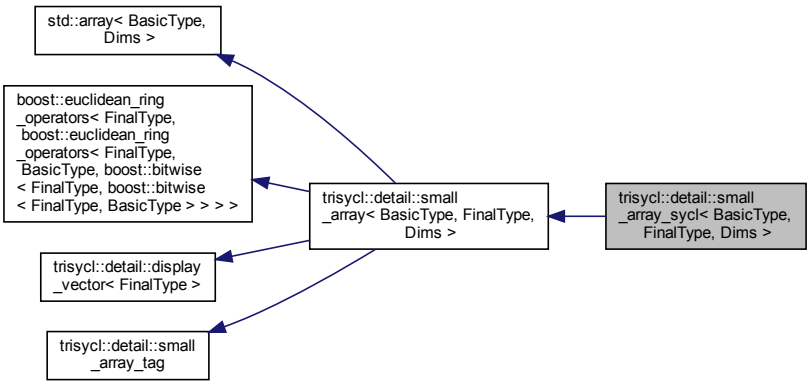
9.13.2.5 struct trisycl::detail::small_array_sycl

template<typename BasicType, typename FinalType, std::size_t Dims>
struct trisycl::detail::small_array_sycl< BasicType, FinalType, Dims >

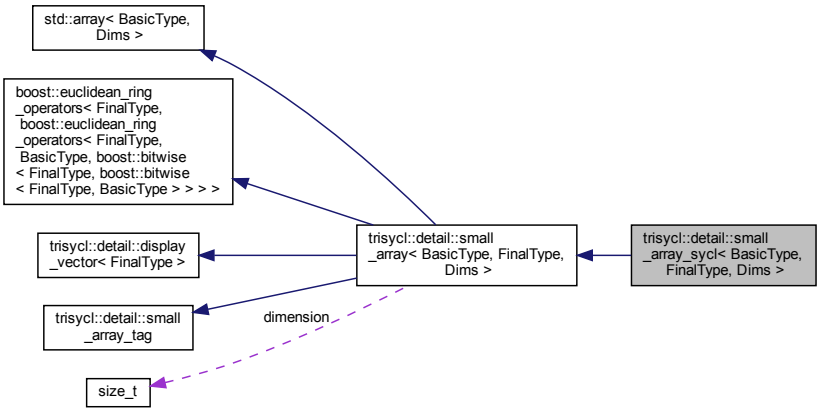
A small array to implement SYCL objects like id, range, item... with 1, 2 or 3 elements (or even more if detail::allow_any_dimension is true), with the implicit constructors.

Definition at line 325 of file small_array.hpp.

Inheritance diagram for trisycl::detail::small_array_sycl< BasicType, FinalType, Dims >:



Collaboration diagram for trisycl::detail::small_array_sycl< BasicType, FinalType, Dims >:



Public Member Functions

- small_array_sycl ()=default

  Keep other constructors.

- template<typename Range >
  requires requires { ranges::copy(std::declval<Range>(), std::declval<BasicType*>()); }
  small_array_sycl (Range &&r)

  Construct from a range-like object inheriting from a compatible std::array.
- template<typename... Args>
  requires requires { (static_cast<BasicType>(std::declval<Args>()), ...); }
  small_array_sycl (Args... args)

  Constructor taking either Dims arguments compatible with BasicType or just 1 argument compatible with BasicType and broadcast its value to all the elements.
- template<auto T = Dims>
  requires (Dims == 1)
  operator BasicType ()

  Conversion operator to scalar BasicType when only 1 dimension.

Additional Inherited Members

9.13.2.5.1 Constructor & Destructor Documentation

9.13.2.5.1.1 small_array_sycl() [1/3]

```
template<typename BasicType , typename FinalType , std::size_t Dims>
trisycl::detail::small_array_sycl< BasicType, FinalType, Dims >::small_array_sycl ( )  [default]
```

Keep other constructors.

9.13.2.5.1.2 small_array_sycl() [2/3]

```
template<typename BasicType , typename FinalType , std::size_t Dims>
template<typename Range >
requires requires { ranges::copy(std::declval<Range>(), std::declval<BasicType*>()); }
trisycl::detail::small_array_sycl< BasicType, FinalType, Dims >::small_array_sycl (
            Range && r )  [inline]
```

Construct from a range-like object inheriting from a compatible std::array.

This useful in implementation details for example to construct a sycl::id from a sycl::range.

Definition at line 343 of file small_array.hpp.

```
00344                                {
00345     // Copy to internal value
00346     ranges::copy(std::forward<Range>(r), this->begin());
```

9.13.2.5.1.3 small_array_sycl() [3/3]

```
template<typename BasicType , typename FinalType , std::size_t Dims>
template<typename... Args>
requires requires { (static_cast<BasicType>(std::declval<Args>()), ...); }
trisycl::detail::small_array_sycl< BasicType, FinalType, Dims >::small_array_sycl (
            Args... args ) [inline]
```

Constructor taking either Dims arguments compatible with BasicType or just 1 argument compatible with BasicType and broadcast its value to all the elements.

Definition at line 356 of file small_array.hpp.

```
00356                     { (static_cast<BasicType>(std::declval<Args>()), ...); }
00357   small_array_sycl(Args... args) {
00358     if constexpr (sizeof...(Args) == Dims) {
00359       // If there is the right number of elements we initialize each
00360       // element from each parameter, allowing some narrowing
00361       static_cast<std::array<BasicType, Dims>&>(
00362           *this) = { static_cast<BasicType>(args)... };
00363     } else if constexpr (sizeof...(Args) == 1) {
00364       // If there is only 1 argument, this is a broadcast
00365       detail::assign_from_tuple(
00366           static_cast<std::array<BasicType, Dims>&>(*this),
00367           detail::expand_to_vector<small_array_sycl, std::tuple<BasicType>,
00368                                    true> {}
00369               .expand(args...));
00370     } else
00371       static_assert(true, "There should be either 1 argument to broadcast the "
00372                           "value or 1 value per dimension");
```

9.13.2.5.2 Member Function Documentation

9.13.2.5.2.1 operator BasicType()

```
template<typename BasicType , typename FinalType , std::size_t Dims>
template<auto T = Dims>
requires (Dims == 1)
trisycl::detail::small_array_sycl< BasicType, FinalType, Dims >::operator BasicType ( ) [inline]
```

Conversion operator to scalar BasicType when only 1 dimension.

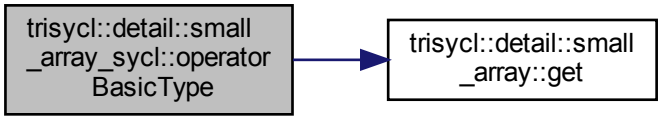
This is used for example to implicitly convert a sycl::id<1> to a std::size_t

Definition at line 381 of file small_array.hpp.

```
00382                           {
00383     return this->get(0);
```

References trisycl::detail::small_array< BasicType, FinalType, Dims >::get().

Here is the call graph for this function:



9.13.3 Macro Definition Documentation

9.13.3.1 TRISYCL_BOOST_OPERATOR_VECTOR_OP

```
#define TRISYCL_BOOST_OPERATOR_VECTOR_OP(
              op )
```

```
#include <include/triSYCL/detail/small_array.hpp>
```

Value:

```
  FinalType operator op(const FinalType &rhs) {          \
    for (std::size_t i = 0; i != Dims; ++i)              \
      (*this)[i] op rhs[i];                              \
    return *this;                                        \
  }                                                      \
  FinalType operator op(const BasicType &rhs) {          \
    for (std::size_t i = 0; i != Dims; ++i)              \
      (*this)[i] op rhs;                                 \
    return *this;                                        \
  }                                                      \
```

Helper macro to declare a vector operation with the given side-effect operator.

This handles both a[i] op b[i] and a[i] op b, where b is a BasicType.

Definition at line 47 of file small_array.hpp.

9.13.3.2 TRISYCL_LOGICAL_OPERATOR_VECTOR_OP

```
#define TRISYCL_LOGICAL_OPERATOR_VECTOR_OP(
              op )
```

```
#include <include/triSYCL/detail/small_array.hpp>
```

Value:

```
  FinalType operator op(const FinalType &rhs)const {             \
    FinalType res;                                                \
    for (std::size_t i = 0; i != Dims; ++i)                       \
      res[i] = (*this)[i] op rhs[i];                              \
    return res;                                                   \
  }                                                               \
  /* Skip this for Dims = 1 to avoid ambiguity with implicit type \
conversion between the vector type and its basic type */      \
  template <typename FT = FinalType,                              \
            typename = std::enable_if_t<Dims != 1, FT>>           \
  FinalType operator op(const BasicType &rhs) {                   \
    FinalType res;                                                \
    for (std::size_t i = 0; i != Dims; ++i)                       \
      res[i] = (*this)[i] op rhs;                                 \
    return res;                                                   \
  }
```

Helper macro to declare a vector operation returning a new type containing the result of the operator.

This handles both a[] op b[] and a[] op b, where b is a BasicType.

Definition at line 64 of file small_array.hpp.

9.13.3.3 TRISYCL_OPERATOR_BASIC_TYPE_OP

```
#define TRISYCL_OPERATOR_BASIC_TYPE_OP(
              op )
```

```
#include <include/triSYCL/detail/small_array.hpp>
```

Value:

```
template <typename BasicType, typename FinalType, std::size_t Dims>    \
struct small_array<BasicType, FinalType, Dims> operator op(            \
  const BasicType &lhs,                                                \
  const struct small_array<BasicType, FinalType, Dims> &rhs)           \
{                                                                      \
  FinalType res;                                                       \
  for (std::size_t i = 0; i != Dims; ++i)                              \
    res[i] = lhs op rhs[i];                                            \
  return res;                                                          \
}
```

Helper macro to declare out-of-class operators for b vs a[] operations.

Definition at line 295 of file small_array.hpp.

9.13.3.4 TRISYCL_UNARY_OPERATOR_VECTOR_OP

```
#define TRISYCL_UNARY_OPERATOR_VECTOR_OP(
              op )
```

```
#include <include/triSYCL/detail/small_array.hpp>
```

Value:

```
  FinalType operator op()const {                \
    FinalType result;                            \
    for (std::size_t i = 0; i != Dims; ++i)      \
      result[i] = op(*this)[i];                  \
    return result;                               \
  }
```

Helper macro to declare a vector unary operation.

This handles op a.

Definition at line 85 of file small_array.hpp.

9.13.3.5 TRISYCL_UNARY_POSTFIX_OPERATOR_VECTOR_OP

```
#define TRISYCL_UNARY_POSTFIX_OPERATOR_VECTOR_OP(
              op )
```

```
#include <include/triSYCL/detail/small_array.hpp>
```

Value:

```
  FinalType operator op(int) {                  \
    FinalType result = *this;                    \
    for (std::size_t i = 0; i != Dims; ++i)      \
      op(*this)[i];                              \
    return result;                               \
  }
```

Helper macro to declare a vector prefix unary operation.

This handles a++ and a–.

Definition at line 108 of file small_array.hpp.

9.13.3.6 TRISYCL_UNARY_PREFIX_OPERATOR_VECTOR_OP

```
#define TRISYCL_UNARY_PREFIX_OPERATOR_VECTOR_OP(
              op )
```

```
#include <include/triSYCL/detail/small_array.hpp>
```

Value:

```
  FinalType operator op() {                          \
    for (std::size_t i = 0; i != Dims; ++i)          \
      op(*this)[i];                                  \
    return *this;                                    \
  }
```

Helper macro to declare a vector prefix unary operation.

This handles ++a and –a.

Definition at line 97 of file small_array.hpp.

9.13.3.7 TRISYCL_UNIMPL

```
#define TRISYCL_UNIMPL trisycl::detail::unimplemented(__func__, __FILE__, __LINE__)
```

```
#include <include/triSYCL/detail/unimplemented.hpp>
```

Definition at line 32 of file unimplemented.hpp.

9.13.4 Function Documentation

9.13.4.1 linear_id()

```
template<typename Range , typename Id >
size_t constexpr trisycl::detail::linear_id (
             Range range,
             Id id,
             Id offset = {} )  [inline], [constexpr]
```

```
#include <include/triSYCL/detail/linear_id.hpp>
```

Compute a linearized array access used in the OpenCL 2 world.

Typically for the get_global_linear_id() and get_local_linear_id() functions.

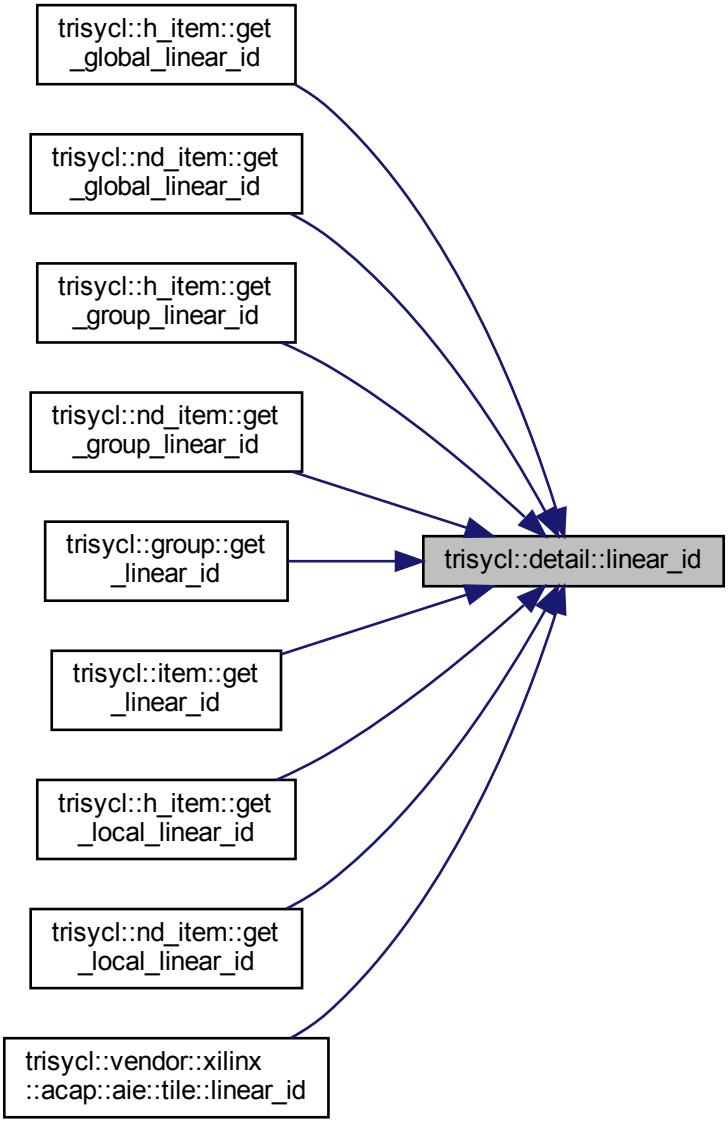
Definition at line 26 of file linear_id.hpp.

```
00026                                                                {}) {
00027   auto dims = std::distance(std::begin(range), std::end(range));
00028
00029   size_t linear_id = 0;
00030   /* A good compiler should unroll this and do partial evaluation to
00031 remove the first multiplication by 0 of this Horner evaluation and
00032 remove the 0 offset evaluation */
00033   for (int i = dims - 1; i >= 0; --i)
```

```
00034     linear_id = linear_id*range[i] + id[i] - offset[i];
00035
00036   return linear_id;
00037 }
```

Referenced by trisycl::h_item< Dimensions >::get_global_linear_id(), trisycl::nd_item< Dimensions >::get_global_linear_id(), trisycl::h_item< Dimensions >::get_group_linear_id(), trisycl::nd_item< Dimensions >::get_group_linear_id(), trisycl::group< Dimensions >::get_linear_id(), trisycl::item< Dimensions, with_offset >::get_linear_id(), trisycl::h_item< Dimensions trisycl::nd_item< Dimensions >::get_local_linear_id(), and trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >::linear_id().

Here is the caller graph for this function:



9.13.4.2 overloaded()

```
template<class...  Ts>
trisycl::detail::overloaded (
            Ts...    )  -> overloaded< Ts...  >
```

```
#include <include/triSYCL/detail/overloaded.hpp>
```

9.13.4.3 unimplemented()

```
void trisycl::detail::unimplemented (
            const char * func,
            const char * file,
            int line )  [inline]
```

```
#include <include/triSYCL/detail/unimplemented.hpp>
```

Display an "unimplemented" message.

Can be changed to call assert(0) or whatever.

Definition at line 23 of file unimplemented.hpp.

```
00023                                                                        {
00024 #ifndef NDEBUG
00025   std::cerr « "Error:  using a non implemented feature :  " « func « std::endl
00026            « "at " « file « ":" « line « std::endl
00027            « "Please contribute to the open source implementation.  :-)"
00028            « std::endl;
00029 #endif
00030 }
```

9.14 Debugging and tracing support

Classes

- struct trisycl::detail::debug< T >

  Class used to trace the construction, copy-construction, move-construction and destruction of classes that inherit from it. More...
- struct trisycl::detail::display_vector< T >

  Class used to display a vector-like type of classes that inherit from it. More...

Functions

- template<typename KernelName , typename Functor >
  auto trisycl::detail::trace_kernel (Functor f)

  Wrap a kernel functor in some tracing messages to have start/stop information when TRISYCL_TRACE_KERNEL macro is defined.

9.14.1 Detailed Description

9.14.2 Class Documentation

9.14.2.1 struct trisycl::detail::debug

template<typename T>
struct trisycl::detail::debug< T >

Class used to trace the construction, copy-construction, move-construction and destruction of classes that inherit from it.

Also trace the copy and move assignments.

Parameters

T	is the real type name to be used in the debug output.

Definition at line 69 of file debug.hpp.

Static Public Member Functions

- static auto constexpr type_pretty_name ()

  Get the pretty name of T itself.

9.14.2.1.1 Member Function Documentation

9.14.2.1.1.1 type_pretty_name() `template<typename T >`

`static auto constexpr trisycl::detail::debug< T >::type_pretty_name ( ) [inline], [static], [constexpr]`

Get the pretty name of T itself.

Definition at line 71 of file debug.hpp.

```
00071                                          {
00072     return boost::typeindex::type_id<T>().pretty_name();
00073   }
```

9.14.2.2 struct trisycl::detail::display_vector

template<typename T>
struct trisycl::detail::display_vector< T >

Class used to display a vector-like type of classes that inherit from it.

Parameters

T	is the real type name to be used in the debug output.

Calling the display() method dump the values on std::cout

Definition at line 163 of file debug.hpp.

Public Member Functions

- void display () const

  To debug and test.

9.14.2.2.1 Member Function Documentation

9.14.2.2.1.1 display() template<typename T >

```
void trisycl::detail::display_vector< T >::display ( ) const  [inline]
```

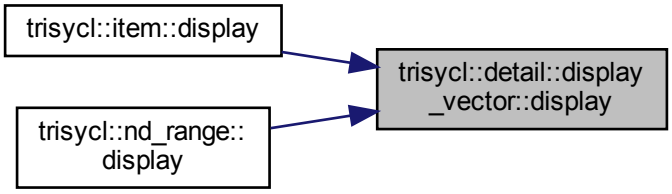
To debug and test.

Definition at line 166 of file debug.hpp.

```
00166                         {
00167 #ifdef TRISYCL_DEBUG
00168     std::cout « boost::typeindex::type_id<T>().pretty_name() « ":";
00169 #endif
00170     // Get a pointer to the real object
00171     for (auto e :  static_cast<const T&>(*this))
00172       std::cout « " " « e;
00173     std::cout « std::endl;
00174   }
```

Referenced by trisycl::item< Dimensions, with_offset >::display(), and trisycl::nd_range< Dimensions >::display().

Here is the caller graph for this function:



9.14.3 Function Documentation

9.14.3.1 trace_kernel()

```
template<typename KernelName , typename Functor >
auto trisycl::detail::trace_kernel (
             Functor f )
```

`#include <include/triSYCL/detail/debug.hpp>`

Wrap a kernel functor in some tracing messages to have start/stop information when TRISYCL_TRACE_KERNEL macro is defined.

Definition at line 133 of file debug.hpp.

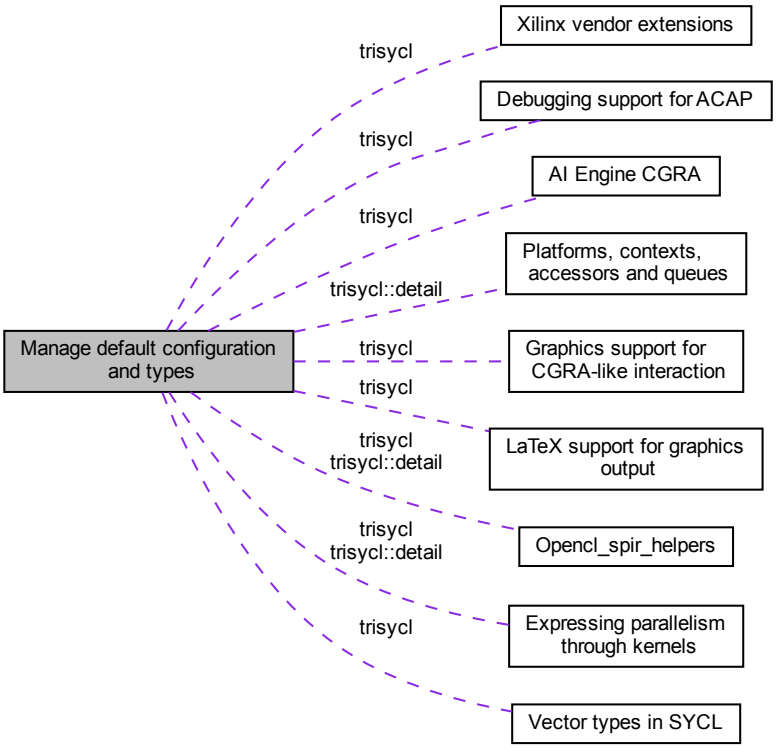
```
00133                               {
00134 #ifdef TRISYCL_TRACE_KERNEL
00135   // Inject tracing message around the kernel
00136   return [=] () mutable {
00137     /* Since the class KernelName may just be declared and not really
00138 defined, just use it through a class pointer to have
00139 typeid().name() not complaining */
00140     TRISYCL_INTERNAL_DUMP(
00141       "Kernel started "
00142       « boost::typeindex::type_id<KernelName *>().pretty_name());
00143     f();
```

```
00144     TRISYCL_INTERNAL_DUMP(
00145       "Kernel stopped "
00146       « boost::typeindex::type_id<KernelName *>().pretty_name());
00147   };
00148 #else
00149   // Identity by default
00150   return f;
00151 #endif
00152 }
```

References TRISYCL_INTERNAL_DUMP.

9.15 Manage default configuration and types

Collaboration diagram for Manage default configuration and types:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Macros

- #define CL_SYCL_LANGUAGE_VERSION 121

  This implements SYCL 1.2.1.
- #define TRISYCL_CL_LANGUAGE_VERSION 121

  This implements triSYCL 1.2.1.
- #define BOOST_COMPUTE_CL_VERSION_1_2
- #define __SYCL_SINGLE_SOURCE__

  This source is compiled by a single source compiler.
- #define TRISYCL_MAKE_BOOST_CIRCULARBUFFER_THREAD_SAFE
- #define TRISYCL_OPENCL_ONLY(...) __VA_ARGS__

  Define TRISYCL_OPENCL to add OpenCL.
- #define TRISYCL_DEVICE_ONLY(...)
- #define TRISYCL_DEBUG_ONLY(...) __VA_ARGS__

9.15.1 Detailed Description

9.15.2 Macro Definition Documentation

9.15.2.1 __SYCL_SINGLE_SOURCE__

```
#define __SYCL_SINGLE_SOURCE__
```

```
#include <include/triSYCL/detail/global_config.hpp>
```

This source is compiled by a single source compiler.

Definition at line 60 of file global_config.hpp.

9.15.2.2 BOOST_COMPUTE_CL_VERSION_1_2

```
#define BOOST_COMPUTE_CL_VERSION_1_2
```

```
#include <include/triSYCL/detail/global_config.hpp>
```

Definition at line 29 of file global_config.hpp.

9.15.2.3 CL_SYCL_LANGUAGE_VERSION

```
#define CL_SYCL_LANGUAGE_VERSION 121
```

```
#include <include/triSYCL/detail/global_config.hpp>
```

This implements SYCL 1.2.1.

Definition at line 21 of file global_config.hpp.

9.15.2.4 TRISYCL_CL_LANGUAGE_VERSION

```
#define TRISYCL_CL_LANGUAGE_VERSION 121
```

```
#include <include/triSYCL/detail/global_config.hpp>
```

This implements triSYCL 1.2.1.

Definition at line 26 of file global_config.hpp.

9.15.2.5 TRISYCL_DEBUG_ONLY

```
#define TRISYCL_DEBUG_ONLY(
              ... ) __VA_ARGS__
```

```
#include <include/triSYCL/detail/global_config.hpp>
```

Definition at line 104 of file global_config.hpp.

9.15.2.6 TRISYCL_DEVICE_ONLY

```
#define TRISYCL_DEVICE_ONLY(
              ... )
```

```
#include <include/triSYCL/detail/global_config.hpp>
```

Definition at line 98 of file global_config.hpp.

9.15.2.7 TRISYCL_MAKE_BOOST_CIRCULARBUFFER_THREAD_SAFE

```
#define TRISYCL_MAKE_BOOST_CIRCULARBUFFER_THREAD_SAFE
```

```
#include <include/triSYCL/detail/global_config.hpp>
```

Definition at line 65 of file global_config.hpp.

9.15.2.8 TRISYCL_OPENCL_ONLY

```
#define TRISYCL_OPENCL_ONLY(
              ... ) __VA_ARGS__
```

```
#include <include/triSYCL/detail/global_config.hpp>
```

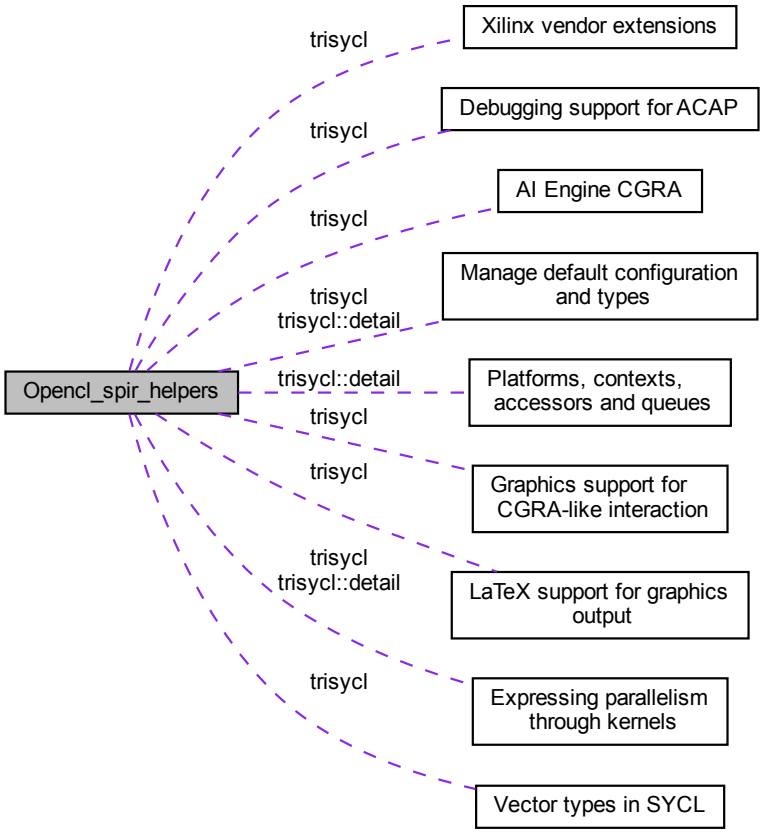
Define TRISYCL_OPENCL to add OpenCL.

triSYCL can indeed work without OpenCL if only host support is needed. A macro to keep some stuff in OpenCL mode

Definition at line 84 of file global_config.hpp.

9.16 Opencl_spir_helpers

Collaboration diagram for Opencl_spir_helpers:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.
- namespace trisycl::detail::spir

Functions

- template<typename T , typename F >
  auto trisycl::detail::spir::make_array_with_func (F unary_func)
- template<int Dimensions>
  nd_range< Dimensions > trisycl::detail::spir::make_spir_nd_range ()
- template<int Dimensions>
  h_item< Dimensions > trisycl::detail::spir::make_spir_h_item ()
- template<int Dimensions, typename Index_T >
  auto trisycl::detail::spir::create_parallel_for_arg (Index_T index_or_range)

9.16.1 Detailed Description

9.16.2 Function Documentation

9.16.2.1 create_parallel_for_arg()

```
template<int Dimensions, typename Index_T >
auto trisycl::detail::spir::create_parallel_for_arg (
             Index_T index_or_range )
```

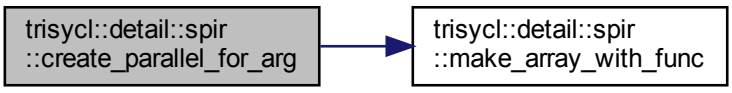
#include <include/triSYCL/detail/SPIR/opencl_spir_helpers.hpp>

Definition at line 73 of file opencl_spir_helpers.hpp.

```
00073                                                    {
00074   if constexpr (std::is_same_v<Index_T, id<Dimensions») {
00075     index_or_range = make_array_with_func<Index_T>(get_global_id);
00076   } else if constexpr (std::is_same_v<Index_T, item<Dimensions») {
00077     // Item's do not contain local information in parallel_for
00078     index_or_range = Index_T {
00079       make_array_with_func<range<Dimensions»(get_global_size),
00080       make_array_with_func<id<Dimensions»(get_global_id),
00081       make_array_with_func<id<Dimensions»(get_global_offset)
00082     };
00083   } else {
00084     static_assert(std::is_same_v<Index_T, item<Dimensions» ||
00085                   std::is_same_v<Index_T, id<Dimensions»,
00086                   "undefined Index_T passed to create_parallel_for_arg");
00087   }
00088
00089   return index_or_range;
00090 }
```

References trisycl::detail::spir::make_array_with_func().

Here is the call graph for this function:



9.16.2.2 make_array_with_func()

```
template<typename T , typename F >
auto trisycl::detail::spir::make_array_with_func (
             F unary_func )
```

```
#include <include/triSYCL/detail/SPIR/opencl_spir_helpers.hpp>
```

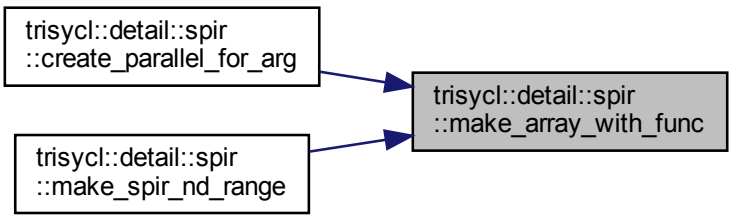
Definition at line 37 of file opencl_spir_helpers.hpp.

```
00037                                          {
00038   T id_or_range {};
00039
00040   for (size_t i = 0; i < id_or_range.dimensionality; ++i)
00041     id_or_range[i] = unary_func(i);
00042
00043   return id_or_range;
00044 }
```

References i.

Referenced by trisycl::detail::spir::create_parallel_for_arg(), and trisycl::detail::spir::make_spir_nd_range().

Here is the caller graph for this function:



9.16.2.3 make_spir_h_item()

```
template<int Dimensions>
h_item< Dimensions > trisycl::detail::spir::make_spir_h_item ( )
```

```
#include <include/triSYCL/detail/SPIR/opencl_spir_helpers.hpp>
```

Definition at line 59 of file opencl_spir_helpers.hpp.

```
00059                                         {
00060   return h_item<Dimensions> {
00061     make_array_with_func<id<Dimensions»(get_global_offset),
00062     make_spir_nd_range<Dimensions>()
00063   };
00064 }
```

9.16.2.4 make_spir_nd_range()

```
template<int Dimensions>
nd_range< Dimensions > trisycl::detail::spir::make_spir_nd_range ( )
```

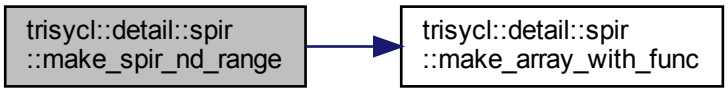
```
#include <include/triSYCL/detail/SPIR/opencl_spir_helpers.hpp>
```

Definition at line 47 of file opencl_spir_helpers.hpp.

```
00047                                          {
00048   return nd_range<Dimensions> {
00049     make_array_with_func<range<Dimensions»(get_global_size),
00050     make_array_with_func<range<Dimensions»(get_local_size),
00051     make_array_with_func<id<Dimensions»(get_global_offset)
00052   };
00053 }
```

References trisycl::detail::spir::make_array_with_func().

Here is the call graph for this function:



9.17 Device-side runtime implementation

Namespaces

- namespace trisycl::drt::code

  The binary code of the kernels.

Classes

- class trisycl::drt::accessor< Accessor >

  SYCL accessor seen from a device perspective. More...

Functions

- TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX trisycl::drt::serialize_arg (detail::task &task, std::size_t index, void *arg, std::size_t arg_size)

  Set an argument value of the kernel.
- TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX trisycl::drt::serialize_accessor_arg (detail::task &task, std::size_t index, void *arg, std::size_t arg_size)

  Set an accessor argument of the kernel.
- TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX trisycl::drt::set_kernel (detail::task &task, const char *kernel_name, const char *kernel_short_name)

  Set the kernel for a later invocation.

Variables

- TRISYCL_WEAK_ATTRIB_PREFIX std::map< std::string, std::vector< unsigned char > > TRISYCL_WEAK_ATTRIB_SUFFIX trisycl::drt::kernel_IR

  The code of the kernels arranged by name.

9.17.1 Detailed Description

9.17.2 Class Documentation

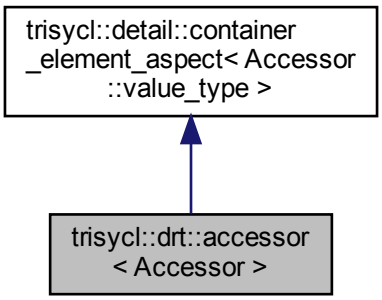
9.17.2.1 class trisycl::drt::accessor

template<typename Accessor>
class trisycl::drt::accessor< Accessor >

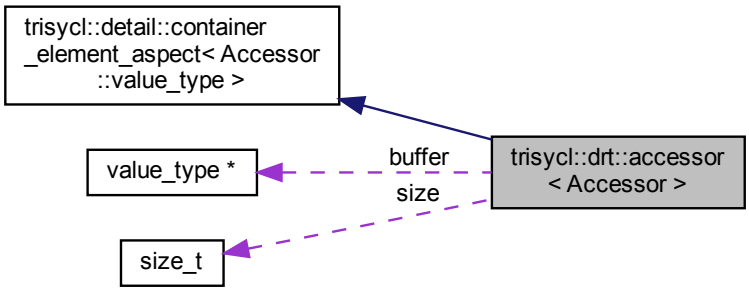
SYCL accessor seen from a device perspective.

Definition at line 49 of file device_runtime.hpp.

Inheritance diagram for trisycl::drt::accessor< Accessor >:



Collaboration diagram for trisycl::drt::accessor< Accessor >:



Public Member Functions

- accessor (Accessor &a)

  Capture the minimal accessor features.
- auto & operator[ ] (std::size_t index) const noexcept

  use the accessor with integers à la []

Private Attributes

- TRISYCL_GLOBAL_AS Accessor::value_type * buffer

  The pointer to the data.
- std::size_t size

  The size of the accessor.

Additional Inherited Members

9.17.2.1.1 Constructor & Destructor Documentation

9.17.2.1.1.1 accessor() template<typename Accessor >

```
trisycl::drt::accessor< Accessor >::accessor (
            Accessor & a )  [inline]
```

Capture the minimal accessor features.

Definition at line 73 of file device_runtime.hpp.

```
00073                         :
00074     /* Initialize the buffer with actually a \c cl_mem since it will be
00075 overridden with the kernel argument by the serializer at
00076 run-time, but at least from the kernel point of view it has to
00077 look like an \c Accessor::value_type * for the kernel
00078 outlining */
00079     buffer {
00080       reinterpret_cast<TRISYCL_GLOBAL_AS typename Accessor::value_type *>
00081         (a.implementation->get_order())
00082         },
00083     size { a.get_count() }
00084   {
00085 #ifndef TRISYCL_DEVICE
00086     /* Register the buffer address to be updated with the final
00087 version before the kernel set arg */
00088     TRISYCL_DUMP_T("drt::accessor(Accessor &a) :  &a = "
00089                    « (void *) &a « '\n'
00090                    « "\t&buffer = " « (void *) &buffer
00091                    « ", Order = " « a.implementation->get_order());
00092 #endif
00093   }
```

References a, and TRISYCL_DUMP_T.

9.17.2.1.2 Member Function Documentation

9.17.2.1.2.1 operator[]() template<typename Accessor >

```
auto & trisycl::drt::accessor< Accessor >::operator[ ] (
            std::size_t index ) const  [inline], [noexcept]
```

use the accessor with integers à la []

Definition at line 97 of file device_runtime.hpp.

```
00097                                                 {
00098     return buffer[index];
00099   }
```

9.17.2.1.3 Member Data Documentation

9.17.2.1.3.1 buffer template<typename Accessor >

```
TRISYCL_GLOBAL_AS Accessor::value_type* trisycl::drt::accessor< Accessor >::buffer  [private]
```

The pointer to the data.

For now none of the following works yet for various reasons:

```
 trisycl::global_ptr<typename Accessor::value_type> buffer;
 trisycl::global<typename Accessor::value_type *> buffer;
__global typename Accessor::pointer buffer;
```

So use a more trivial approach

Definition at line 63 of file device_runtime.hpp.

9.17.2.1.3.2 size template<typename Accessor >

```
std::size_t trisycl::drt::accessor< Accessor >::size  [private]
```

The size of the accessor.

The compiler should optimize it out if it is unused

Definition at line 68 of file device_runtime.hpp.

9.17.3 Function Documentation

9.17.3.1 serialize_accessor_arg()

```
TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX trisycl::drt::serialize_accessor_↩
arg (
            detail::task & task,
            std::size_t index,
            void * arg,
            std::size_t arg_size )
```

```
#include <include/triSYCL/device_runtime.hpp>
```

Set an accessor argument of the kernel.

Parameters

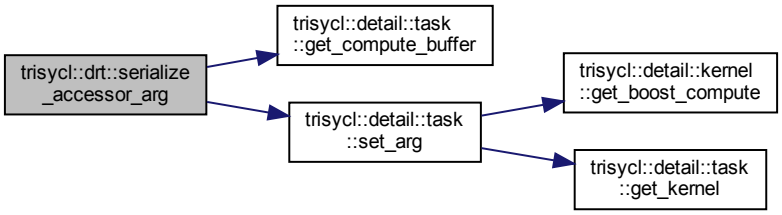
in	*task*	is the implementation detail of a triSYCL kernel
in	*index*	is the order of the argument to set (0 is the first one)
in	*arg*	points to the argument value
in	*arg_size*	is the size of the argument

Definition at line 136 of file device_runtime.hpp.

```
00139                                       {
00140   TRISYCL_DUMP_T("serialize_accessor_arg index = " « index
00141                  « ", size = " « arg_size « ", arg = " « arg);
00142 #ifdef TRISYCL_OPENCL
00143   /* The address of the argument has been actually initialized by the
00144 accessor constructor to the accessor order to relate an argument
00145 to its accessor */
00146   task.set_arg(index,
00147                task.get_compute_buffer(reinterpret_cast<std::size_t>(arg)));
00148 #endif
00149 }
```

References trisycl::detail::task::get_compute_buffer(), trisycl::detail::task::set_arg(), and TRISYCL_DUMP_T.

Here is the call graph for this function:



9.17.3.2 serialize_arg()

```
TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX trisycl::drt::serialize_arg (
            detail::task & task,
            std::size_t index,
            void * arg,
            std::size_t arg_size )
```

```
#include <include/triSYCL/device_runtime.hpp>
```

Set an argument value of the kernel.

Parameters

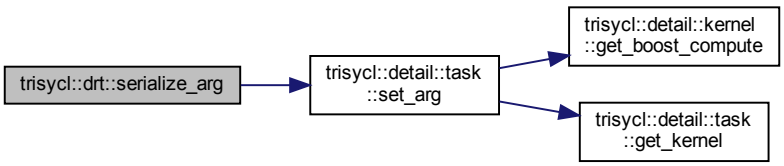
in	*task*	is the implementation detail of a triSYCL kernel
in	*index*	is the order of the argument to set (0 is the first one)
in	*arg*	points to the argument value
in	*arg_size*	is the size of the argument

Definition at line 115 of file device_runtime.hpp.

```
00118                                       {
00119   TRISYCL_DUMP_T("serialize_arg index = " « index
00120                  « ", size = " « arg_size « ", arg = " « arg);
00121   task.set_arg(index, arg_size, arg);
00122 }
```

References trisycl::detail::task::set_arg(), and TRISYCL_DUMP_T.

Here is the call graph for this function:



9.17.3.3 set_kernel()

```
TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX trisycl::drt::set_kernel (
            detail::task & task,
            const char * kernel_name,
            const char * kernel_short_name )
```

`#include <include/triSYCL/device_runtime.hpp>`

Set the kernel for a later invocation.

Parameters

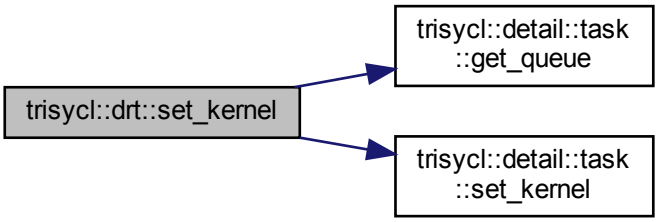
in	*task*	is the implementation detail of a triSYCL kernel
in	*kernel_name*	is set by the device compiler to identify the kernel to call

Definition at line 198 of file device_runtime.hpp.

```
00200                                   {
00201   TRISYCL_DUMP_T("set_kernel setting up " « kernel_name
00202                  « "\n\taka " « kernel_short_name);
00203   // \todo Add kernel caching per device
00204   // auto binary = kernel_IR.find(kernel_name);
00205   // if (binary == kernel_IR.end()) {
00206 #ifdef TRISYCL_OPENCL
00207   auto context = task.get_queue()->get_boost_compute().get_context();
00208   // Construct an OpenCL program from the precompiled kernel file
00209   auto program = boost::compute::program::create_with_binary
00210     (code::program::p->binary,
00211      code::program::p->binary_size,
00212      context);
00213   TRISYCL_DUMP_T("...on device with name "
00214                  « task.get_queue()->get_boost_compute().get_device().name());
00215   // Build the OpenCL program
00216   program.build();
00217
00218   // Build a SYCL kernel from the OpenCL kernel
00219   trisycl::kernel k { boost::compute::kernel { program, kernel_short_name } };
00220
00221   task.set_kernel(k.implementation);
00222 #endif
00223 }
```

References trisycl::detail::task::get_queue(), trisycl::drt::code::program::p, trisycl::detail::task::set_kernel(), and TRISYCL_DUMP_T.

Here is the call graph for this function:



9.17.4 Variable Documentation

9.17.4.1 kernel_IR

```
TRISYCL_WEAK_ATTRIB_PREFIX std::map<std::string, std::vector<unsigned char> > TRISYCL_WEAK_ATTRIB_SUFFIX
trisycl::drt::kernel_IR
```

```
#include <include/triSYCL/device_runtime.hpp>
```

The code of the kernels arranged by name.

Todo use std::byte when moving to C++17

Definition at line 187 of file device_runtime.hpp.

9.18 Error handling

Namespaces

- namespace trisycl::trisycl

Classes

- struct trisycl::error_handler

  User supplied error handler to call a user-provided function when an error happens from a SYCL object that was constructed with this error handler. More...
- struct trisycl::exception_list

  Exception list to store several exceptions. More...
- class trisycl::exception

  Encapsulate a SYCL error information. More...
- class trisycl::cl_exception

  Returns the OpenCL error code encapsulated in the exception. More...
- struct trisycl::async_exception

  An error stored in an exception_list for asynchronous errors. More...
- class trisycl::runtime_error
- class trisycl::kernel_error

  Error that occurred before or while enqueuing the SYCL kernel. More...
- class trisycl::accessor_error

  Error regarding the trisycl::accessor objects defined. More...
- class trisycl::nd_range_error

  Error regarding the trisycl::nd_range specified for the SYCL kernel. More...
- class trisycl::event_error

  Error regarding associated trisycl::event objects. More...
- class trisycl::invalid_parameter_error

  Error regarding parameters to the SYCL kernel, it may apply to any captured parameters to the kernel lambda. More...
- class trisycl::device_error

  The SYCL device will trigger this exception on error. More...
- class trisycl::compile_program_error

  Error while compiling the SYCL kernel to a SYCL device. More...
- class trisycl::link_program_error

  Error while linking the SYCL kernel to a SYCL device. More...
- class trisycl::invalid_object_error

  Error regarding any memory objects being used inside the kernel. More...
- class trisycl::memory_allocation_error

  Error on memory allocation on the SYCL device for a SYCL kernel. More...
- class trisycl::pipe_error

  A failing pipe error will trigger this exception on error. More...
- class trisycl::platform_error

  The SYCL platform will trigger this exception on error. More...
- class trisycl::profiling_error

  The SYCL runtime will trigger this error if there is an error when profiling info is enabled. More...
- class trisycl::feature_not_supported

  Exception thrown when an optional feature or extension is used in a kernel but its not available on the device the SYCL kernel is being enqueued on. More...
- class trisycl::non_cl_error

  Exception for an OpenCL operation requested in a non OpenCL area. More...

Typedefs

- using trisycl::exception_ptr = std::exception_ptr

  A shared pointer to an exception as in C++ specification.
- using trisycl::async_handler = function_class< void, exception_list >

9.18.1 Detailed Description

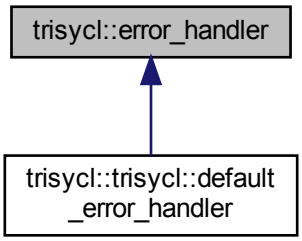
9.18.2 Class Documentation

9.18.2.1 struct trisycl::error_handler

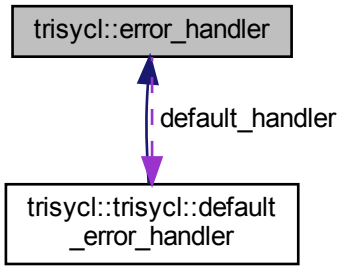
User supplied error handler to call a user-provided function when an error happens from a SYCL object that was constructed with this error handler.

Definition at line 31 of file error_handler.hpp.

Inheritance diagram for trisycl::error_handler:



Collaboration diagram for trisycl::error_handler:



Public Member Functions

- virtual void report_error (exception &error)=0

  The method to define to be called in the case of an error.
- virtual ~error_handler ()=0

Static Public Attributes

- static trisycl::default_error_handler default_handler

  Add a default_handler to be used by default.

9.18.2.1.1 Constructor & Destructor Documentation

9.18.2.1.1.1 ~error_handler() `virtual trisycl::error_handler::~error_handler ( ) [pure virtual]`

9.18.2.1.2 Member Function Documentation

9.18.2.1.2.1 report_error() `virtual void trisycl::error_handler::report_error ( exception & error ) [pure virtual]`

The method to define to be called in the case of an error.

Todo Add "virtual void" to the specification

Implemented in trisycl::trisycl::default_error_handler.

9.18.2.1.3 Member Data Documentation

9.18.2.1.3.1 default_handler `trisycl::default_error_handler trisycl::error_handler::default_handler [static]`

Add a default_handler to be used by default.

Todo add this concept to the specification?

Definition at line 42 of file error_handler.hpp.

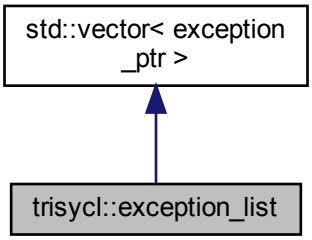
9.18.2.2 struct trisycl::exception_list

Exception list to store several exceptions.

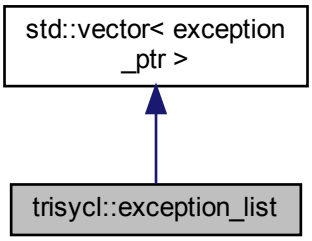
Todo Do we need to define it in SYCL or can we rely on plain C++17 one?

Definition at line 32 of file exception.hpp.

Inheritance diagram for trisycl::exception_list:



Collaboration diagram for trisycl::exception_list:

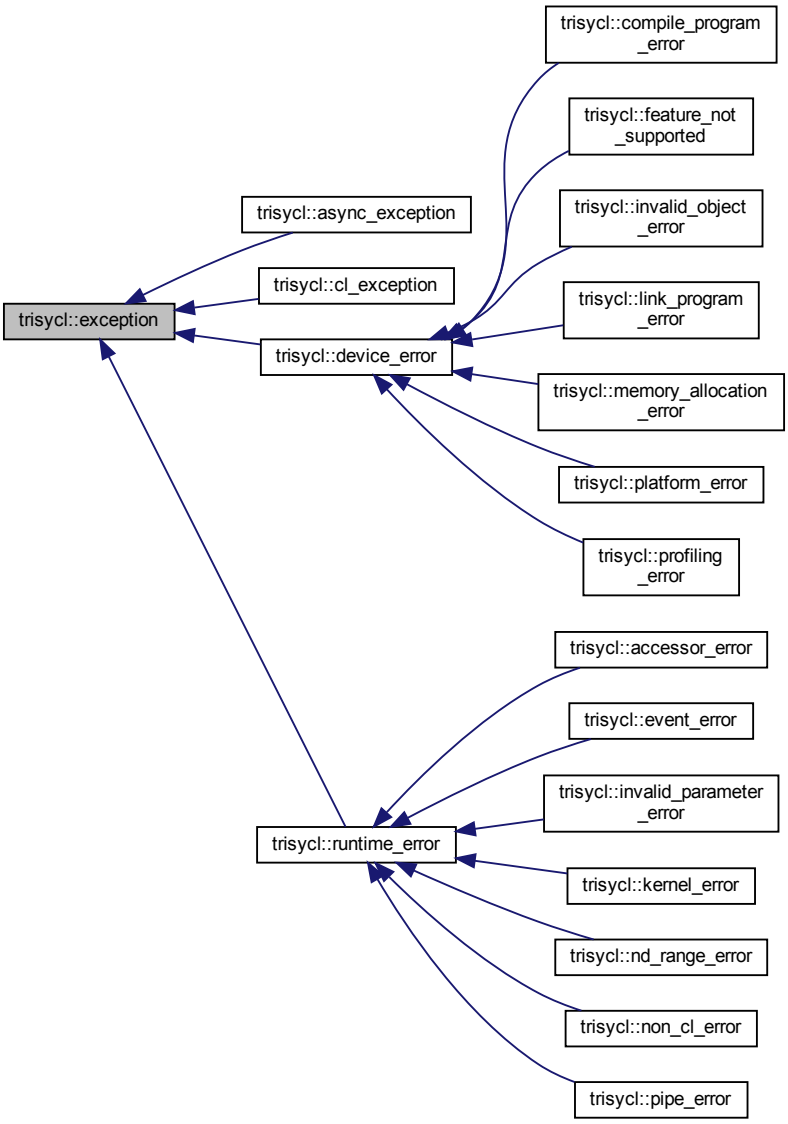


9.18.2.3 class trisycl::exception

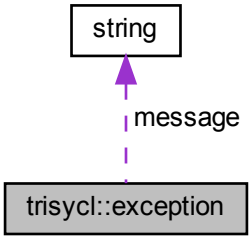
Encapsulate a SYCL error information.

Definition at line 40 of file exception.hpp.

Inheritance diagram for trisycl::exception:



Collaboration diagram for trisycl::exception:



Public Member Functions

- exception (const string_class &message)

  Construct an exception with a message for internal use.
- string_class what () const

  Returns a descriptive string for the error, if available.

Private Attributes

- string_class message

  The error message to return.

9.18.2.3.1 Constructor & Destructor Documentation

9.18.2.3.1.1 exception() `trisycl::exception::exception ( const string_class & message ) [inline]`

Construct an exception with a message for internal use.

Definition at line 48 of file exception.hpp.

```
00048 :  message { message } {}
```

9.18.2.3.2 Member Function Documentation

9.18.2.3.2.1 what() `string_class trisycl::exception::what ( ) const [inline]`

Returns a descriptive string for the error, if available.

Definition at line 51 of file exception.hpp.

```
00051                              {
00052     return message;
00053   }
```

References message.

9.18.2.3.3 Member Data Documentation

9.18.2.3.3.1 message `string_class trisycl::exception::message [private]`

The error message to return.

Definition at line 43 of file exception.hpp.

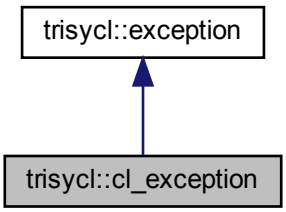
Referenced by what().

9.18.2.4 class trisycl::cl_exception

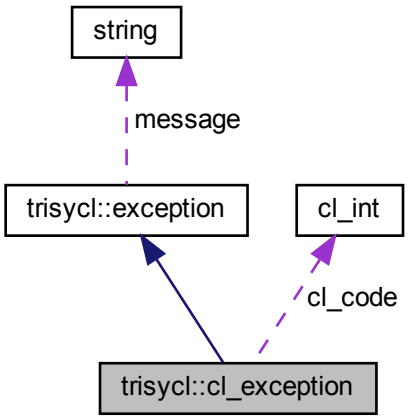
Returns the OpenCL error code encapsulated in the exception.

Definition at line 68 of file exception.hpp.

Inheritance diagram for trisycl::cl_exception:



Collaboration diagram for trisycl::cl_exception:



Public Member Functions

- cl_exception (const string_class &message, cl_int cl_code)

  Construct an exception with a message and OpenCL error code for internal use.
- cl_int get_cl_code () const

Private Attributes

- cl_int cl_code

  The OpenCL error code to return.

9.18.2.4.1 Constructor & Destructor Documentation

9.18.2.4.1.1 cl_exception() `trisycl::cl_exception::cl_exception ( const string_class & message, cl_int cl_code ) [inline]`

Construct an exception with a message and OpenCL error code for internal use.

Definition at line 79 of file exception.hpp.

```
00080     :  exception { message }, cl_code { cl_code } {}
```

9.18.2.4.2 Member Function Documentation

9.18.2.4.2.1 get_cl_code() `cl_int trisycl::cl_exception::get_cl_code ( ) const [inline]`

Definition at line 83 of file exception.hpp.

```
00083                                {
00084     return cl_code;
00085   }
```

References cl_code.

9.18.2.4.3 Member Data Documentation

9.18.2.4.3.1 cl_code `cl_int trisycl::cl_exception::cl_code [private]`

The OpenCL error code to return.

Definition at line 73 of file exception.hpp.

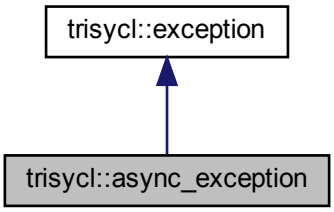
Referenced by get_cl_code().

9.18.2.5 struct trisycl::async_exception

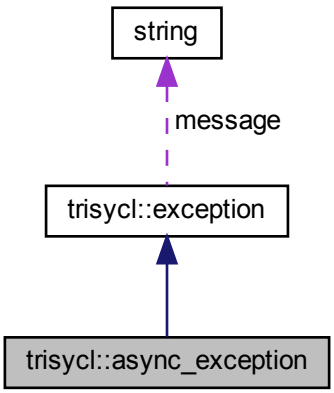
An error stored in an exception_list for asynchronous errors.

Definition at line 92 of file exception.hpp.

Inheritance diagram for trisycl::async_exception:



Collaboration diagram for trisycl::async_exception:



Public Member Functions

- exception (const string_class &message)

  Construct an exception with a message for internal use.

9.18.2.5.1 Member Function Documentation

9.18.2.5.1.1 exception() `trisycl::exception::exception ( const string_class & message ) [inline]`

Construct an exception with a message for internal use.

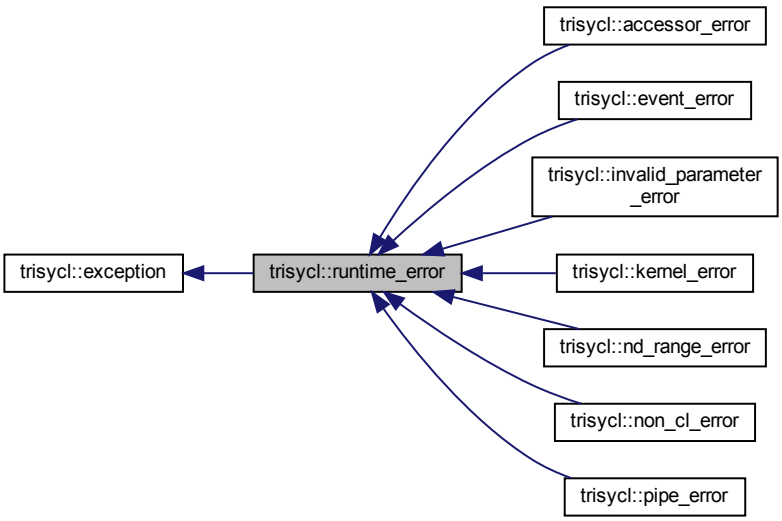
Definition at line 48 of file exception.hpp.

```
00048 :  message { message } {}
```

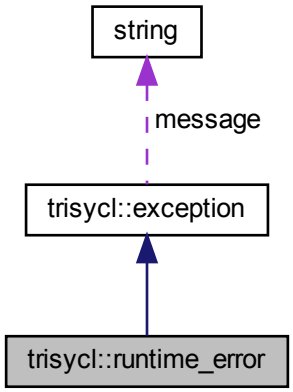
9.18.2.6 class trisycl::runtime_error

Definition at line 97 of file exception.hpp.

Inheritance diagram for trisycl::runtime_error:



Collaboration diagram for trisycl::runtime_error:



Private Member Functions

- exception (const string_class &message)

  Construct an exception with a message for internal use.

Additional Inherited Members

9.18.2.6.1 Member Function Documentation

9.18.2.6.1.1 exception() `trisycl::exception::exception (const string_class & message ) [inline], [private]`

Construct an exception with a message for internal use.

Definition at line 48 of file exception.hpp.

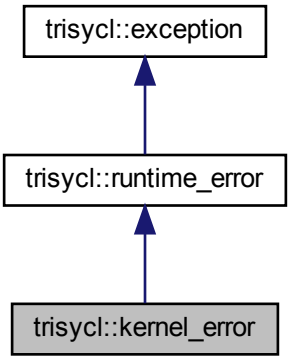
```
00048 : message { message } {}
```

9.18.2.7 class trisycl::kernel_error

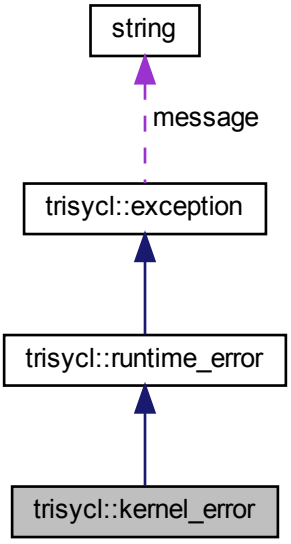
Error that occurred before or while enqueuing the SYCL kernel.

Definition at line 103 of file exception.hpp.

Inheritance diagram for trisycl::kernel_error:



Collaboration diagram for trisycl::kernel_error:



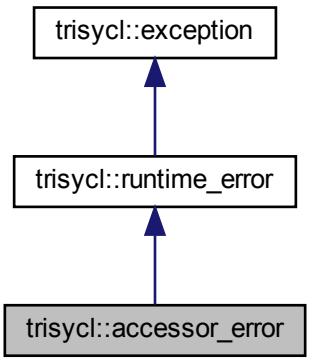
Additional Inherited Members

9.18.2.8 class trisycl::accessor_error

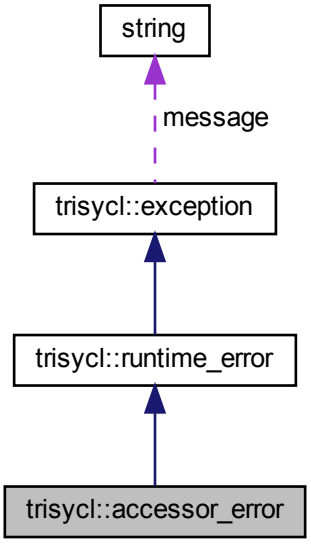
Error regarding the trisycl::accessor objects defined.

Definition at line 109 of file exception.hpp.

Inheritance diagram for trisycl::accessor_error:



Collaboration diagram for trisycl::accessor_error:



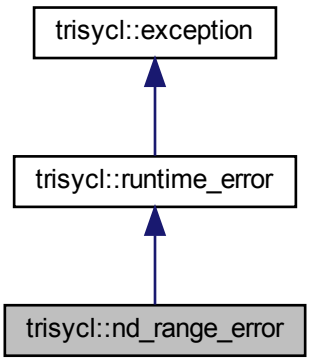
Additional Inherited Members

9.18.2.9 class trisycl::nd_range_error

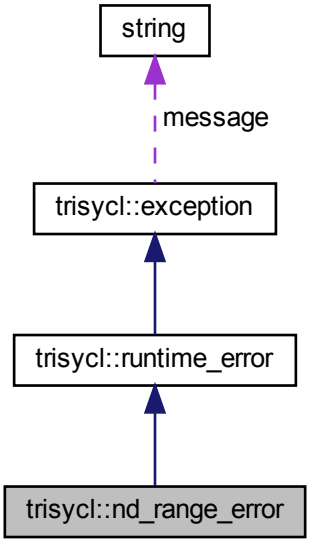
Error regarding the trisycl::nd_range specified for the SYCL kernel.

Definition at line 115 of file exception.hpp.

Inheritance diagram for trisycl::nd_range_error:



Collaboration diagram for trisycl::nd_range_error:



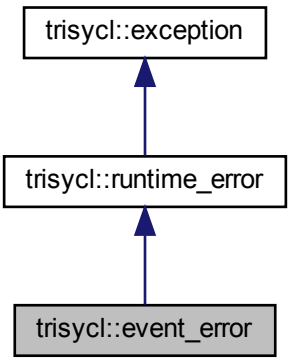
Additional Inherited Members

9.18.2.10 class trisycl::event_error

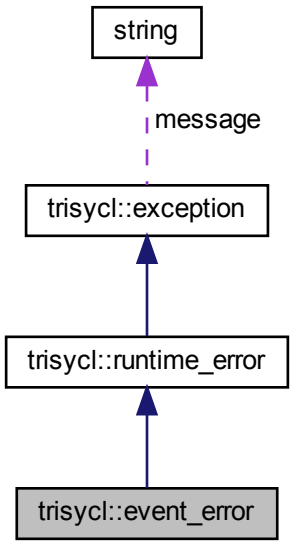
Error regarding associated trisycl::event objects.

Definition at line 121 of file exception.hpp.

Inheritance diagram for trisycl::event_error:



Collaboration diagram for trisycl::event_error:



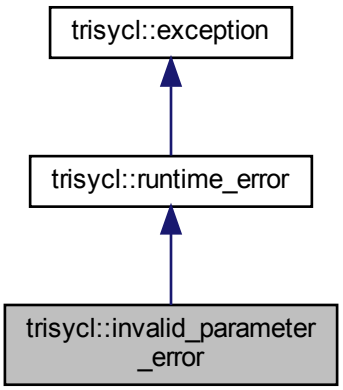
Additional Inherited Members

9.18.2.11 class trisycl::invalid_parameter_error

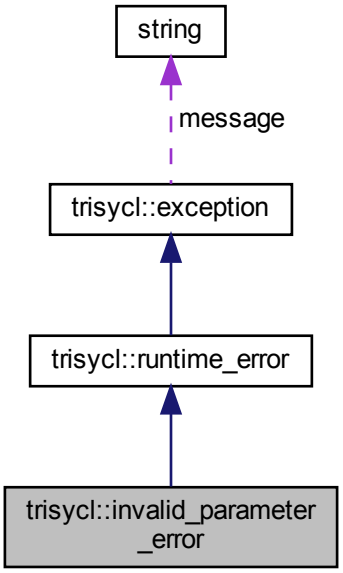
Error regarding parameters to the SYCL kernel, it may apply to any captured parameters to the kernel lambda.

Definition at line 129 of file exception.hpp.

Inheritance diagram for trisycl::invalid_parameter_error:



Collaboration diagram for trisycl::invalid_parameter_error:



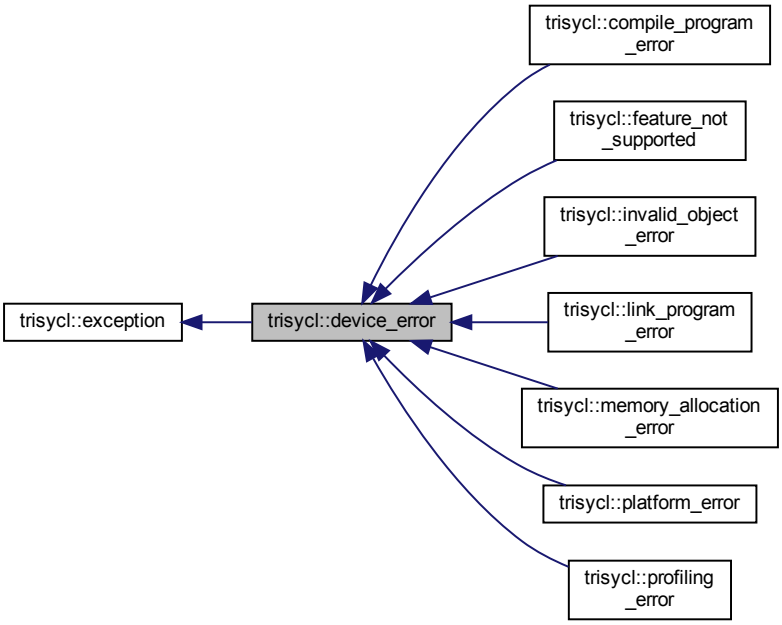
Additional Inherited Members

9.18.2.12 class trisycl::device_error

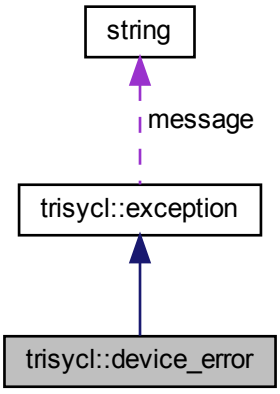
The SYCL device will trigger this exception on error.

Definition at line 135 of file exception.hpp.

Inheritance diagram for trisycl::device_error:



Collaboration diagram for trisycl::device_error:



Private Member Functions

- exception (const string_class &message)

  Construct an exception with a message for internal use.

Additional Inherited Members

9.18.2.12.1 Member Function Documentation

9.18.2.12.1.1 exception() `trisycl::exception::exception (`

`const string_class & message ) [inline], [private]`

Construct an exception with a message for internal use.

Definition at line 48 of file exception.hpp.

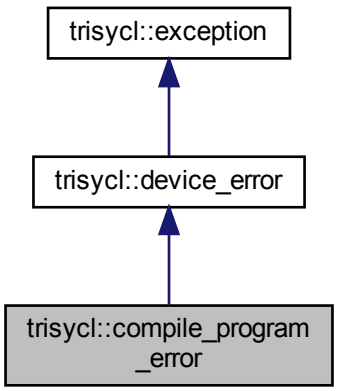
```
00048 :  message { message } {}
```

9.18.2.13 class trisycl::compile_program_error

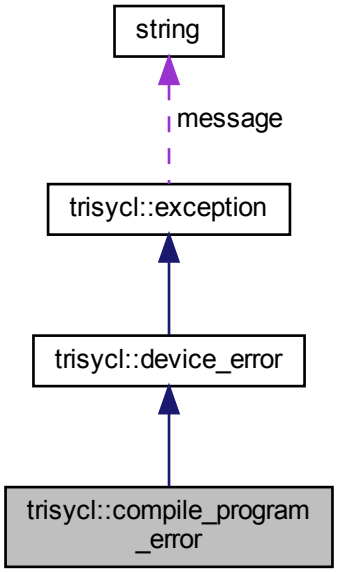
Error while compiling the SYCL kernel to a SYCL device.

Definition at line 141 of file exception.hpp.

Inheritance diagram for trisycl::compile_program_error:



Collaboration diagram for trisycl::compile_program_error:



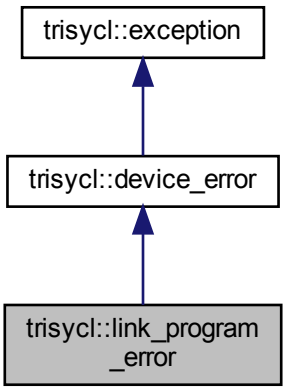
Additional Inherited Members

9.18.2.14 class trisycl::link_program_error

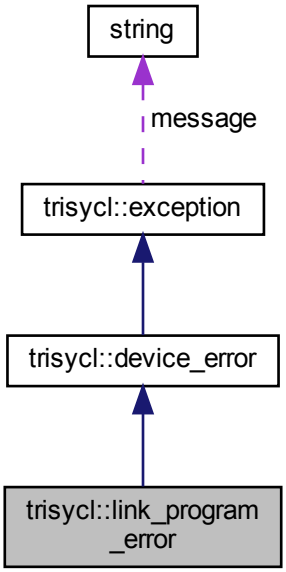
Error while linking the SYCL kernel to a SYCL device.

Definition at line 147 of file exception.hpp.

Inheritance diagram for trisycl::link_program_error:



Collaboration diagram for trisycl::link_program_error:



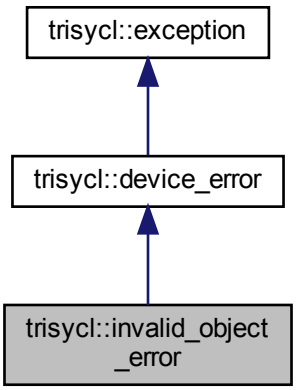
Additional Inherited Members

9.18.2.15 class trisycl::invalid_object_error

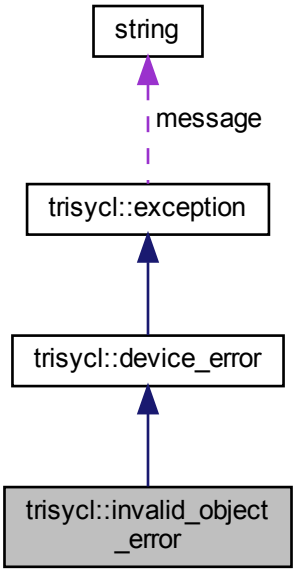
Error regarding any memory objects being used inside the kernel.

Definition at line 153 of file exception.hpp.

Inheritance diagram for trisycl::invalid_object_error:



Collaboration diagram for trisycl::invalid_object_error:



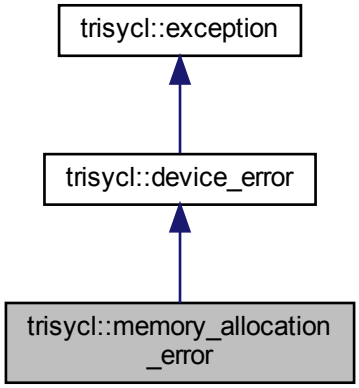
Additional Inherited Members

9.18.2.16 class trisycl::memory_allocation_error

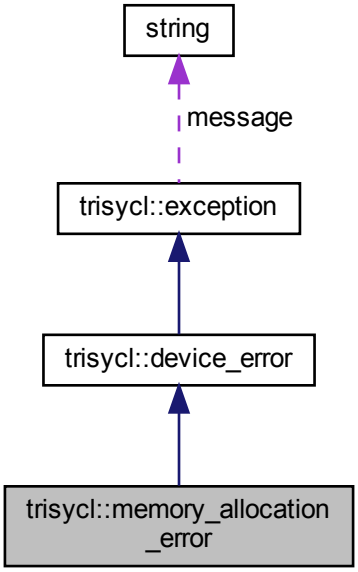
Error on memory allocation on the SYCL device for a SYCL kernel.

Definition at line 159 of file exception.hpp.

Inheritance diagram for trisycl::memory_allocation_error:



Collaboration diagram for trisycl::memory_allocation_error:



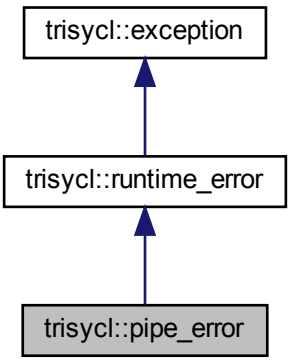
Additional Inherited Members

9.18.2.17 class trisycl::pipe_error

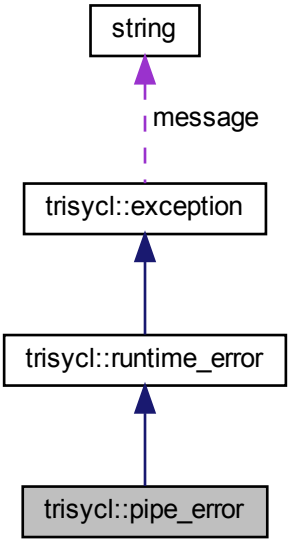
A failing pipe error will trigger this exception on error.

Definition at line 165 of file exception.hpp.

Inheritance diagram for trisycl::pipe_error:



Collaboration diagram for trisycl::pipe_error:



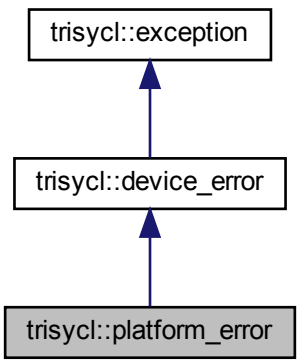
Additional Inherited Members

9.18.2.18 class trisycl::platform_error

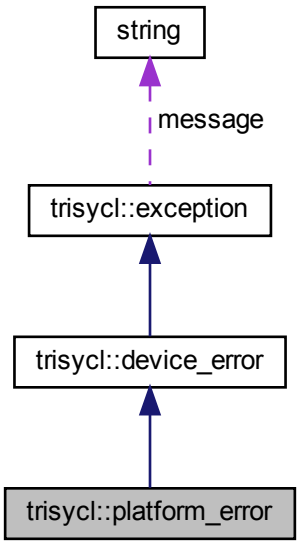
The SYCL platform will trigger this exception on error.

Definition at line 171 of file exception.hpp.

Inheritance diagram for trisycl::platform_error:



Collaboration diagram for trisycl::platform_error:



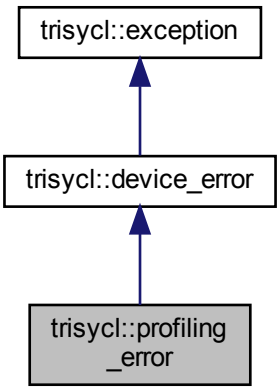
Additional Inherited Members

9.18.2.19 class trisycl::profiling_error

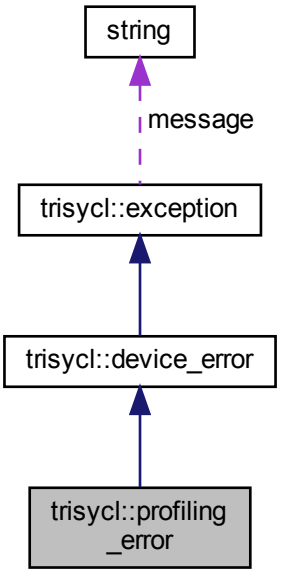
The SYCL runtime will trigger this error if there is an error when profiling info is enabled.

Definition at line 179 of file exception.hpp.

Inheritance diagram for trisycl::profiling_error:



Collaboration diagram for trisycl::profiling_error:



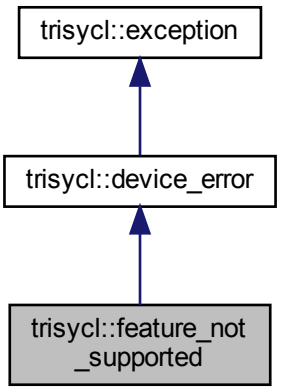
Additional Inherited Members

9.18.2.20 class trisycl::feature_not_supported

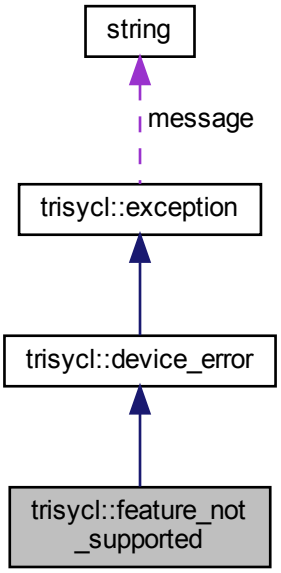
Exception thrown when an optional feature or extension is used in a kernel but its not available on the device the SYCL kernel is being enqueued on.

Definition at line 188 of file exception.hpp.

Inheritance diagram for trisycl::feature_not_supported:



Collaboration diagram for trisycl::feature_not_supported:



Additional Inherited Members

9.18.2.21 class trisycl::non_cl_error

Exception for an OpenCL operation requested in a non OpenCL area.

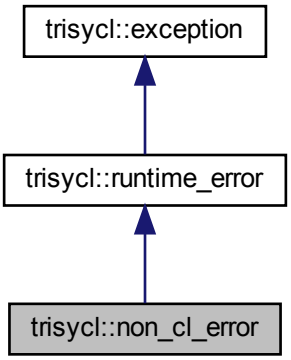
Todo Add to the specification

Todo Clean implementation

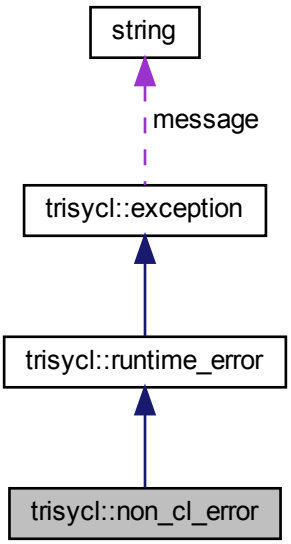
Todo Exceptions are named error in C++

Definition at line 201 of file exception.hpp.

Inheritance diagram for trisycl::non_cl_error:



Collaboration diagram for trisycl::non_cl_error:



Additional Inherited Members

9.18.3 Typedef Documentation

9.18.3.1 async_handler

```
using trisycl::async_handler = typedef function_class<void, exception_list>
```

```
#include <include/triSYCL/exception.hpp>
```

Definition at line 36 of file exception.hpp.

9.18.3.2 exception_ptr

```
using trisycl::exception_ptr = typedef std::exception_ptr
```

```
#include <include/triSYCL/exception.hpp>
```

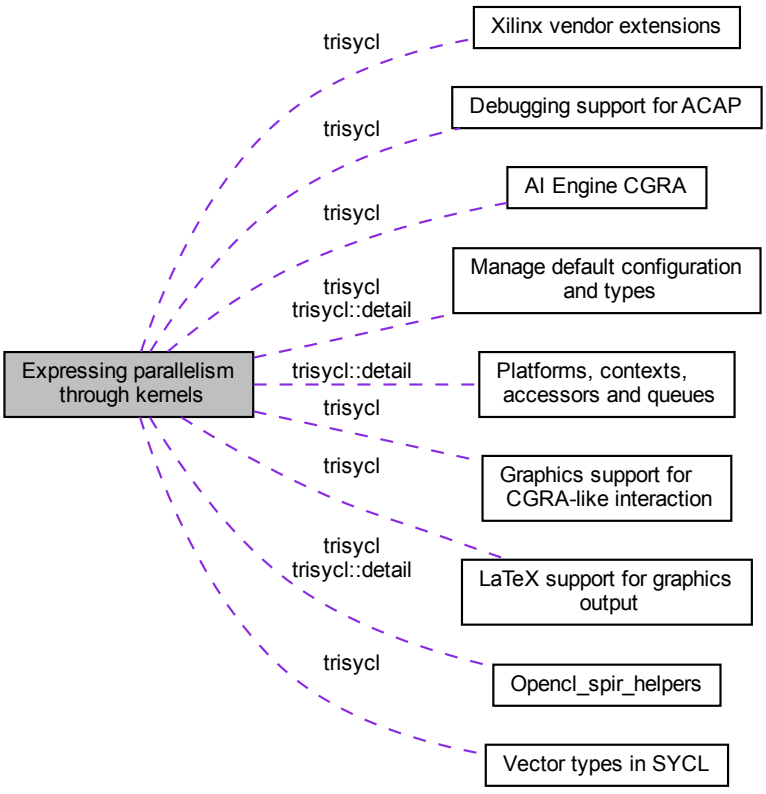
A shared pointer to an exception as in C++ specification.

Todo Do we need this instead of reusing directly the one from C++11?

Definition at line 25 of file exception.hpp.

9.19 Expressing parallelism through kernels

Collaboration diagram for Expressing parallelism through kernels:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Classes

- struct trisycl::group< Dimensions >

  A group index used in a parallel_for_workitem to specify a work_group. More...
- struct trisycl::h_item< Dimensions >

  `h_item<int dimensions>` identifies an instance of a `group::parallel_for_work_item` function object executing at each point in a local `range<int dimensions>` passed to a `parallel_for_work_item` call or to the corresponding `parallel_for_work_group` call if no range is passed to the `parallel_for_work_item` call. More...
- class trisycl::id< Dimensions >

  Define a multi-dimensional index, used for example to locate a work item. More...
- class trisycl::item< Dimensions, with_offset >

  A SYCL item stores information on a work-item with some more context such as the definition range and offset. More...
- struct trisycl::nd_item< Dimensions >

  A SYCL nd_item stores information on a work-item within a work-group, with some more context such as the definition ranges. More...
- struct trisycl::nd_range< Dimensions >

  A ND-range, made by a global and local range, to specify work-group and work-item organization. More...
- struct trisycl::detail::parallel_for_iterate< level, Range, ParallelForFunctor, Id >

  A recursive multi-dimensional iterator that ends up calling f. More...
- struct trisycl::detail::parallel_for_iterate< 0, Range, ParallelForFunctor, Id >

  Stop the recursion when level reaches 0 by simply calling the kernel functor with the constructed id. More...
- class trisycl::range< Dimensions >

  A SYCL range defines a multi-dimensional index range that can be used to define launch parallel computation extent or buffer sizes. More...

Functions

- template<typename... BasicType>
  trisycl::id (BasicType... Args) -> id< sizeof...(Args)>

  User-defined deduction guides to deduce the rank of an id from the number of arguments.
- template<int Dimensions = 1, typename ParallelForFunctor , typename Id >
  void trisycl::detail::parallel_for (range< Dimensions > r, ParallelForFunctor f, Id)

  Implementation of a data parallel computation with parallelism specified at launch time by a range<>.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for (range< Dimensions > r, ParallelForFunctor f, item< Dimensions >)

  Implementation of a data parallel computation with parallelism specified at launch time by a range<>.
- template<typename F , typename R , typename A >
  auto trisycl::detail::capture_arg_v (R(F::*)(A))
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for (range< Dimensions > r, ParallelForFunctor f)

  Calls the appropriate ternary parallel_for overload based on the index type of the kernel function object f.

- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for_global_offset (range< Dimensions > global_size, id< Dimensions > offset, ParallelForFunctor f)

  Implementation of parallel_for with a range<> and an offset.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for_workgroup (nd_range< Dimensions > r, ParallelForFunctor f)

  Implement the loop on the work-groups.
- template<int Dimensions, typename T_Item , typename ParallelForFunctor >
  void trisycl::detail::parallel_for_workitem (const group< Dimensions > &g, ParallelForFunctor f)

  Implement the loop on the work-items inside a work-group.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for (nd_range< Dimensions > r, ParallelForFunctor f)

  Implement a variation of parallel_for to take into account a nd_range<>
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for_workitem_in_group (const group< Dimensions > &g, ParallelForFunctor f)

  Implement the loop on the work-items inside a work-group.
- static auto trisycl::detail::to_tbb_range (const range< 1 > &r)
- static auto trisycl::detail::to_tbb_range (const range< 2 > &r)
- static auto trisycl::detail::to_tbb_range (const range< 3 > &r)
- static auto trisycl::detail::to_sycl_id (const tbb::blocked_range< size_t > &in)
- static auto trisycl::detail::to_sycl_id (const tbb::blocked_range2d< size_t > &in)
- static auto trisycl::detail::to_sycl_id (const tbb::blocked_range3d< size_t > &in)
- template<typename Range , typename ParallelForFunctor >
  void trisycl::detail::parallel_for_iterate (Range r, ParallelForFunctor &f)
- template<typename... BasicType>
  trisycl::range (BasicType... Args) -> range< sizeof...(Args)>

  User-defined deduction guides to deduce the rank of a range from the number of arguments.

9.19.1 Detailed Description

9.19.2 Class Documentation

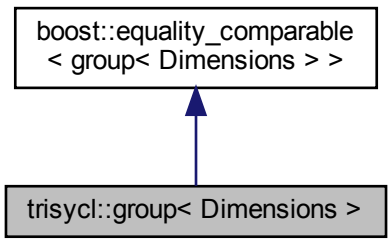
9.19.2.1 struct trisycl::group

template<int Dimensions>
struct trisycl::group< Dimensions >

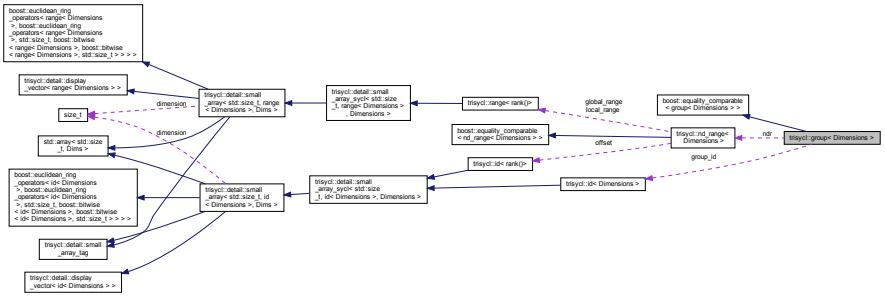
A group index used in a parallel_for_workitem to specify a work_group.

Definition at line 41 of file group.hpp.

Inheritance diagram for trisycl::group< Dimensions >:



Collaboration diagram for trisycl::group< Dimensions >:



Public Member Functions

- group (const nd_range< Dimensions > &ndr)

  Create a group from an nd_range<> with a 0 id<>
- group (const id< Dimensions > &i, const nd_range< Dimensions > &ndr)

  Create a group from an id and a nd_range<>
- group ()=default

  To be able to copy and assign group, use default constructors too.
- id< Dimensions > get_id () const

  Return an id representing the index of the group within the nd_range for every dimension.
- size_t get_id (int dimension) const

  Return the index of the group in the given dimension.
- auto & operator[ ] (int dimension)

  Return the index of the group in the given dimension within the nd_range<>
- range< Dimensions > get_group_range () const

  Return a range<> representing the dimensions of the current group.
- size_t get_group_range (int dimension) const

  Return element dimension from the con stituent group range.
- range< Dimensions > get_global_range () const

  Get the local range for this work_group.
- size_t get_global_range (int dimension) const

  Return element dimension from the constituent global range.
- range< Dimensions > get_local_range () const

  Get the local range for this work_group.
- size_t get_local_range (int dimension) const

  Return element dimension from the constituent local range.
- id< Dimensions > get_offset () const

  Get the offset of the NDRange.
- size_t get_offset (int dimension) const

  Get the offset of the NDRange.
- nd_range< Dimensions > get_nd_range () const
- size_t get_linear_id () const

  Get a linearized version of the group ID.
- void parallel_for_work_item (std::function< void(h_item< rank()>)> f) const

  Loop on the work-items inside a work-group.
- bool operator== (const group &groupB) const

Static Public Member Functions

- static auto constexpr rank ()

  The number of dimensions of the group.

Private Attributes

- id< Dimensions > group_id

  The coordinate of the group item.
- nd_range< Dimensions > ndr

  Keep a reference on the nd_range to serve potential query on it.

9.19.2.1.1 Constructor & Destructor Documentation

9.19.2.1.1.1 group() [1/3] template<int Dimensions>

```
trisycl::group< Dimensions >::group (
            const nd_range< Dimensions > & ndr )  [inline]
```

Create a group from an nd_range<> with a 0 id<>

Todo This should be private since it is only used by the triSYCL implementation

Definition at line 60 of file group.hpp.

```
00060 :  ndr { ndr } {}
```

9.19.2.1.1.2 group() [2/3] template<int Dimensions>

```
trisycl::group< Dimensions >::group (
            const id< Dimensions > & i,
            const nd_range< Dimensions > & ndr )  [inline]
```

Create a group from an id and a nd_range<>

Todo This should be private somehow, but it is used by the validation infrastructure

Definition at line 68 of file group.hpp.

```
00068                                                                 :
00069     group_id { i }, ndr { ndr } {}
```

9.19.2.1.1.3 group() [3/3] template<int Dimensions>

```
trisycl::group< Dimensions >::group ( )  [default]
```

To be able to copy and assign group, use default constructors too.

Todo Make most of them protected, reserved to implementation

9.19.2.1.2 Member Function Documentation

9.19.2.1.2.1 get_global_range() [1/2] template<int Dimensions>

range< Dimensions > trisycl::group< Dimensions >::get_global_range ( ) const [inline]

Get the local range for this work_group.

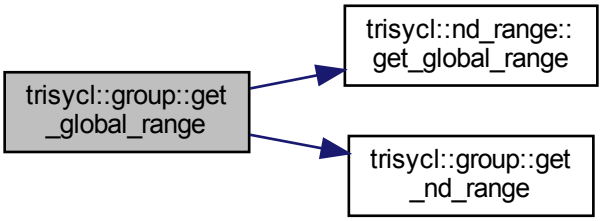
Definition at line 121 of file group.hpp.

```
00121                                                {
00122     return get_nd_range().get_global_range();
00123   }
```

References trisycl::nd_range< Dimensions >::get_global_range(), and trisycl::group< Dimensions >::get_nd_range().

Referenced by trisycl::group< Dimensions >::get_global_range().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.2.1.2.2 get_global_range() [2/2] template<int Dimensions>

```
size_t trisycl::group< Dimensions >::get_global_range (
            int dimension ) const  [inline]
```

Return element dimension from the constituent global range.

Definition at line 127 of file group.hpp.

```
00127                                          {
00128     return get_global_range()[dimension];
00129   }
```

References trisycl::group< Dimensions >::get_global_range().

Here is the call graph for this function:



9.19.2.1.2.3 get_group_range() [1/2] template<int Dimensions>

```
range< Dimensions > trisycl::group< Dimensions >::get_group_range ( ) const  [inline]
```

Return a range<> representing the dimensions of the current group.

This local range may have been provided by the programmer, or chosen by the runtime.

Todo Fix this comment and the specification

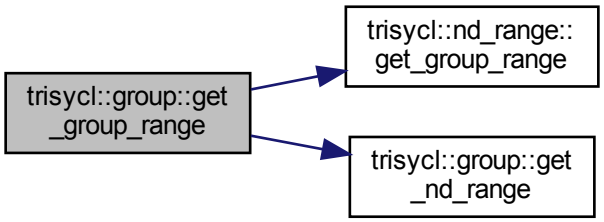
Definition at line 109 of file group.hpp.

```
00109                                        {
00110     return get_nd_range().get_group_range();
00111   }
```

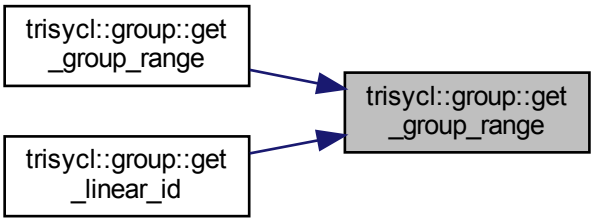
References trisycl::nd_range< Dimensions >::get_group_range(), and trisycl::group< Dimensions >::get_nd_range().

Referenced by trisycl::group< Dimensions >::get_group_range(), and trisycl::group< Dimensions >::get_linear_id().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.2.1.2.4 get_group_range() [2/2] template<int Dimensions>

```
size_t trisycl::group< Dimensions >::get_group_range (
            int dimension ) const  [inline]
```

Return element dimension from the con stituent group range.

Definition at line 115 of file group.hpp.

```
00115                                          {
00116     return get_group_range()[dimension];
00117   }
```

References trisycl::group< Dimensions >::get_group_range().

Here is the call graph for this function:



9.19.2.1.2.5 get_id() [1/2] template<int Dimensions>

```
id< Dimensions > trisycl::group< Dimensions >::get_id ( ) const  [inline]
```

Return an id representing the index of the group within the nd_range for every dimension.

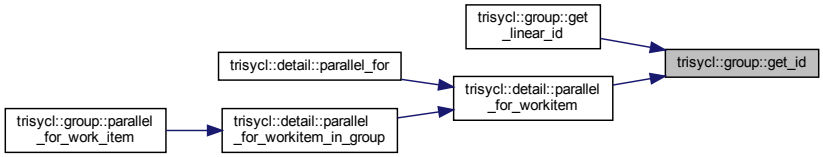
Definition at line 82 of file group.hpp.

```
00082 { return group_id; }
```

References trisycl::group< Dimensions >::group_id.

Referenced by trisycl::group< Dimensions >::get_linear_id(), and trisycl::detail::parallel_for_workitem().

Here is the caller graph for this function:



9.19.2.1.2.6 get_id() [2/2] template<int Dimensions>

```
size_t trisycl::group< Dimensions >::get_id (
            int dimension ) const  [inline]
```

Return the index of the group in the given dimension.

Definition at line 86 of file group.hpp.

```
00086 { return get_id()[dimension]; }
```

References trisycl::group< Dimensions >::get_id().

Referenced by trisycl::group< Dimensions >::get_id().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.2.1.2.7 get_linear_id() template<int Dimensions>

```
size_t trisycl::group< Dimensions >::get_linear_id ( ) const  [inline]
```

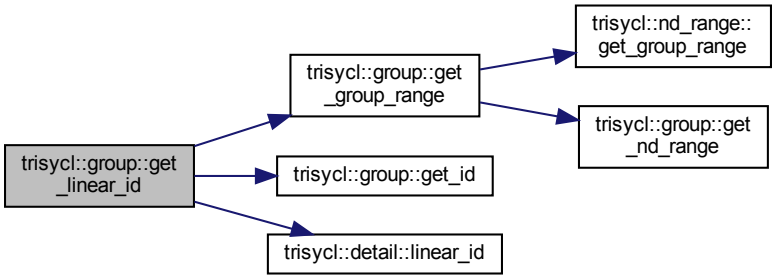
Get a linearized version of the group ID.

Definition at line 171 of file group.hpp.

```
00171                                {
00172     return detail::linear_id(get_group_range(), get_id());
00173   }
```

References trisycl::group< Dimensions >::get_group_range(), trisycl::group< Dimensions >::get_id(), and trisycl::detail::linear_id().

Here is the call graph for this function:



9.19.2.1.2.8 get_local_range() [1/2] template<int Dimensions>

```
range< Dimensions > trisycl::group< Dimensions >::get_local_range ( ) const  [inline]
```

Get the local range for this work_group.

Todo Add to the specification

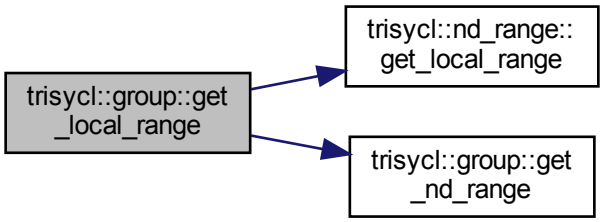
Definition at line 136 of file group.hpp.

```
00136                                                {
00137     return get_nd_range().get_local_range();
00138   }
```

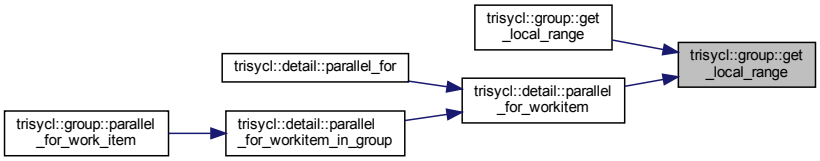
References trisycl::nd_range< Dimensions >::get_local_range(), and trisycl::group< Dimensions >::get_nd_range().

Referenced by trisycl::group< Dimensions >::get_local_range(), and trisycl::detail::parallel_for_workitem().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.2.1.2.9 get_local_range() [2/2] template<int Dimensions>

```
size_t trisycl::group< Dimensions >::get_local_range (
            int dimension ) const  [inline]
```

Return element dimension from the constituent local range.

Todo Add to the specification

Definition at line 145 of file group.hpp.

```
00145                                                {
00146     return get_local_range()[dimension];
00147   }
```

References trisycl::group< Dimensions >::get_local_range().

Here is the call graph for this function:



9.19.2.1.2.10 get_nd_range() template<int Dimensions>

```
nd_range< Dimensions > trisycl::group< Dimensions >::get_nd_range ( ) const  [inline]
```

Todo Also provide this access to the current nd_range

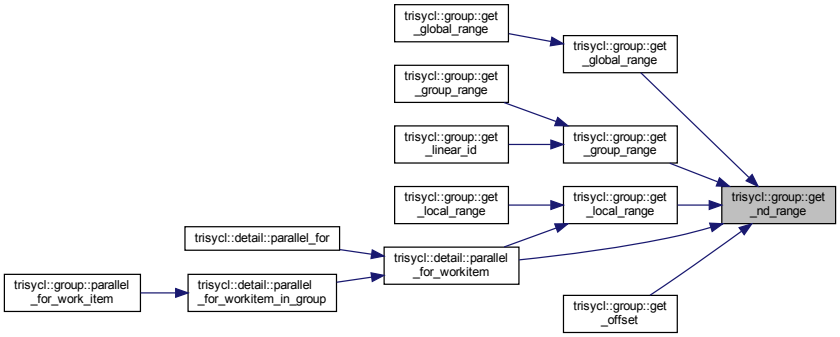
Definition at line 165 of file group.hpp.

```
00165 { return ndr; }
```

References trisycl::group< Dimensions >::ndr.

Referenced by trisycl::group< Dimensions >::get_global_range(), trisycl::group< Dimensions >::get_group_range(), trisycl::group< Dimensions >::get_local_range(), trisycl::group< Dimensions >::get_offset(), and trisycl::detail::parallel_for_workitem

Here is the caller graph for this function:



9.19.2.1.2.11 get_offset() [1/2] template<int Dimensions>

```
id< Dimensions > trisycl::group< Dimensions >::get_offset ( ) const  [inline]
```

Get the offset of the NDRange.

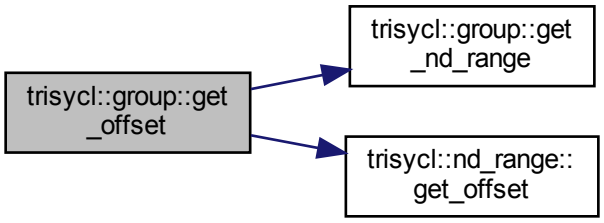
Todo Add to the specification

Definition at line 154 of file group.hpp.

```
00154 { return get_nd_range().get_offset(); }
```

References trisycl::group< Dimensions >::get_nd_range(), and trisycl::nd_range< Dimensions >::get_offset().

Here is the call graph for this function:



9.19.2.1.2.12 get_offset() [2/2] template<int Dimensions>

```
size_t trisycl::group< Dimensions >::get_offset (
            int dimension ) const  [inline]
```

Get the offset of the NDRange.

Todo Add to the specification

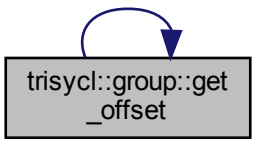
Definition at line 161 of file group.hpp.

```
00161 { return get_offset()[dimension]; }
```

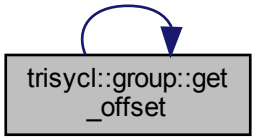
References trisycl::group< Dimensions >::get_offset().

Referenced by trisycl::group< Dimensions >::get_offset().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.2.1.2.13 operator==() template<int Dimensions>

```
bool trisycl::group< Dimensions >::operator== (
            const group< Dimensions > & groupB ) const  [inline]
```

Definition at line 185 of file group.hpp.

```
00185                                                 {
00186     return (group_id == groupB.group_id &&
00187             ndr == groupB.ndr);
00188   }
```

References trisycl::group< Dimensions >::group_id, and trisycl::group< Dimensions >::ndr.

9.19.2.1.2.14 operator[]() template<int Dimensions>

```
auto & trisycl::group< Dimensions >::operator[] (
            int dimension )  [inline]
```

Return the index of the group in the given dimension within the nd_range<>

Todo In this implementation it is not const because the group<> is written in the parallel_for iterators. To fix according to the specification

Definition at line 96 of file group.hpp.

```
00096                                  {
00097     return group_id[dimension];
00098   }
```

References trisycl::group< Dimensions >::group_id.

9.19.2.1.2.15 parallel_for_work_item() template<int Dimensions>

```
void trisycl::group< Dimensions >::parallel_for_work_item (
            std::function< void(h_item< rank()>)> f ) const  [inline]
```

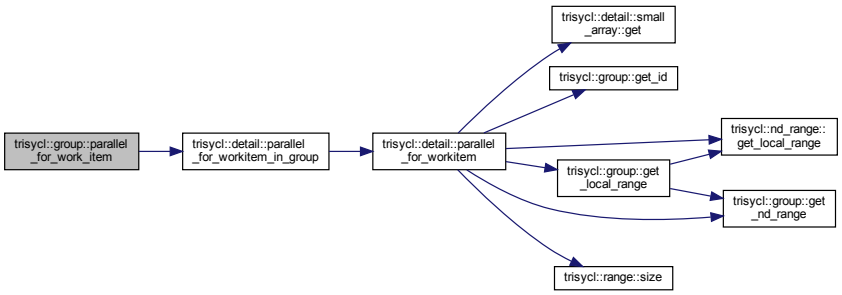
Loop on the work-items inside a work-group.

Definition at line 178 of file group.hpp.

```
00179           {
00180     detail::parallel_for_workitem_in_group(*this, f);
00181   }
```

References trisycl::detail::parallel_for_workitem_in_group().

Here is the call graph for this function:



9.19.2.1.2.16 rank() template<int Dimensions>

```
static auto constexpr trisycl::group< Dimensions >::rank ( )  [inline], [static], [constexpr]
```

The number of dimensions of the group.

Definition at line 43 of file group.hpp.

```
00043 { return Dimensions; }
```

9.19.2.1.3 Member Data Documentation

9.19.2.1.3.1 group_id template<int Dimensions>

```
id<Dimensions> trisycl::group< Dimensions >::group_id  [private]
```

The coordinate of the group item.

Definition at line 48 of file group.hpp.

Referenced by trisycl::group< Dimensions >::get_id(), trisycl::group< Dimensions >::operator==(), and trisycl::group< Dimensions >::operator[]().

9.19.2.1.3.2 ndr template<int Dimensions>

nd_range<Dimensions> trisycl::group< Dimensions >::ndr [private]

Keep a reference on the nd_range to serve potential query on it.

Definition at line 51 of file group.hpp.

Referenced by trisycl::group< Dimensions >::get_nd_range(), and trisycl::group< Dimensions >::operator==().

9.19.2.2 struct trisycl::h_item

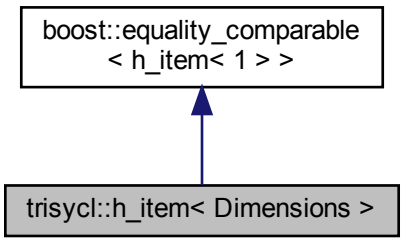
template<int Dimensions = 1>
struct trisycl::h_item< Dimensions >

`h_item<int dimensions>` identifies an instance of a `group::parallel_for_work_item` function object executing at each point in a local `range<int dimensions>` passed to a `parallel_for_work_item` call or to the corresponding `parallel_for_work_group` call if no range is passed to the `parallel_for_work_item` call.

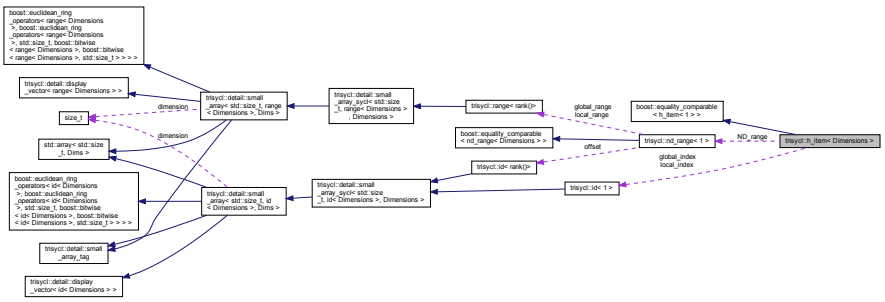
It encapsulates enough information to identify the work-item's local and global items according to the information given to `parallel_for_work_group` (physical ids) as well as the work-item's logical local items in the flexible range. All returned items objects are offset-less. Instances of the `h_item<int dimensions>` class are not user-constructible and are passed by the runtime to each instance of the function object.

Definition at line 42 of file h_item.hpp.

Inheritance diagram for trisycl::h_item< Dimensions >:



Collaboration diagram for trisycl::h_item< Dimensions >:



Public Member Functions

- h_item (nd_range< Dimensions > ndr)

  Create an empty nd_item<> from an nd_range<>
- h_item (id< Dimensions > global_index, nd_range< Dimensions > ndr)

  Create a full nd_item.
- h_item ()=default

  To be able to copy and assign nd_item, use default constructors too.
- id< Dimensions > get_global_id () const

  Return the constituent global id representing the work-item's position in the global iteration space.
- size_t get_global_id (int dimension) const

  Return the constituent element of the global id representing the work-item's position in the global iteration space in the given dimension.
- size_t get_global_linear_id () const

  Return the flattened id of the current work-item after subtracting the offset.
- id< Dimensions > get_local_id () const

  Return the constituent local id representing the work-item's position within the current work-group.
- size_t get_local_id (int dimension) const

  Return the constituent element of the local id representing the work-item's position within the current work-group in the given dimension.
- size_t get_local_linear_id () const

  Return the flattened id of the current work-item within the current work-group.
- id< Dimensions > get_group () const

  Return the constituent group group representing the work-group's position within the overall nd_range.
- size_t get_group (int dimension) const

  Return the constituent element of the group id representing the work-group's position within the overall nd_range in the given dimension.
- size_t get_group_linear_id () const

  Return the flattened id of the current work-group.
- id< Dimensions > get_group_range () const

  Return the number of groups in the nd_range.
- size_t get_group_range (int dimension) const

  Return the number of groups for dimension in the nd_range.
- range< Dimensions > get_global_range () const

  Return a range<> representing the dimensions of the nd_range<>
- range< Dimensions > get_local_range () const

  Return a range<> representing the dimensions of the current work-group.
- id< Dimensions > get_offset () const

  Return an id<> representing the n-dimensional offset provided to the constructor of the nd_range<> and that is added by the runtime to the global-ID of each work-item.
- nd_range< Dimensions > get_nd_range () const

  Return the nd_range<> of the current execution.
- item< Dimensions > get_item () const

  Allows projection down to an item.
- void barrier (access::fence_space flag=access::fence_space::global_and_local) const

  Execute a barrier with memory ordering on the local address space, global address space or both based on the value of flag.
- void set_local (id< Dimensions > Index)
- void set_global (id< Dimensions > Index)
- bool operator== (const h_item< Dimensions > &itemB) const

  comparison operators

Static Public Member Functions

- static auto constexpr rank ()

  The number of dimensions of the h_item.

Private Attributes

- id< Dimensions > global_index
- id< Dimensions > local_index
- nd_range< Dimensions > ND_range

9.19.2.2.1 Constructor & Destructor Documentation

9.19.2.2.1.1 h_item() [1/3] template<int Dimensions = 1>

```
trisycl::h_item< Dimensions >::h_item (
            nd_range< Dimensions > ndr )  [inline]
```

Create an empty nd_item<> from an nd_range<>

Todo This is for the triSYCL implementation which is expected to call set_global() and set_local() later. This should be hidden to the user.

Definition at line 62 of file h_item.hpp.

```
00062 :  ND_range { ndr } {}
```

9.19.2.2.1.2 h_item() [2/3] template<int Dimensions = 1>

```
trisycl::h_item< Dimensions >::h_item (
            id< Dimensions > global_index,
            nd_range< Dimensions > ndr )  [inline]
```

Create a full nd_item.

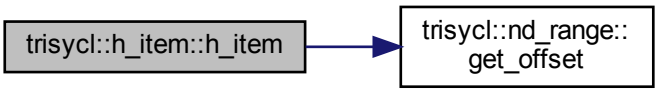
Todo This is for validation purpose. Hide this to the programmer somehow

Definition at line 70 of file h_item.hpp.

```
00071                                    :
00072     global_index { global_index },
00073     // Compute the local index using the offset and the group size
00074     local_index
00075       { (global_index - ndr.get_offset())%id<Dimensions> { ndr.get_local() } },
00076     ND_range { ndr }
00077   {}
```

References trisycl::nd_range< Dimensions >::get_offset(), and trisycl::h_item< Dimensions >::global_index.

Here is the call graph for this function:



9.19.2.2.1.3 h_item() [3/3] template<int Dimensions = 1>

trisycl::h_item< Dimensions >::h_item ( ) [default]

To be able to copy and assign nd_item, use default constructors too.

Todo Make most of them protected, reserved to implementation

9.19.2.2.2 Member Function Documentation

9.19.2.2.2.1 barrier() template<int Dimensions = 1>

void trisycl::h_item< Dimensions >::barrier (
access::fence_space *flag* = *access::fence_space::global_and_local*) const [inline]

Execute a barrier with memory ordering on the local address space, global address space or both based on the value of flag.

The current work-item will wait at the barrier until all work-items in the current work-group have reached the barrier.

In addition, the barrier performs a fence operation ensuring that all memory accesses in the specified address space issued before the barrier complete before those issued after the barrier

Definition at line 209 of file h_item.hpp.

```
00210                                                    {
00211 #if defined(_OPENMP) && !defined(TRISYCL_NO_BARRIER)
00212     /* Use OpenMP barrier in the implementation with 1 OpenMP thread per
00213 work-item of the work-group */
00214 #pragma omp barrier
00215 #else
00216     // \todo To be implemented efficiently otherwise
00217     TRISYCL_UNIMPL;
00218 #endif
00219   }
```

References TRISYCL_UNIMPL.

9.19.2.2.2.2 get_global_id() [1/2] template<int Dimensions = 1>

id< Dimensions > trisycl::h_item< Dimensions >::get_global_id ( ) const [inline]

Return the constituent global id representing the work-item's position in the global iteration space.

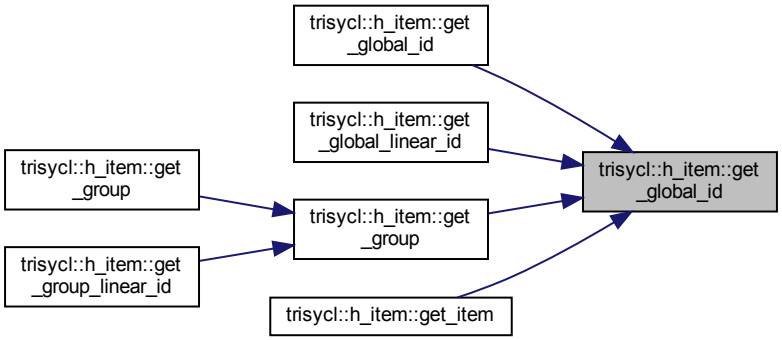
Definition at line 90 of file h_item.hpp.

```
00090 { return global_index; }
```

References trisycl::h_item< Dimensions >::global_index.

Referenced by trisycl::h_item< Dimensions >::get_global_id(), trisycl::h_item< Dimensions >::get_global_linear_id(), trisycl::h_item< Dimensions >::get_group(), and trisycl::h_item< Dimensions >::get_item().

Here is the caller graph for this function:



9.19.2.2.2.3 get_global_id() [2/2] template<int Dimensions = 1>

```
size_t trisycl::h_item< Dimensions >::get_global_id (
            int dimension ) const  [inline]
```

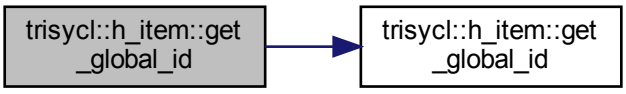
Return the constituent element of the global id representing the work-item's position in the global iteration space in the given dimension.

Definition at line 97 of file h_item.hpp.

```
00097                                          {
00098     return get_global_id()[dimension];
00099   }
```

References trisycl::h_item< Dimensions >::get_global_id().

Here is the call graph for this function:



9.19.2.2.4 get_global_linear_id() template<int Dimensions = 1>

```
size_t trisycl::h_item< Dimensions >::get_global_linear_id ( ) const  [inline]
```

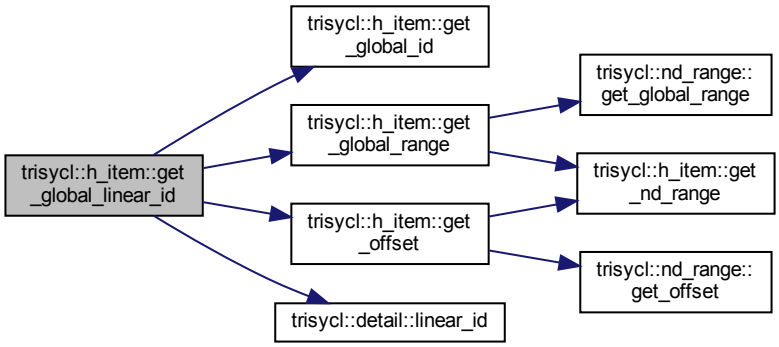
Return the flattened id of the current work-item after subtracting the offset.

Definition at line 105 of file h_item.hpp.

```
00105                                       {
00106     return detail::linear_id(get_global_range(), get_global_id(), get_offset());
00107   }
```

References trisycl::h_item< Dimensions >::get_global_id(), trisycl::h_item< Dimensions >::get_global_range(), trisycl::h_item< Dimensions >::get_offset(), and trisycl::detail::linear_id().

Here is the call graph for this function:



9.19.2.2.5 get_global_range() template<int Dimensions = 1>

```
range< Dimensions > trisycl::h_item< Dimensions >::get_global_range ( ) const  [inline]
```

Return a range<> representing the dimensions of the nd_range<>

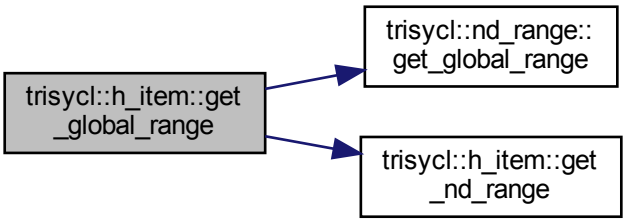
Definition at line 168 of file h_item.hpp.

```
00168                                       {
00169     return get_nd_range().get_global_range();
00170   }
```

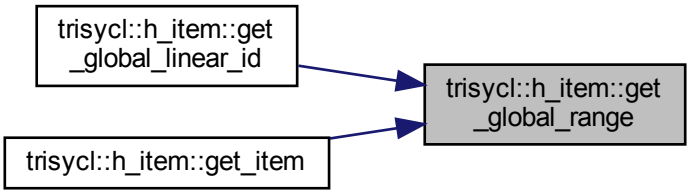
References trisycl::nd_range< Dimensions >::get_global_range(), and trisycl::h_item< Dimensions >::get_nd_range().

Referenced by trisycl::h_item< Dimensions >::get_global_linear_id(), and trisycl::h_item< Dimensions >::get_item().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.2.2.2.6 get_group() [1/2] template<int Dimensions = 1>

```
id< Dimensions > trisycl::h_item< Dimensions >::get_group ( ) const  [inline]
```

Return the constituent group group representing the work-group's position within the overall nd_range.

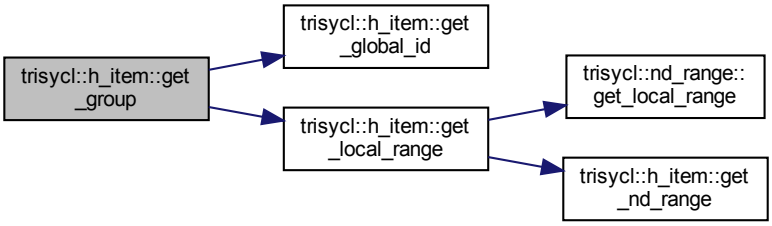
Definition at line 134 of file h_item.hpp.

```
00134                                  {
00135     /* Convert get_local_range() to an id<> to remove ambiguity into using
00136 implicit conversion either from range<> to id<> or the opposite */
00137     return get_global_id()/id<Dimensions> { get_local_range() };
00138   }
```

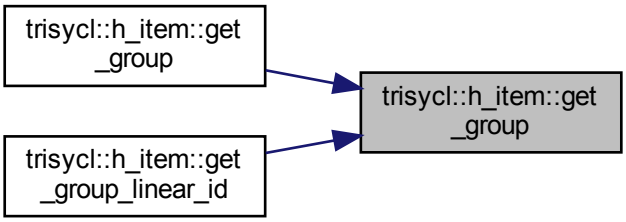
References trisycl::h_item< Dimensions >::get_global_id(), and trisycl::h_item< Dimensions >::get_local_range().

Referenced by trisycl::h_item< Dimensions >::get_group(), and trisycl::h_item< Dimensions >::get_group_linear_id().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.2.2.7 get_group() [2/2] template<int Dimensions = 1>

```
size_t trisycl::h_item< Dimensions >::get_group (
            int dimension ) const  [inline]
```

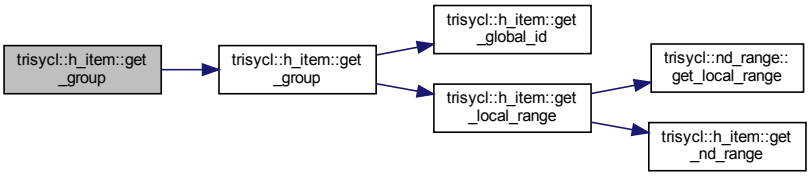
Return the constituent element of the group id representing the work-group's position within the overall nd_range in the given dimension.

Definition at line 145 of file h_item.hpp.

```
00145                                       {
00146     return get_group()[dimension];
00147   }
```

References trisycl::h_item< Dimensions >::get_group().

Here is the call graph for this function:



9.19.2.2.2.8 get_group_linear_id() template<int Dimensions = 1>

```
size_t trisycl::h_item< Dimensions >::get_group_linear_id ( ) const  [inline]
```

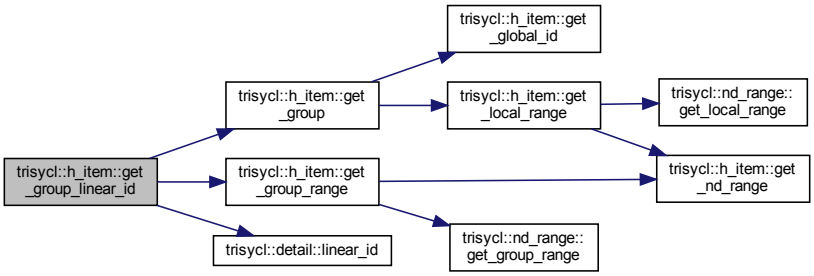
Return the flattened id of the current work-group.

Definition at line 151 of file h_item.hpp.

```
00151                                       {
00152     return detail::linear_id(get_group_range(), get_group());
00153   }
```

References trisycl::h_item< Dimensions >::get_group(), trisycl::h_item< Dimensions >::get_group_range(), and trisycl::detail::linear_id().

Here is the call graph for this function:



9.19.2.2.2.9 get_group_range() [1/2] template<int Dimensions = 1>

```
id< Dimensions > trisycl::h_item< Dimensions >::get_group_range ( ) const  [inline]
```

Return the number of groups in the nd_range.

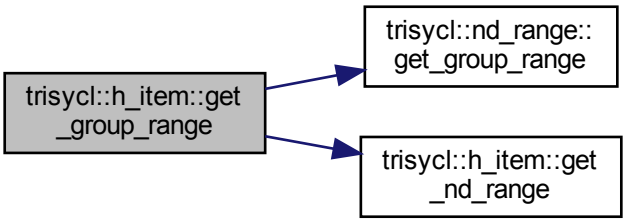
Definition at line 157 of file h_item.hpp.

```
00157                                       {
00158     return get_nd_range().get_group_range();
00159   }
```

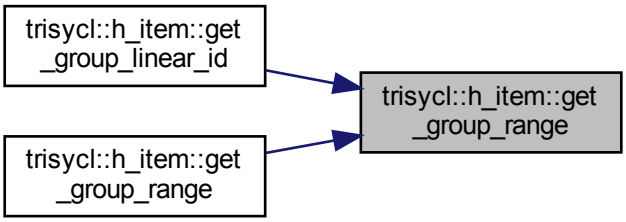
References trisycl::nd_range< Dimensions >::get_group_range(), and trisycl::h_item< Dimensions >::get_nd_range().

Referenced by trisycl::h_item< Dimensions >::get_group_linear_id(), and trisycl::h_item< Dimensions >::get_group_range().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.2.2.10 get_group_range() [2/2] template<int Dimensions = 1>

```
size_t trisycl::h_item< Dimensions >::get_group_range (
            int dimension ) const  [inline]
```

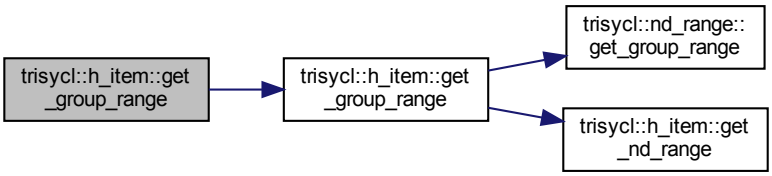
Return the number of groups for dimension in the nd_range.

Definition at line 162 of file h_item.hpp.

```
00162                                          {
00163       return get_group_range()[dimension];
00164   }
```

References trisycl::h_item< Dimensions >::get_group_range().

Here is the call graph for this function:



9.19.2.2.2.11 get_item() template<int Dimensions = 1>

item< Dimensions > trisycl::h_item< Dimensions >::get_item ( ) const [inline]

Allows projection down to an item.

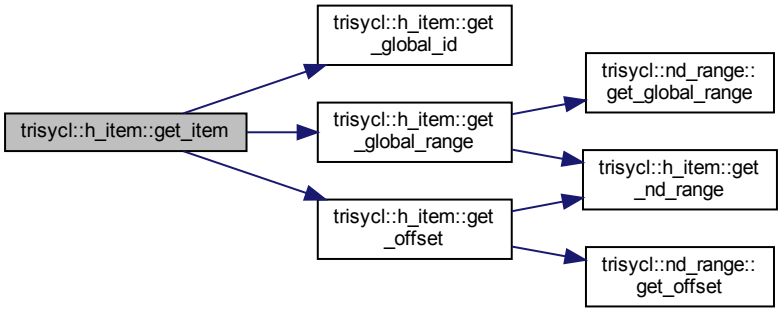
Todo Add to the specification

Definition at line 194 of file h_item.hpp.

```
00194                                  {
00195     return { get_global_range(), get_global_id(), get_offset() };
00196   }
```

References trisycl::h_item< Dimensions >::get_global_id(), trisycl::h_item< Dimensions >::get_global_range(), and trisycl::h_item< Dimensions >::get_offset().

Here is the call graph for this function:



9.19.2.2.2.12 get_local_id() [1/2] template<int Dimensions = 1>

id< Dimensions > trisycl::h_item< Dimensions >::get_local_id ( ) const [inline]

Return the constituent local id representing the work-item's position within the current work-group.

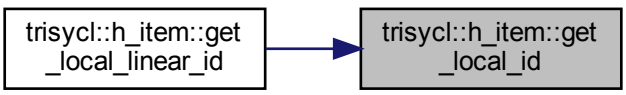
Definition at line 113 of file h_item.hpp.

```
00113 { return local_index; }
```

References trisycl::h_item< Dimensions >::local_index.

Referenced by trisycl::h_item< Dimensions >::get_local_linear_id().

Here is the caller graph for this function:



9.19.2.2.13 get_local_id() [2/2] template<int Dimensions = 1>

```
size_t trisycl::h_item< Dimensions >::get_local_id (
            int dimension ) const  [inline]
```

Return the constituent element of the local id representing the work-item's position within the current work-group in the given dimension.

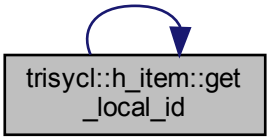
Definition at line 120 of file h_item.hpp.

```
00120 { return get_local_id()[dimension]; }
```

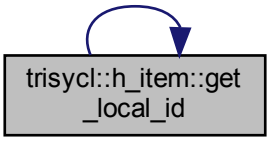
References trisycl::h_item< Dimensions >::get_local_id().

Referenced by trisycl::h_item< Dimensions >::get_local_id().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.2.2.14 get_local_linear_id() template<int Dimensions = 1>

```
size_t trisycl::h_item< Dimensions >::get_local_linear_id ( ) const  [inline]
```

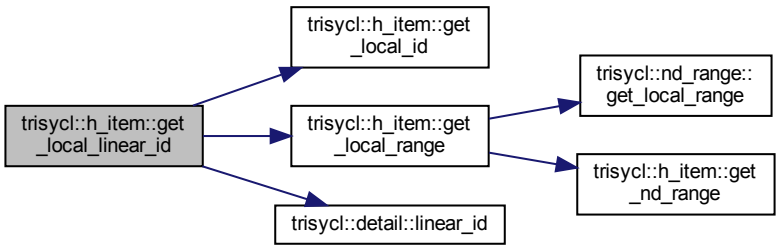
Return the flattened id of the current work-item within the current work-group.

Definition at line 126 of file h_item.hpp.

```
00126                                       {
00127     return detail::linear_id(get_local_range(), get_local_id());
00128   }
```

References trisycl::h_item< Dimensions >::get_local_id(), trisycl::h_item< Dimensions >::get_local_range(), and trisycl::detail::linear_id().

Here is the call graph for this function:



9.19.2.2.2.15 get_local_range() template<int Dimensions = 1>

range< Dimensions > trisycl::h_item< Dimensions >::get_local_range ( ) const [inline]

Return a range<> representing the dimensions of the current work-group.

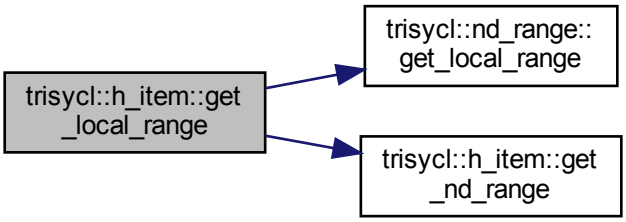
Definition at line 174 of file h_item.hpp.

```
00174                                       {
00175     return get_nd_range().get_local_range();
00176   }
```

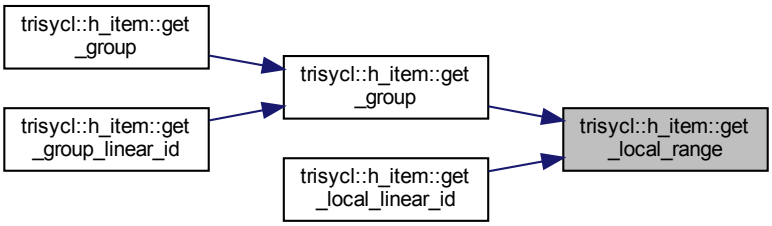
References trisycl::nd_range< Dimensions >::get_local_range(), and trisycl::h_item< Dimensions >::get_nd_range().

Referenced by trisycl::h_item< Dimensions >::get_group(), and trisycl::h_item< Dimensions >::get_local_linear_id().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.2.2.2.16 get_nd_range() template<int Dimensions = 1>

```
nd_range< Dimensions > trisycl::h_item< Dimensions >::get_nd_range ( ) const  [inline]
```

Return the nd_range<> of the current execution.

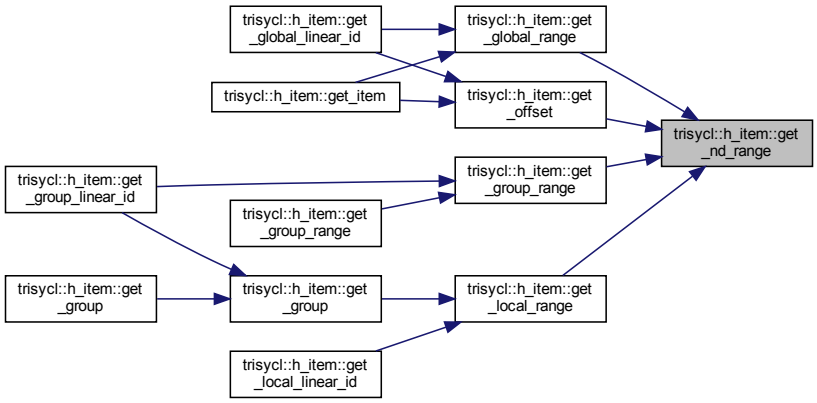
Definition at line 187 of file h_item.hpp.

```
00187 { return ND_range; }
```

References trisycl::h_item< Dimensions >::ND_range.

Referenced by trisycl::h_item< Dimensions >::get_global_range(), trisycl::h_item< Dimensions >::get_group_range(), trisycl::h_item< Dimensions >::get_local_range(), and trisycl::h_item< Dimensions >::get_offset().

Here is the caller graph for this function:



9.19.2.2.17 get_offset() template<int Dimensions = 1>

```
id< Dimensions > trisycl::h_item< Dimensions >::get_offset ( ) const  [inline]
```

Return an id<> representing the n-dimensional offset provided to the constructor of the nd_range<> and that is added by the runtime to the global-ID of each work-item.

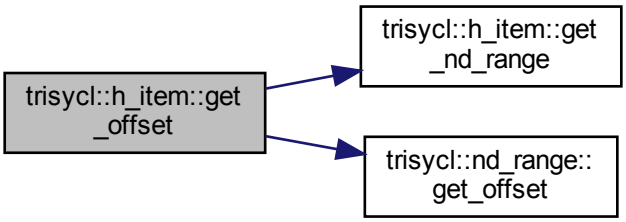
Definition at line 183 of file h_item.hpp.

```
00183 { return get_nd_range().get_offset(); }
```

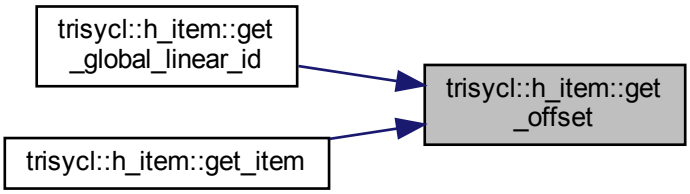
References trisycl::h_item< Dimensions >::get_nd_range(), and trisycl::nd_range< Dimensions >::get_offset().

Referenced by trisycl::h_item< Dimensions >::get_global_linear_id(), and trisycl::h_item< Dimensions >::get_item().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.2.2.18 operator==() template<int Dimensions = 1>

```
bool trisycl::h_item< Dimensions >::operator== (
              const h_item< Dimensions > & itemB ) const  [inline]
```

comparison operators

Definition at line 230 of file h_item.hpp.

```
00230                                                  {
00231     return (ND_range == itemB.ND_range &&
00232             global_index == itemB.global_index);
00233   }
```

References trisycl::h_item< Dimensions >::global_index, and trisycl::h_item< Dimensions >::ND_range.

9.19.2.2.19 rank() template<int Dimensions = 1>

```
static auto constexpr trisycl::h_item< Dimensions >::rank ( )  [inline], [static], [constexpr]
```

The number of dimensions of the h_item.

Definition at line 44 of file h_item.hpp.

```
00044 { return Dimensions; }
```

9.19.2.2.20 set_global() template<int Dimensions = 1>

```
void trisycl::h_item< Dimensions >::set_global (
              id< Dimensions > Index )  [inline]
```

Definition at line 227 of file h_item.hpp.

```
00227 { global_index = Index; }
```

References trisycl::h_item< Dimensions >::global_index.

9.19.2.2.21 set_local() template<int Dimensions = 1>

```
void trisycl::h_item< Dimensions >::set_local (
              id< Dimensions > Index )  [inline]
```

Definition at line 223 of file h_item.hpp.

```
00223 { local_index = Index; }
```

References trisycl::h_item< Dimensions >::local_index.

9.19.2.2.3 Member Data Documentation

9.19.2.2.3.1 global_index template<int Dimensions = 1>

```
id<Dimensions> trisycl::h_item< Dimensions >::global_index  [private]
```

Definition at line 48 of file h_item.hpp.

Referenced by trisycl::h_item< Dimensions >::get_global_id(), trisycl::h_item< Dimensions >::h_item(), trisycl::h_item< Dimensions
and trisycl::h_item< Dimensions >::set_global().

9.19.2.2.3.2 local_index template<int Dimensions = 1>

```
id<Dimensions> trisycl::h_item< Dimensions >::local_index  [private]
```

Definition at line 51 of file h_item.hpp.

Referenced by trisycl::h_item< Dimensions >::get_local_id(), and trisycl::h_item< Dimensions >::set_local().

9.19.2.2.3.3 ND_range `template<int Dimensions = 1>`

`nd_range<Dimensions> trisycl::h_item< Dimensions >::ND_range [private]`

Definition at line 52 of file h_item.hpp.

Referenced by trisycl::h_item< Dimensions >::get_nd_range(), and trisycl::h_item< Dimensions >::operator==().

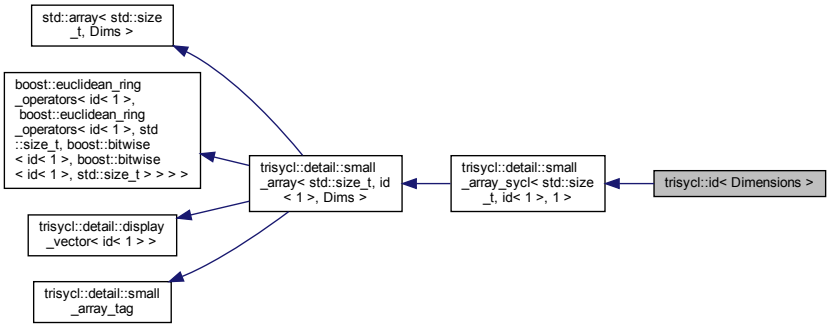
9.19.2.3 class trisycl::id

template<int Dimensions = 1>
class trisycl::id< Dimensions >

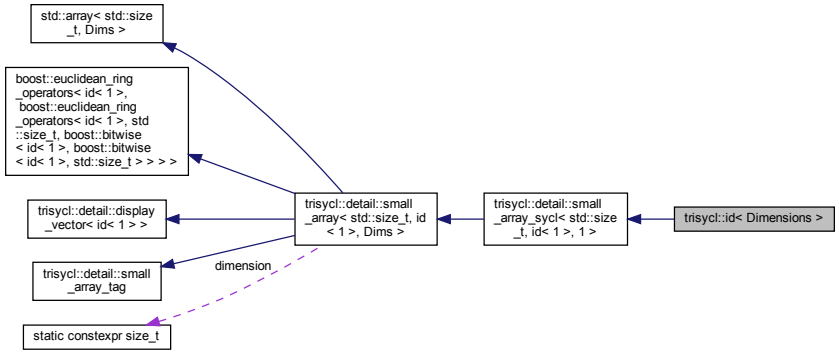
Define a multi-dimensional index, used for example to locate a work item.

Definition at line 32 of file id.hpp.

Inheritance diagram for trisycl::id< Dimensions >:



Collaboration diagram for trisycl::id< Dimensions >:



Public Member Functions

- id (const range< Dimensions > &range_size)

  Construct an id from a range.
- id (const item< Dimensions, true > &rhs)

  Construct an id from an item global_id.
- id ()

  Default constructor must 0 all elements.

Static Public Member Functions

- static auto constexpr rank ()

  The number of dimensions of the id.

Additional Inherited Members

9.19.2.3.1 Constructor & Destructor Documentation

9.19.2.3.1.1 id() [1/3] template<int Dimensions = 1>

```
trisycl::id< Dimensions >::id (
            const range< Dimensions > & range_size )  [inline]
```

Construct an id from a range.

Use the fact we have a constructor of a small_array from a another kind of small_array

Definition at line 47 of file id.hpp.

```
00051       :  detail::small_array_sycl<std::size_t, id<Dimensions>, Dimensions> {
00052         range_size
00053       } {}
```

9.19.2.3.1.2 id() [2/3] template<int Dimensions = 1>

```
trisycl::id< Dimensions >::id (
            const item< Dimensions, true > & rhs )  [inline]
```

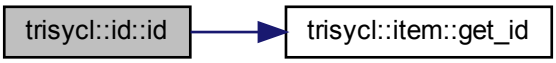
Construct an id from an item global_id.

Definition at line 56 of file id.hpp.

```
00057     :  detail::small_array_sycl<std::size_t, id<Dimensions>, Dimensions>
00058       { rhs.get_id() }
00059   {}
```

References trisycl::item< Dimensions, with_offset >::get_id().

Here is the call graph for this function:



9.19.2.3.1.3 id() [3/3] template<int Dimensions = 1>

trisycl::id< Dimensions >::id ( ) [inline]

Default constructor must 0 all elements.

Definition at line 62 of file id.hpp.

```
00062         :  detail::small_array_sycl<std::size_t, id<Dimensions>, Dimensions> { 0 }
00063   {}
```

9.19.2.3.2 Member Function Documentation

9.19.2.3.2.1 rank() template<int Dimensions = 1>

static auto constexpr trisycl::id< Dimensions >::rank ( ) [inline], [static], [constexpr]

The number of dimensions of the id.

Definition at line 39 of file id.hpp.

```
00039 { return Dimensions; }
```

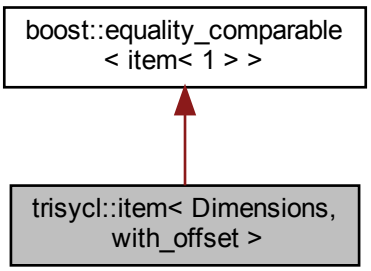
9.19.2.4 class trisycl::item

template<int Dimensions = 1, bool with_offset = true>
class trisycl::item< Dimensions, with_offset >

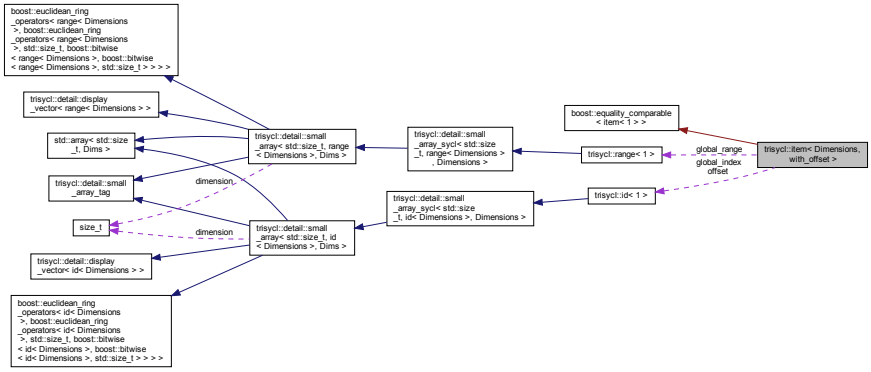
A SYCL item stores information on a work-item with some more context such as the definition range and offset.

Definition at line 28 of file item.hpp.

Inheritance diagram for trisycl::item< Dimensions, with_offset >:



Collaboration diagram for trisycl::item< Dimensions, with_offset >:



Public Member Functions

- item (range< Dimensions > global_size, id< Dimensions > global_index, id< Dimensions > offset={})

  Create an item from a local size and an optional offset.
- item ()=default

  To be able to copy and assign item, use default constructors too.
- id< Dimensions > get_id () const

  Return the constituent local or global id<> representing the work-item's position in the iteration space.
- size_t get_id (int dimension) const

  Return the requested dimension of the constituent id<> representing the work-item's position in the iteration space.
- auto & operator[ ] (int dimension)

  Return the constituent id<> l-value representing the work-item's position in the iteration space in the given dimension.
- range< Dimensions > get_range () const

  Returns a range<> representing the dimensions of the range of possible values of the item.
- size_t get_range (int dimension) const

  Return the same value as get_range().get(dimension)
- id< Dimensions > get_offset () const

  Returns an id<> representing the n-dimensional offset provided to the parallel_for and that is added by the runtime to the global-ID of each work-item, if this item represents a global range.
- size_t get_linear_id () const

  Return the linearized ID in the item's range.
- void set (id< Dimensions > Index)

  For the implementation, need to set the global index.
- operator item< Dimensions, true > () const

  Returns an item with same dimensions but offset set to 0.
- void display () const

  Display the value for debugging and validation purpose.
- bool operator== (const item< Dimensions > &itemB) const

  Comparison operators for item.

Static Public Member Functions

- static auto constexpr rank ()

  The number of dimensions of the item.

Private Attributes

- range< Dimensions > global_range
- id< Dimensions > global_index
- id< Dimensions > offset

9.19.2.4.1 Constructor & Destructor Documentation

9.19.2.4.1.1 item() [1/2] template<int Dimensions = 1, bool with_offset = true>

```
trisycl::item< Dimensions, with_offset >::item (
            range< Dimensions > global_size,
            id< Dimensions > global_index,
            id< Dimensions > offset = {} )  [inline]
```

Create an item from a local size and an optional offset.

This constructor is used by the triSYCL implementation and the non-regression testing.

Definition at line 47 of file item.hpp.

```
00049                                {}) :
00050     global_range { global_size },
00051     global_index { global_index },
00052     offset { offset }
00053   {}
```

9.19.2.4.1.2 item() [2/2] template<int Dimensions = 1, bool with_offset = true>

```
trisycl::item< Dimensions, with_offset >::item ( )  [default]
```

To be able to copy and assign item, use default constructors too.

Todo Make most of them protected, reserved to implementation

9.19.2.4.2 Member Function Documentation

9.19.2.4.2.1 display() template<int Dimensions = 1, bool with_offset = true>

```
void trisycl::item< Dimensions, with_offset >::display ( ) const  [inline]
```

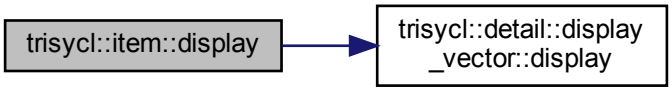
Display the value for debugging and validation purpose.

Definition at line 124 of file item.hpp.

```
00124                        {
00125     global_range.display();
00126     global_index.display();
00127     offset.display();
00128   }
```

References trisycl::detail::display_vector< T >::display(), trisycl::item< Dimensions, with_offset >::global_index, and trisycl::item< Dimensions, with_offset >::offset.

Here is the call graph for this function:



9.19.2.4.2.2 get_id() [1/2] template<int Dimensions = 1, bool with_offset = true>

```
id< Dimensions > trisycl::item< Dimensions, with_offset >::get_id ( ) const  [inline]
```

Return the constituent local or global id<> representing the work-item's position in the iteration space.

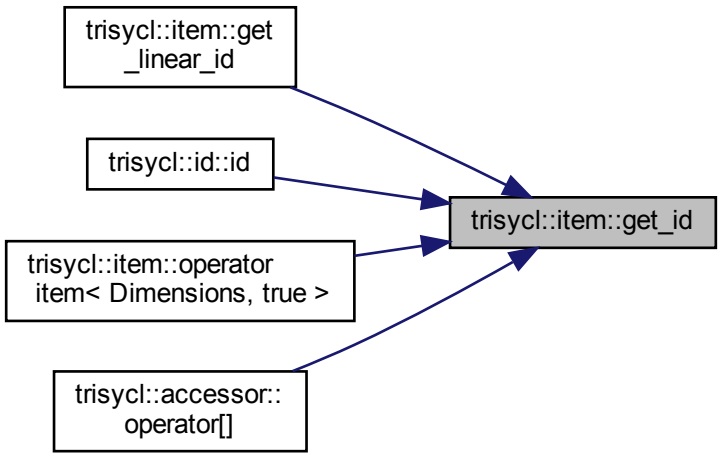
Definition at line 66 of file item.hpp.

```
00066 { return global_index; }
```

References trisycl::item< Dimensions, with_offset >::global_index.

Referenced by trisycl::item< Dimensions, with_offset >::get_linear_id(), trisycl::id< Dimensions >::id(), trisycl::item< Dimensions, w and trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator[]().

Here is the caller graph for this function:



9.19.2.4.2.3 get_id() [2/2] template<int Dimensions = 1, bool with_offset = true>

```
size_t trisycl::item< Dimensions, with_offset >::get_id (
            int dimension ) const  [inline]
```

Return the requested dimension of the constituent id<> representing the work-item's position in the iteration space.

Definition at line 72 of file item.hpp.

```
00072 { return get_id()[dimension]; }
```

References trisycl::item< Dimensions, with_offset >::get_id().

Referenced by trisycl::item< Dimensions, with_offset >::get_id().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.2.4.2.4 get_linear_id()

```
template<int Dimensions = 1, bool with_offset = true>
size_t trisycl::item< Dimensions, with_offset >::get_linear_id ( ) const  [inline]
```

Return the linearized ID in the item's range.

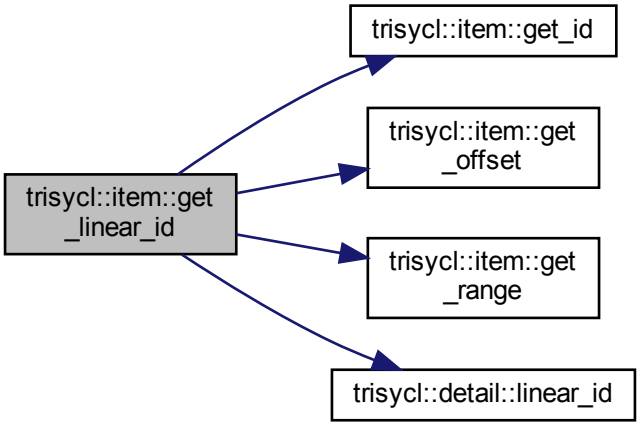
Computed as the flatted ID after the offset is subtracted.

Definition at line 106 of file item.hpp.

```
00106                                {
00107     return detail::linear_id(get_range(), get_id(), get_offset());
00108   }
```

References trisycl::item< Dimensions, with_offset >::get_id(), trisycl::item< Dimensions, with_offset >::get_offset(), trisycl::item< Dimensions, with_offset >::get_range(), and trisycl::detail::linear_id().

Here is the call graph for this function:



9.19.2.4.2.5 get_offset()

```
template<int Dimensions = 1, bool with_offset = true>
id< Dimensions > trisycl::item< Dimensions, with_offset >::get_offset ( ) const  [inline]
```

Returns an id<> representing the n-dimensional offset provided to the parallel_for and that is added by the runtime to the global-ID of each work-item, if this item represents a global range.

For an item representing a local range of where no offset was passed this will always return an id of all 0 values.

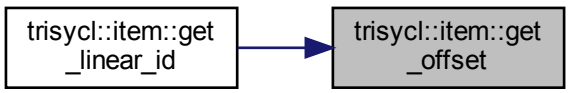
Definition at line 96 of file item.hpp.

```
00096                                  {
00097     static_assert(with_offset == true, "get_offset only callable with has_offset = true");
00098     return offset;
00099   }
```

References trisycl::item< Dimensions, with_offset >::offset.

Referenced by trisycl::item< Dimensions, with_offset >::get_linear_id().

Here is the caller graph for this function:



9.19.2.4.2.6 get_range() [1/2] template<int Dimensions = 1, bool with_offset = true>

```
range< Dimensions > trisycl::item< Dimensions, with_offset >::get_range ( ) const  [inline]
```

Returns a range<> representing the dimensions of the range of possible values of the item.

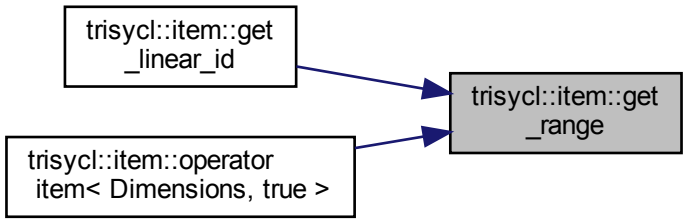
Definition at line 84 of file item.hpp.

```
00084 { return global_range; }
```

References trisycl::item< Dimensions, with_offset >::global_range.

Referenced by trisycl::item< Dimensions, with_offset >::get_linear_id(), and trisycl::item< Dimensions, with_offset >::operator item<

Here is the caller graph for this function:



9.19.2.4.2.7 get_range() [2/2] template<int Dimensions = 1, bool with_offset = true>

```
size_t trisycl::item< Dimensions, with_offset >::get_range (
            int dimension ) const  [inline]
```

Return the same value as get_range().get(dimension)

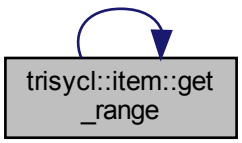
Definition at line 87 of file item.hpp.

```
00087 { return get_range().get(dimension); }
```

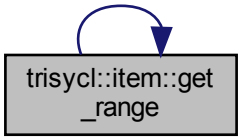
References trisycl::item< Dimensions, with_offset >::get_range().

Referenced by trisycl::item< Dimensions, with_offset >::get_range().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.2.4.2.8 operator item< Dimensions, true >() template<int Dimensions = 1, bool with_offset = true>

trisycl::item< Dimensions, with_offset >::operator item< Dimensions, true > ( ) const [inline]

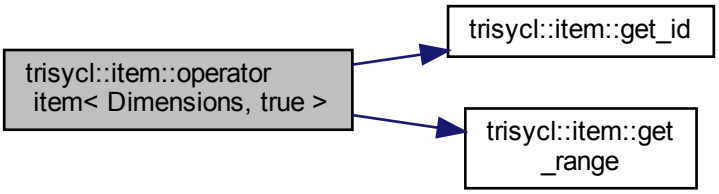
Returns an item with same dimensions but offset set to 0.

Definition at line 118 of file item.hpp.

```
00118                                       {
00119     static_assert(with_offset == false, "get_offset only callable with has_offset = true");
00120     return { this->get_range(), this->get_id(), 0 };
00121   }
```

References trisycl::item< Dimensions, with_offset >::get_id(), and trisycl::item< Dimensions, with_offset >::get_range().

Here is the call graph for this function:



9.19.2.4.2.9 operator==() template<int Dimensions = 1, bool with_offset = true>

```
bool trisycl::item< Dimensions, with_offset >::operator== (
            const item< Dimensions > & itemB ) const  [inline]
```

Comparison operators for item.

Definition at line 131 of file item.hpp.

```
00131                                                    {
00132     return (global_range == itemB.global_range &&
00133             global_index == itemB.global_index &&
00134             offset == itemB.offset);
00135   }
```

References trisycl::item< Dimensions, with_offset >::global_index, trisycl::item< Dimensions, with_offset >::global_range, and trisycl::item< Dimensions, with_offset >::offset.

9.19.2.4.2.10 operator[]() template<int Dimensions = 1, bool with_offset = true>

```
auto & trisycl::item< Dimensions, with_offset >::operator[] (
            int dimension )  [inline]
```

Return the constituent id<> l-value representing the work-item's position in the iteration space in the given dimension.

Definition at line 78 of file item.hpp.

```
00078 { return global_index[dimension]; }
```

References trisycl::item< Dimensions, with_offset >::global_index.

9.19.2.4.2.11 rank() template<int Dimensions = 1, bool with_offset = true>

```
static auto constexpr trisycl::item< Dimensions, with_offset >::rank ( )  [inline], [static],
[constexpr]
```

The number of dimensions of the item.

Definition at line 32 of file item.hpp.

```
00032 { return Dimensions; }
```

9.19.2.4.2.12 set() template<int Dimensions = 1, bool with_offset = true>

```
void trisycl::item< Dimensions, with_offset >::set (
            id< Dimensions > Index )  [inline]
```

For the implementation, need to set the global index.

Todo Move to private and add friends

Definition at line 115 of file item.hpp.

```
00115 { global_index = Index; }
```

References trisycl::item< Dimensions, with_offset >::global_index.

9.19.2.4.3 Member Data Documentation

9.19.2.4.3.1 global_index `template<int Dimensions = 1, bool with_offset = true>`

`id<Dimensions> trisycl::item< Dimensions, with_offset >::global_index [private]`

Definition at line 37 of file item.hpp.

Referenced by trisycl::item< Dimensions, with_offset >::display(), trisycl::item< Dimensions, with_offset >::get_id(), trisycl::item< Dimensions, with_offset >::operator==(), trisycl::item< Dimensions, with_offset >::operator[ ](), and trisycl::item< Dimensions, with_offset >::set().

9.19.2.4.3.2 global_range `template<int Dimensions = 1, bool with_offset = true>`

`range<Dimensions> trisycl::item< Dimensions, with_offset >::global_range [private]`

Definition at line 36 of file item.hpp.

Referenced by trisycl::item< Dimensions, with_offset >::get_range(), and trisycl::item< Dimensions, with_offset >::operator==().

9.19.2.4.3.3 offset `template<int Dimensions = 1, bool with_offset = true>`

`id<Dimensions> trisycl::item< Dimensions, with_offset >::offset [private]`

Definition at line 38 of file item.hpp.

Referenced by trisycl::item< Dimensions, with_offset >::display(), trisycl::item< Dimensions, with_offset >::get_offset(), and trisycl::item< Dimensions, with_offset >::operator==().

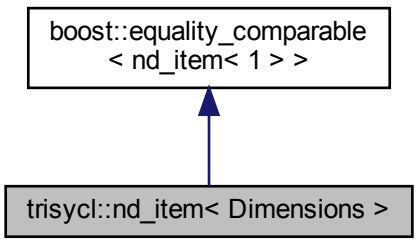
9.19.2.5 struct trisycl::nd_item

template< int Dimensions = 1 >
struct trisycl::nd_item< Dimensions >

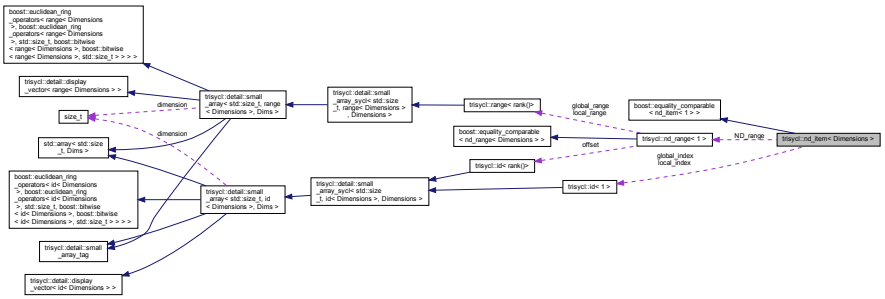
A SYCL nd_item stores information on a work-item within a work-group, with some more context such as the definition ranges.

Definition at line 31 of file nd_item.hpp.

Inheritance diagram for trisycl::nd_item< Dimensions >:



Collaboration diagram for trisycl::nd_item< Dimensions >:



Public Member Functions

- nd_item (nd_range< Dimensions > ndr)

  Create an empty nd_item<> from an nd_range<>
- nd_item (id< Dimensions > global_index, nd_range< Dimensions > ndr)

  Create a full nd_item.
- nd_item ()=default

  To be able to copy and assign nd_item, use default constructors too.
- id< Dimensions > get_global_id () const

  Return the constituent global id representing the work-item's position in the global iteration space.
- size_t get_global_id (int dimension) const

  Return the constituent element of the global id representing the work-item's position in the global iteration space in the given dimension.
- size_t get_global_linear_id () const

  Return the flattened id of the current work-item after subtracting the offset.
- id< Dimensions > get_local_id () const

  Return the constituent local id representing the work-item's position within the current work-group.
- size_t get_local_id (int dimension) const

  Return the constituent element of the local id representing the work-item's position within the current work-group in the given dimension.
- size_t get_local_linear_id () const

  Return the flattened id of the current work-item within the current work-group.
- id< Dimensions > get_group () const

  Return the constituent group representing the work-group's position within the overall nd_range.
- size_t get_group (int dimension) const

  Return the constituent element of the group id representing the work-group;s position within the overall nd_range in the given dimension.
- size_t get_group_linear_id () const

  Return the flattened id of the current work-group.
- id< Dimensions > get_group_range () const

  Return the number of groups in the nd_range.
- size_t get_group_range (int dimension) const

  Return the number of groups for dimension in the nd_range.
- range< Dimensions > get_global_range () const

  Return a range<> representing the dimensions of the nd_range<>
- range< Dimensions > get_local_range () const

  Return a range<> representing the dimensions of the current work-group.
- id< Dimensions > get_offset () const

Return an id<> representing the n-dimensional offset provided to the constructor of the nd_range<> and that is added by the runtime to the global-ID of each work-item.

- nd_range< Dimensions > get_nd_range () const

  Return the nd_range<> of the current execution.

- item< Dimensions > get_item () const

  Allows projection down to an item.

- void barrier (access::fence_space flag=access::fence_space::global_and_local) const

  Execute a barrier with memory ordering on the local address space, global address space or both based on the value of flag.

- void set_local (id< Dimensions > Index)
- void set_global (id< Dimensions > Index)
- bool operator== (const nd_item< Dimensions > &nd_itemB) const

Static Public Member Functions

- static auto constexpr rank ()

  The number of dimensions of the nd_item.

Private Attributes

- id< Dimensions > global_index
- id< Dimensions > local_index
- nd_range< Dimensions > ND_range

9.19.2.5.1 Constructor & Destructor Documentation

9.19.2.5.1.1 nd_item() [1/3] template<int Dimensions = 1>

```
trisycl::nd_item< Dimensions >::nd_item (
            nd_range< Dimensions > ndr )  [inline]
```

Create an empty nd_item<> from an nd_range<>

Todo This is for the triSYCL implementation which is expected to call set_global() and set_local() later. This should be hidden to the user.

Definition at line 51 of file nd_item.hpp.

```
00051 :  ND_range { ndr } {}
```

9.19.2.5.1.2 nd_item() [2/3] template<int Dimensions = 1>

```
trisycl::nd_item< Dimensions >::nd_item (
            id< Dimensions > global_index,
            nd_range< Dimensions > ndr )  [inline]
```

Create a full nd_item.

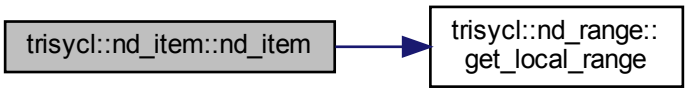
Todo This is for validation purpose. Hide this to the programmer somehow

Definition at line 59 of file nd_item.hpp.

```
00060                                  :
00061     global_index { global_index },
00062     // Compute the local index using the offset and the group size
00063     local_index { (global_index - ndr.get_offset())%id<Dimensions> {
00064         ndr.get_local_range() } },
00065     ND_range { ndr }
00066   {}
```

References trisycl::nd_range< Dimensions >::get_local_range().

Here is the call graph for this function:



9.19.2.5.1.3 nd_item() [3/3] template<int Dimensions = 1>

```
trisycl::nd_item< Dimensions >::nd_item ( )  [default]
```

To be able to copy and assign nd_item, use default constructors too.

Todo Make most of them protected, reserved to implementation

9.19.2.5.2 Member Function Documentation

9.19.2.5.2.1 barrier() template<int Dimensions = 1>

```
void trisycl::nd_item< Dimensions >::barrier (
            access::fence_space flag = access::fence_space::global_and_local ) const  [inline]
```

Execute a barrier with memory ordering on the local address space, global address space or both based on the value of flag.

The current work-item will wait at the barrier until all work-items in the current work-group have reached the barrier.

In addition, the barrier performs a fence operation ensuring that all memory accesses in the specified address space issued before the barrier complete before those issued after the barrier

Definition at line 202 of file nd_item.hpp.

```
00203                                                 {
00204 #if defined(_OPENMP) && !defined(TRISYCL_NO_BARRIER)
00205     /* Use OpenMP barrier in the implementation with 1 OpenMP thread per
00206 work-item of the work-group */
00207 #pragma omp barrier
00208 #else
00209     // \todo To be implemented efficiently otherwise
00210     TRISYCL_UNIMPL;
00211 #endif
00212   }
```

References TRISYCL_UNIMPL.

9.19.2.5.2.2 get_global_id() [1/2] template<int Dimensions = 1>

```
id< Dimensions > trisycl::nd_item< Dimensions >::get_global_id ( ) const  [inline]
```

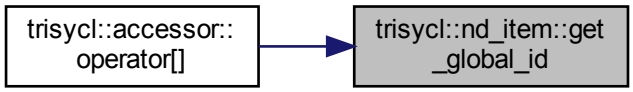
Return the constituent global id representing the work-item's position in the global iteration space.

Definition at line 79 of file nd_item.hpp.

```
00079 { return global_index; }
```

Referenced by trisycl::accessor< DataType, Dimensions, AccessMode, Target >::operator[]().

Here is the caller graph for this function:



9.19.2.5.2.3 get_global_id() [2/2] template<int Dimensions = 1>

```
size_t trisycl::nd_item< Dimensions >::get_global_id (
            int dimension ) const  [inline]
```

Return the constituent element of the global id representing the work-item's position in the global iteration space in the given dimension.

Definition at line 86 of file nd_item.hpp.

```
00086                                           {
00087     return get_global_id()[dimension];
00088   }
```

9.19.2.5.2.4 get_global_linear_id() template<int Dimensions = 1>

```
size_t trisycl::nd_item< Dimensions >::get_global_linear_id ( ) const  [inline]
```

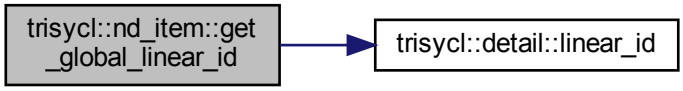
Return the flattened id of the current work-item after subtracting the offset.

Definition at line 94 of file nd_item.hpp.

```
00094                                       {
00095     return detail::linear_id(get_global_range(),
00096                              get_global_id(),
00097                              get_offset());
00098   }
```

References trisycl::detail::linear_id().

Here is the call graph for this function:



9.19.2.5.2.5 get_global_range() template<int Dimensions = 1>

```
range< Dimensions > trisycl::nd_item< Dimensions >::get_global_range ( ) const  [inline]
```

Return a range<> representing the dimensions of the nd_range<>

Definition at line 161 of file nd_item.hpp.

```
00161                                          {
00162     return get_nd_range().get_global_range();
00163   }
```

9.19.2.5.2.6 get_group() [1/2] template<int Dimensions = 1>

```
id< Dimensions > trisycl::nd_item< Dimensions >::get_group ( ) const  [inline]
```

Return the constituent group representing the work-group's position within the overall nd_range.

Definition at line 127 of file nd_item.hpp.

```
00127                                    {
00128     /* Convert get_local_range() to an id<> to remove ambiguity into using
00129 implicit conversion either from range<> to id<> or the opposite */
00130     return get_global_id()/id<Dimensions> { get_local_range() };
00131   }
```

9.19.2.5.2.7 get_group() [2/2] template<int Dimensions = 1>

```
size_t trisycl::nd_item< Dimensions >::get_group (
            int dimension ) const  [inline]
```

Return the constituent element of the group id representing the work-group;s position within the overall nd_range in the given dimension.

Definition at line 138 of file nd_item.hpp.

```
00138                                          {
00139     return get_group()[dimension];
00140   }
```

9.19.2.5.2.8 get_group_linear_id() template<int Dimensions = 1>

```
size_t trisycl::nd_item< Dimensions >::get_group_linear_id ( ) const  [inline]
```

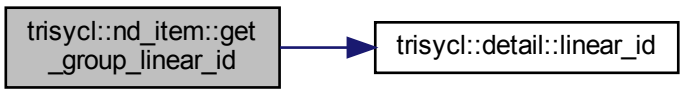
Return the flattened id of the current work-group.

Definition at line 144 of file nd_item.hpp.

```
00144                                       {
00145     return detail::linear_id(get_group_range(), get_group());
00146   }
```

References trisycl::detail::linear_id().

Here is the call graph for this function:



9.19.2.5.2.9 get_group_range() [1/2] template<int Dimensions = 1>

```
id< Dimensions > trisycl::nd_item< Dimensions >::get_group_range ( ) const  [inline]
```

Return the number of groups in the nd_range.

Definition at line 150 of file nd_item.hpp.

```
00150                                            {
00151     return get_nd_range().get_group_range();
00152   }
```

9.19.2.5.2.10 get_group_range() [2/2] template<int Dimensions = 1>

```
size_t trisycl::nd_item< Dimensions >::get_group_range (
            int dimension ) const  [inline]
```

Return the number of groups for dimension in the nd_range.

Definition at line 155 of file nd_item.hpp.

```
00155                                              {
00156       return get_group_range()[dimension];
00157     }
```

9.19.2.5.2.11 get_item() template<int Dimensions = 1>

```
item< Dimensions > trisycl::nd_item< Dimensions >::get_item ( ) const  [inline]
```

Allows projection down to an item.

Todo Add to the specification

Definition at line 187 of file nd_item.hpp.

```
00187                                       {
00188       return { get_global_range(), get_global_id(), get_offset() };
00189     }
```

9.19.2.5.2.12 get_local_id() [1/2] template<int Dimensions = 1>

```
id< Dimensions > trisycl::nd_item< Dimensions >::get_local_id ( ) const  [inline]
```

Return the constituent local id representing the work-item's position within the current work-group.

Definition at line 104 of file nd_item.hpp.

```
00104 { return local_index; }
```

9.19.2.5.2.13 get_local_id() [2/2] template<int Dimensions = 1>

```
size_t trisycl::nd_item< Dimensions >::get_local_id (
            int dimension ) const  [inline]
```

Return the constituent element of the local id representing the work-item's position within the current work-group in the given dimension.

Definition at line 111 of file nd_item.hpp.

```
00111                                            {
00112       return get_local_id()[dimension];
00113     }
```

9.19.2.5.2.14 get_local_linear_id() template<int Dimensions = 1>

```
size_t trisycl::nd_item< Dimensions >::get_local_linear_id ( ) const  [inline]
```

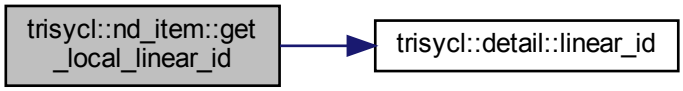
Return the flattened id of the current work-item within the current work-group.

Definition at line 119 of file nd_item.hpp.

```
00119                                  {
00120     return detail::linear_id(get_local_range(), get_local_id());
00121   }
```

References trisycl::detail::linear_id().

Here is the call graph for this function:



9.19.2.5.2.15 get_local_range() template<int Dimensions = 1>

```
range< Dimensions > trisycl::nd_item< Dimensions >::get_local_range ( ) const  [inline]
```

Return a range<> representing the dimensions of the current work-group.

Definition at line 167 of file nd_item.hpp.

```
00167                                        {
00168     return get_nd_range().get_local_range();
00169   }
```

9.19.2.5.2.16 get_nd_range() template<int Dimensions = 1>

```
nd_range< Dimensions > trisycl::nd_item< Dimensions >::get_nd_range ( ) const  [inline]
```

Return the nd_range<> of the current execution.

Definition at line 180 of file nd_item.hpp.

```
00180 { return ND_range; }
```

9.19.2.5.2.17 get_offset() template<int Dimensions = 1>

```
id< Dimensions > trisycl::nd_item< Dimensions >::get_offset ( ) const  [inline]
```

Return an id<> representing the n-dimensional offset provided to the constructor of the nd_range<> and that is added by the runtime to the global-ID of each work-item.

Definition at line 176 of file nd_item.hpp.

```
00176 { return get_nd_range().get_offset(); }
```

9.19.2.5.2.18 operator==() template<int Dimensions = 1>

```
bool trisycl::nd_item< Dimensions >::operator== (
            const nd_item< Dimensions > & nd_itemB ) const  [inline]
```

Definition at line 223 of file nd_item.hpp.

```
00223                                                       {
00224     return (ND_range == nd_itemB.ND_range &&
00225         global_index == nd_itemB.global_index);
00226   }
```

References trisycl::nd_item< Dimensions >::global_index, and trisycl::nd_item< Dimensions >::ND_range.

9.19.2.5.2.19 rank() template<int Dimensions = 1>

```
static auto constexpr trisycl::nd_item< Dimensions >::rank ( )  [inline], [static], [constexpr]
```

The number of dimensions of the nd_item.

Definition at line 33 of file nd_item.hpp.

```
00033 { return Dimensions; }
```

9.19.2.5.2.20 set_global() template<int Dimensions = 1>

```
void trisycl::nd_item< Dimensions >::set_global (
            id< Dimensions > Index )  [inline]
```

Definition at line 220 of file nd_item.hpp.

```
00220 { global_index = Index; }
```

9.19.2.5.2.21 set_local() template<int Dimensions = 1>

```
void trisycl::nd_item< Dimensions >::set_local (
            id< Dimensions > Index )  [inline]
```

Definition at line 216 of file nd_item.hpp.

```
00216 { local_index = Index; }
```

9.19.2.5.3 Member Data Documentation

9.19.2.5.3.1 global_index template<int Dimensions = 1>

```
id<Dimensions> trisycl::nd_item< Dimensions >::global_index  [private]
```

Definition at line 37 of file nd_item.hpp.

Referenced by trisycl::nd_item< Dimensions >::operator==().

9.19.2.5.3.2 local_index `template<int Dimensions = 1>`

`id<Dimensions> trisycl::nd_item< Dimensions >::local_index [private]`

Definition at line 40 of file nd_item.hpp.

9.19.2.5.3.3 ND_range `template<int Dimensions = 1>`

`nd_range<Dimensions> trisycl::nd_item< Dimensions >::ND_range [private]`

Definition at line 41 of file nd_item.hpp.

Referenced by trisycl::nd_item< Dimensions >::operator==().

9.19.2.6 struct trisycl::nd_range

template<int Dimensions = 1>
struct trisycl::nd_range< Dimensions >

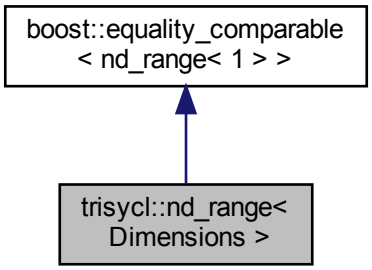
A ND-range, made by a global and local range, to specify work-group and work-item organization.

The local offset is used to translate the iteration space origin if needed.

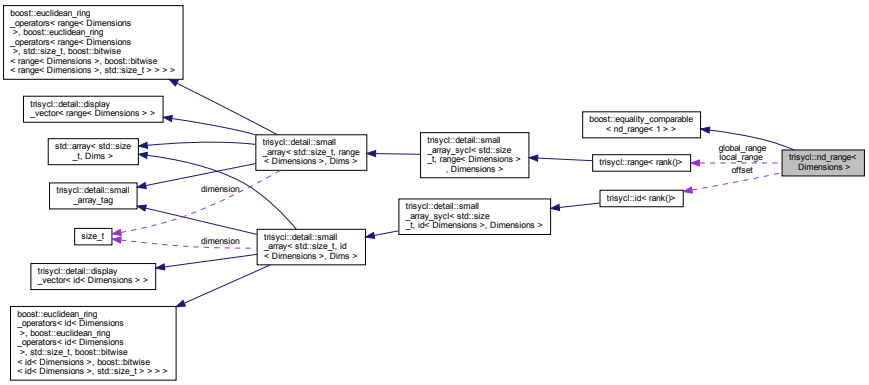
Todo add copy constructors in the specification

Definition at line 32 of file nd_range.hpp.

Inheritance diagram for trisycl::nd_range< Dimensions >:



Collaboration diagram for trisycl::nd_range< Dimensions >:



Public Member Functions

- nd_range (range< Dimensions > global_size, range< Dimensions > local_size, id< Dimensions > offset={})

  Construct a ND-range with all the details available in OpenCL.
- range< Dimensions > get_global_range () const

  Get the global iteration space range.
- range< Dimensions > get_local_range () const

  Get the local part of the iteration space range.
- auto get_group_range () const

  Get the range of work-groups needed to run this ND-range.
- id< Dimensions > get_offset () const
- void display () const

  Display the value for debugging and validation purpose.
- bool operator== (const nd_range &nd_rangeB) const

  Comparison operators for nd_range.

Static Public Member Functions

- static auto constexpr rank ()

  The number of dimensions of the nd_range.

Private Attributes

- range< rank()> global_range
- range< rank()> local_range
- id< rank()> offset

9.19.2.6.1 Constructor & Destructor Documentation

9.19.2.6.1.1 nd_range() template<int Dimensions = 1>

```
trisycl::nd_range< Dimensions >::nd_range (
            range< Dimensions > global_size,
            range< Dimensions > local_size,
            id< Dimensions > offset = {} )  [inline]
```

Construct a ND-range with all the details available in OpenCL.

By default use a zero offset, that is iterations start at 0

Definition at line 48 of file nd_range.hpp.

```
00050                                {}) :
00051     global_range { global_size }, local_range { local_size }, offset { offset }
00052   { }
```

9.19.2.6.2 Member Function Documentation

9.19.2.6.2.1 display() template<int Dimensions = 1>

```
void trisycl::nd_range< Dimensions >::display ( ) const  [inline]
```

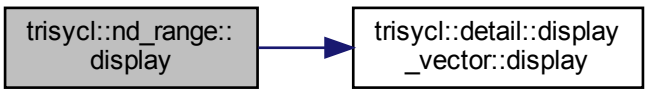
Display the value for debugging and validation purpose.

Definition at line 78 of file nd_range.hpp.

```
00078                        {
00079     global_range.display();
00080     local_range.display();
00081     offset.display();
00082   }
```

References trisycl::detail::display_vector< T >::display(), trisycl::nd_range< Dimensions >::global_range, trisycl::nd_range< Dimensions >::local_range, and trisycl::nd_range< Dimensions >::offset.

Here is the call graph for this function:



9.19.2.6.2.2 get_global_range() template<int Dimensions = 1>

range< Dimensions > trisycl::nd_range< Dimensions >::get_global_range ( ) const [inline]

Get the global iteration space range.

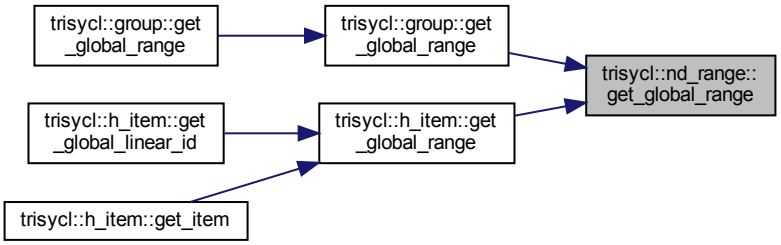
Definition at line 56 of file nd_range.hpp.

```
00056 { return global_range; }
```

References trisycl::nd_range< Dimensions >::global_range.

Referenced by trisycl::group< Dimensions >::get_global_range(), and trisycl::h_item< Dimensions >::get_global_range().

Here is the caller graph for this function:



9.19.2.6.2.3 get_group_range() template<int Dimensions = 1>

auto trisycl::nd_range< Dimensions >::get_group_range ( ) const [inline]

Get the range of work-groups needed to run this ND-range.

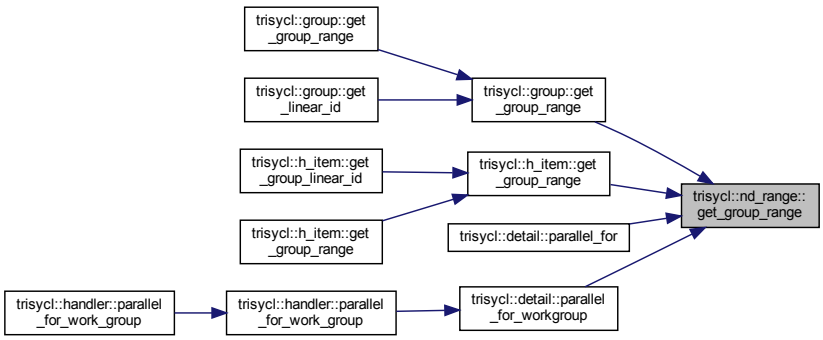
Definition at line 64 of file nd_range.hpp.

```
00064                                {
00065     /* This is basically global_range/local_range, round up to the
00066 next integer, in case the global range is not a multiple of the
00067 local range.  Note this is a motivating example to build a range
00068 from a scalar with a broadcasting constructor.  */
00069     return (global_range + local_range - range<Dimensions>{ 1 })/local_range;
00070   }
```

References trisycl::nd_range< Dimensions >::global_range, and trisycl::nd_range< Dimensions >::local_range.

Referenced by trisycl::group< Dimensions >::get_group_range(), trisycl::h_item< Dimensions >::get_group_range(), trisycl::detail::parallel_for(), and trisycl::detail::parallel_for_workgroup().

Here is the caller graph for this function:



9.19.2.6.2.4 get_local_range() template<int Dimensions = 1>

range< Dimensions > trisycl::nd_range< Dimensions >::get_local_range ( ) const [inline]

Get the local part of the iteration space range.

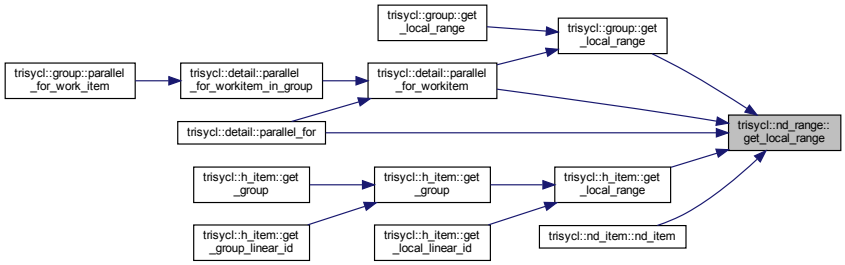
Definition at line 60 of file nd_range.hpp.

```
00060 { return local_range; }
```

References trisycl::nd_range< Dimensions >::local_range.

Referenced by trisycl::group< Dimensions >::get_local_range(), trisycl::h_item< Dimensions >::get_local_range(), trisycl::nd_item< Dimensions >::nd_item(), trisycl::detail::parallel_for(), and trisycl::detail::parallel_for_workitem().

Here is the caller graph for this function:



9.19.2.6.2.5 get_offset() template<int Dimensions = 1>

id< Dimensions > trisycl::nd_range< Dimensions >::get_offset ( ) const [inline]

Todo get_offset() is lacking in the specification

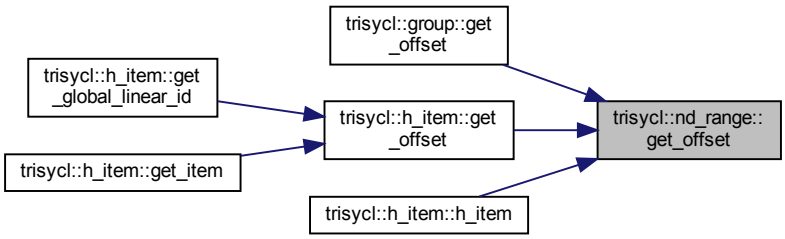
Definition at line 74 of file nd_range.hpp.

```
00074 { return offset; }
```

References trisycl::nd_range< Dimensions >::offset.

Referenced by trisycl::group< Dimensions >::get_offset(), trisycl::h_item< Dimensions >::get_offset(), and trisycl::h_item< Dimensions >::h_item().

Here is the caller graph for this function:



9.19.2.6.2.6 operator==() template<int Dimensions = 1>

```
bool trisycl::nd_range< Dimensions >::operator== (
            const nd_range< Dimensions > & nd_rangeB ) const  [inline]
```

Comparison operators for nd_range.

Definition at line 85 of file nd_range.hpp.

```
00085                                                {
00086     return (global_range == nd_rangeB.global_range &&
00087             offset == nd_rangeB.offset);
00088   }
```

References trisycl::nd_range< Dimensions >::global_range, and trisycl::nd_range< Dimensions >::offset.

9.19.2.6.2.7 rank() template<int Dimensions = 1>

```
static auto constexpr trisycl::nd_range< Dimensions >::rank ( )  [inline], [static], [constexpr]
```

The number of dimensions of the nd_range.

Definition at line 34 of file nd_range.hpp.

```
00034 { return Dimensions; }
```

9.19.2.6.3 Member Data Documentation

9.19.2.6.3.1 global_range template<int Dimensions = 1>

```
range<rank()> trisycl::nd_range< Dimensions >::global_range  [private]
```

Definition at line 38 of file nd_range.hpp.

Referenced by trisycl::nd_range< Dimensions >::display(), trisycl::nd_range< Dimensions >::get_global_range(), trisycl::nd_range< Dimensions >::get_group_range(), and trisycl::nd_range< Dimensions >::operator==().

9.19.2.6.3.2 local_range template<int Dimensions = 1>

```
range<rank()> trisycl::nd_range< Dimensions >::local_range  [private]
```

Definition at line 39 of file nd_range.hpp.

Referenced by trisycl::nd_range< Dimensions >::display(), trisycl::nd_range< Dimensions >::get_group_range(), and trisycl::nd_range< Dimensions >::get_local_range().

9.19.2.6.3.3 offset template<int Dimensions = 1>

```
id<rank()> trisycl::nd_range< Dimensions >::offset  [private]
```

Definition at line 40 of file nd_range.hpp.

Referenced by trisycl::nd_range< Dimensions >::display(), trisycl::nd_range< Dimensions >::get_offset(), and trisycl::nd_range< Dimensions >::operator==().

9.19.2.7 struct trisycl::detail::parallel_for_iterate

template<std::size_t level, typename Range, typename ParallelForFunctor, typename Id>
struct trisycl::detail::parallel_for_iterate< level, Range, ParallelForFunctor, Id >

A recursive multi-dimensional iterator that ends up calling f.

The iteration order may be changed later.

Since partial specialization of function template is not possible in C++14, use a class template instead with everything in the constructor.

Definition at line 54 of file parallelism.hpp.

Public Member Functions

- parallel_for_iterate (Range r, ParallelForFunctor &f, Id &index)

9.19.2.7.1 Constructor & Destructor Documentation

9.19.2.7.1.1 parallel_for_iterate() `template<std::size_t level, typename Range , typename ParallelForFunctor , typename Id >`

`trisycl::detail::parallel_for_iterate< level, Range, ParallelForFunctor, Id >::parallel_for_iterate (`

```
Range r,
ParallelForFunctor & f,
Id & index )  [inline]
```

Definition at line 55 of file parallelism.hpp.

```
00055                                                                    {
00056     for (std::size_t _sycl_index = 0,
00057            _sycl_end = r[r.rank() - level];
00058          _sycl_index < _sycl_end;
00059          _sycl_index++) {
00060       // Set the current value of the index for this dimension
00061       index[r.rank() - level] = _sycl_index;
00062       // Iterate further on lower dimensions
00063       parallel_for_iterate<level - 1,
00064                            Range,
00065                            ParallelForFunctor,
00066                            Id> { r, f, index };
00067     }
00068   }
```

9.19.2.8 struct trisycl::detail::parallel_for_iterate< 0, Range, ParallelForFunctor, Id >

template<typename Range, typename ParallelForFunctor, typename Id>
struct trisycl::detail::parallel_for_iterate< 0, Range, ParallelForFunctor, Id >

Stop the recursion when level reaches 0 by simply calling the kernel functor with the constructed id.

Definition at line 116 of file parallelism.hpp.

Public Member Functions

- parallel_for_iterate (Range r, ParallelForFunctor &f, Id &index)

9.19.2.8.1 Constructor & Destructor Documentation

9.19.2.8.1.1 parallel_for_iterate() `template<typename Range , typename ParallelForFunctor , typename Id >`

```
trisycl::detail::parallel_for_iterate< 0, Range, ParallelForFunctor, Id >::parallel_for_↩
iterate (
            Range r,
            ParallelForFunctor & f,
            Id & index )  [inline]
```

Definition at line 117 of file parallelism.hpp.

```
00117                                                                    {
00118     f(index);
00119   }
```

9.19.2.9 class trisycl::range

template< int Dimensions = 1 >
class trisycl::range< Dimensions >

A SYCL range defines a multi-dimensional index range that can be used to define launch parallel computation extent or buffer sizes.

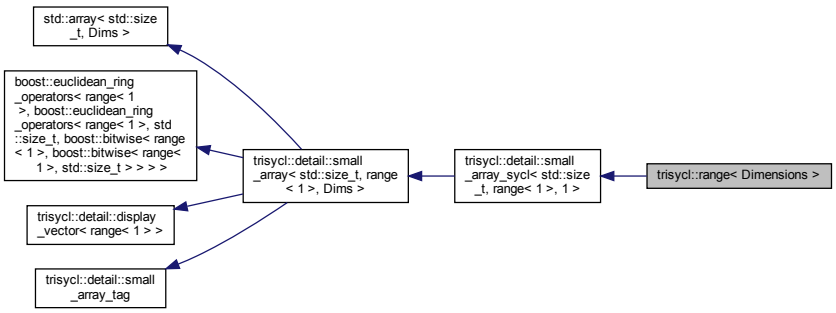
Todo use std::size_t Dimensions instead of int Dimensions in the specification?

Todo add to the specification this default parameter value?

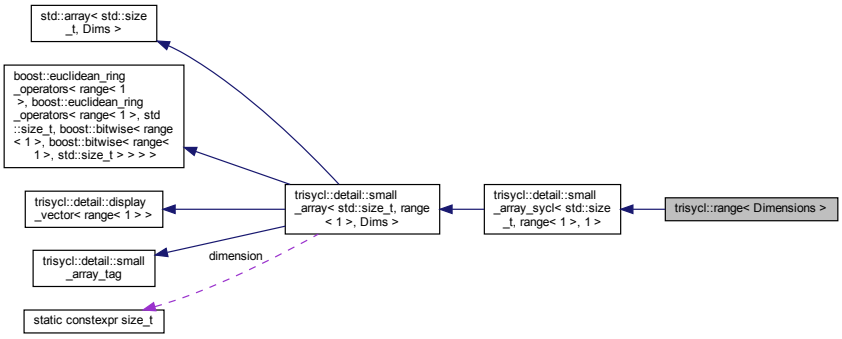
Todo add to the specification some way to specify an offset?

Definition at line 35 of file range.hpp.

Inheritance diagram for trisycl::range< Dimensions >:



Collaboration diagram for trisycl::range< Dimensions >:



Public Member Functions

- range ()=default
- size_t size () const

  Return the number of elements in the range.

Static Public Member Functions

- static auto constexpr rank ()

  The number of dimensions of the range.

Additional Inherited Members

9.19.2.9.1 Constructor & Destructor Documentation

9.19.2.9.1.1 range() `template<int Dimensions = 1>`

`trisycl::range< Dimensions >::range ( ) [default]`

9.19.2.9.2 Member Function Documentation

9.19.2.9.2.1 rank() `template<int Dimensions = 1>`

`static auto constexpr trisycl::range< Dimensions >::rank ( ) [inline], [static], [constexpr]`

The number of dimensions of the range.

Definition at line 49 of file range.hpp.

```
00049 { return Dimensions; }
```

9.19.2.9.2.2 size() template<int Dimensions = 1>

```
size_t trisycl::range< Dimensions >::size ( ) const  [inline]
```

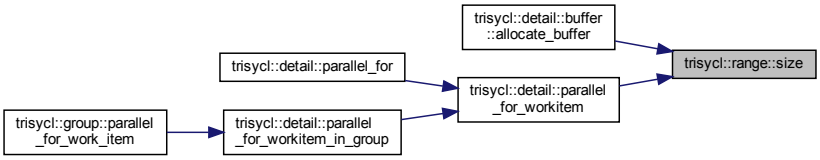
Return the number of elements in the range.

Definition at line 53 of file range.hpp.

```
00053                       {
00054     // Return the product of the sizes in each dimension
00055     return std::accumulate(this->cbegin(),
00056                            this->cend(),
00057                            1,
00058                            std::multiplies<size_t> {});
00059   }
```

Referenced by trisycl::detail::buffer< T, Dimensions >::allocate_buffer(), and trisycl::detail::parallel_for_workitem().

Here is the caller graph for this function:



9.19.3 Function Documentation

9.19.3.1 capture_arg_v()

```
template<typename F , typename R , typename A >
auto trisycl::detail::capture_arg_v (
             R(F::*)(A)  )
```

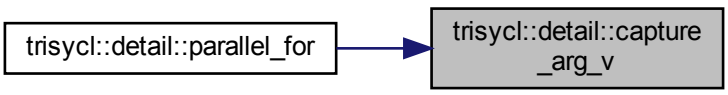
```
#include <include/triSYCL/parallelism/detail/parallelism.hpp>
```

Definition at line 198 of file parallelism.hpp.

```
00198 { return A{}; }
```

Referenced by trisycl::detail::parallel_for().

Here is the caller graph for this function:



9.19.3.2 id()

```
template<typename...  BasicType>
trisycl::id (
            BasicType...  Args ) -> id< sizeof...(Args)>
```

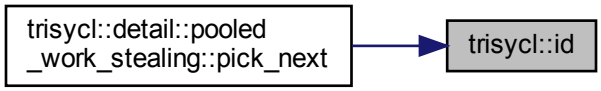
`#include <include/triSYCL/id.hpp>`

User-defined deduction guides to deduce the rank of an id from the number of arguments.

For example id { 4, 5, 7 } will define an id<3> { 4, 5, 7 }

Referenced by trisycl::detail::pooled_work_stealing::pick_next().

Here is the caller graph for this function:



9.19.3.3 parallel_for() [1/4]

```
template<int Dimensions = 1, typename ParallelForFunctor >
void trisycl::detail::parallel_for (
            nd_range< Dimensions > r,
            ParallelForFunctor f )
```

`#include <include/triSYCL/parallelism/detail/parallelism.hpp>`

Implement a variation of parallel_for to take into account a nd_range<>

Todo Add an OpenMP implementation

Todo Deal with incomplete work-groups

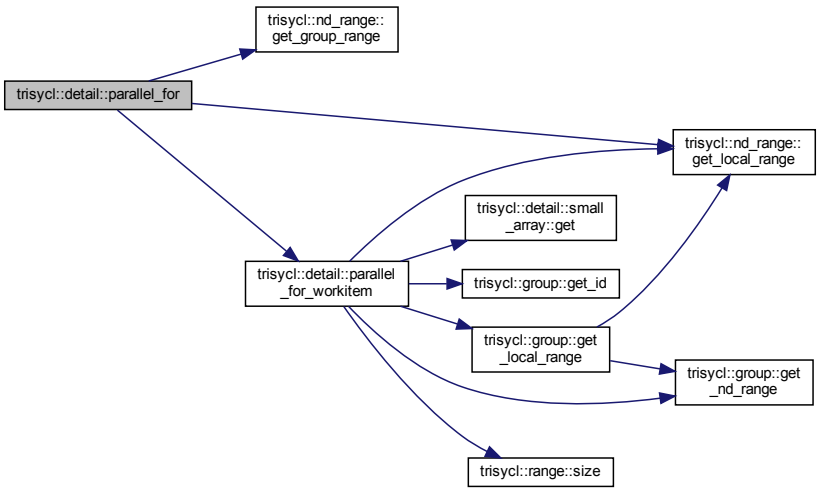
Todo Implement with parallel_for_workgroup()/parallel_for_workitem()

Definition at line 388 of file parallelism.hpp.

```
00389                                     {
00390   // To iterate on the work-group
00391   id<Dimensions> group;
00392   range<Dimensions> group_range = r.get_group_range();
00393
00394 #ifdef _OPENMP
00395
00396   auto iterate_in_work_group = [&] (id<Dimensions> g) {
00397     //group.display();
00398
00399     // Then iterate on the local work-groups
00400     trisycl::group<Dimensions> wg {g, r};
00401     parallel_for_workitem<Dimensions,
00402                           nd_item<Dimensions>,
00403                           decltype(f)>(wg, f);
00404   };
00405
00406 #else
00407
00408   // In a sequential execution there is only one index processed at a time
00409   nd_item<Dimensions> index { r };
00410
00411   // To iterate on the local work-item
00412   id<Dimensions> local;
00413   range<Dimensions> local_range = r.get_local_range();
00414
00415   // Reconstruct the item from its group and local id
00416   auto reconstruct_item = [&] (id<Dimensions> l) {
00417     //local.display();
00418     // Reconstruct the global item
00419     index.set_local(local);
00420     // Upgrade local_range to an id<> so that we can * with the group (an id<>)
00421     index.set_global(local + id<Dimensions>(local_range)*group);
00422     // Call the user kernel at last
00423     f(index);
00424   };
00425
00426   /* To recycle the parallel_for on range<>, wrap the ParallelForFunctor f
00427 into another functor that iterates inside the work-group and then
00428 calls f */
00429   auto iterate_in_work_group = [&] (id<Dimensions> g) {
00430     //group.display();
00431     // Then iterate on the local work-groups
00432     parallel_for_iterate<Dimensions,
00433                          range<Dimensions>,
00434                          decltype(reconstruct_item),
00435                          id<Dimensions>> { local_range,
00436                                            reconstruct_item,
00437                                            local };
00438   };
00439
00440 #endif
00441
00442   // First iterate on all the work-groups
00443   parallel_for_iterate<Dimensions,
00444                        range<Dimensions>,
00445                        decltype(iterate_in_work_group),
00446                        id<Dimensions>> { group_range,
00447                                          iterate_in_work_group,
00448                                          group };
00449 }
```

References trisycl::nd_range< Dimensions >::get_group_range(), trisycl::nd_range< Dimensions >::get_local_range(), and trisycl::detail::parallel_for_workitem().

Here is the call graph for this function:



9.19.3.4 parallel_for() [2/4]

```
template<int Dimensions = 1, typename ParallelForFunctor >
void trisycl::detail::parallel_for (
            range< Dimensions > r,
            ParallelForFunctor f )
```

`#include <include/triSYCL/parallelism/detail/parallelism.hpp>`

Calls the appropriate ternary parallel_for overload based on the index type of the kernel function object f.

capture_arg_v generates a value from the type of the passed in kernels (lambdas) argument (item or id in this case) and passes it onwards to parallel_for so that it's aware which index class it's supposed to generate data for.

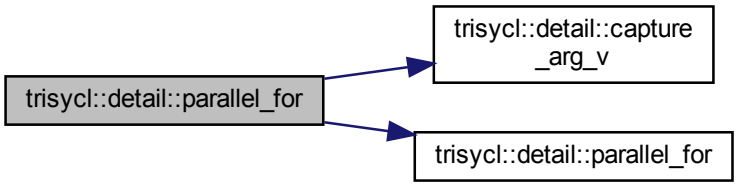
We erase the type then retrieve it again so that we can avoid redefining the parallel_for interface in handler.hpp for item and id as the parallel_for in the SYCL 1.2.1 spec that accepts a range can be overloaded for both.

Definition at line 218 of file parallelism.hpp.

```
00218                                                    {
00219   parallel_for(r,f, capture_arg_v(&ParallelForFunctor::operator()));
00220 }
```

References trisycl::detail::capture_arg_v(), and trisycl::detail::parallel_for().

Here is the call graph for this function:



9.19.3.5 parallel_for() [3/4]

```
template<int Dimensions = 1, typename ParallelForFunctor , typename Id >
void trisycl::detail::parallel_for (
             range< Dimensions > r,
             ParallelForFunctor f,
             Id  )
```

`#include <include/triSYCL/parallelism/detail/parallelism.hpp>`

Implementation of a data parallel computation with parallelism specified at launch time by a range<>.

Kernel index is id or int.

This implementation use OpenMP 3 if compiled with the right flag.

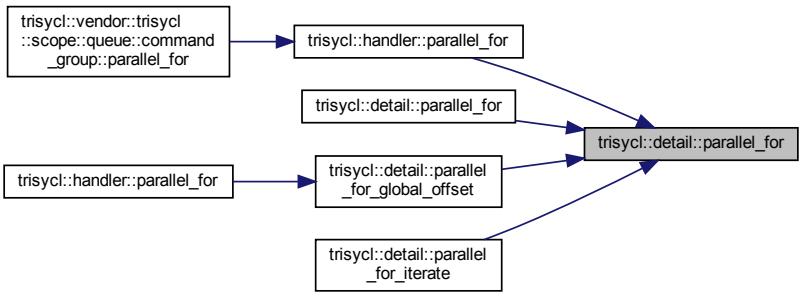
Kernel index is id or int.

Definition at line 129 of file parallelism.hpp.

```
00131                       {
00132 #ifdef _OPENMP
00133   // Use OpenMP for the top loop level
00134   parallel_OpenMP_for_iterate<Dimensions,
00135                               range<Dimensions>,
00136                               ParallelForFunctor,
00137                               id<Dimensions» { r, f };
00138 #else
00139   // In a sequential execution there is only one index processed at a time
00140   id<Dimensions> index;
00141   parallel_for_iterate<Dimensions,
00142                        range<Dimensions>,
00143                        ParallelForFunctor,
00144                        id<Dimensions» { r, f, index };
00145 #endif
00146 }
```

Referenced by trisycl::handler::parallel_for(), trisycl::detail::parallel_for(), trisycl::detail::parallel_for_global_offset(), and trisycl::detail::parallel_for_iterate().

Here is the caller graph for this function:



9.19.3.6 parallel_for() [4/4]

```
template<int Dimensions = 1, typename ParallelForFunctor >
void trisycl::detail::parallel_for (
             range< Dimensions > r,
             ParallelForFunctor f,
             item< Dimensions >  )
```

`#include <include/triSYCL/parallelism/detail/parallelism.hpp>`

Implementation of a data parallel computation with parallelism specified at launch time by a range<>.

Kernel index is item.

This implementation use OpenMP 3 if compiled with the right flag.

Kernel index is item.

Definition at line 155 of file parallelism.hpp.

```
00157                                    {
00158   auto reconstruct_item = [&] (id<Dimensions> l) {
00159     // Reconstruct the global item
00160     item<Dimensions> index { r, l };
00161     // Call the user kernel with the item<> instead of the id<>
00162     f(index);
00163   };
00164 #ifdef _OPENMP
00165   // Use OpenMP for the top loop level
00166   parallel_OpenMP_for_iterate<Dimensions,
00167                               range<Dimensions>,
00168                               decltype(reconstruct_item),
00169                               id<Dimensions» { r, reconstruct_item };
00170 #else
00171   // In a sequential execution there is only one index processed at a time
00172   id<Dimensions> index;
00173   parallel_for_iterate<Dimensions,
00174                        range<Dimensions>,
00175                        decltype(reconstruct_item),
00176                        id<Dimensions» { r, reconstruct_item, index };
00177 #endif
00178 }
```

9.19.3.7 parallel_for_global_offset()

```
template<int Dimensions = 1, typename ParallelForFunctor >
void trisycl::detail::parallel_for_global_offset (
             range< Dimensions > global_size,
             id< Dimensions > offset,
             ParallelForFunctor f )
```

`#include <include/triSYCL/parallelism/detail/parallelism.hpp>`

Implementation of parallel_for with a range<> and an offset.

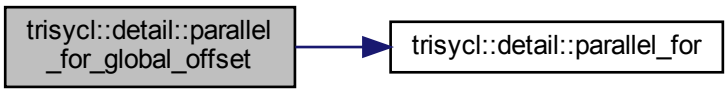
Definition at line 232 of file parallelism.hpp.

```
00234                                                    {
00235   // Reconstruct the item from its id<> and its offset
00236   auto reconstruct_item = [&] (id<Dimensions> l) {
00237     // Reconstruct the global item
00238     item<Dimensions> index { global_size, l + offset, offset };
00239     // Call the user kernel with the item<> instead of the id<>
00240     f(index);
00241   };
00242
00243   // First iterate on all the work-groups
00244   parallel_for(global_size, reconstruct_item);
00245 }
```

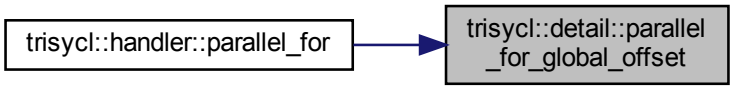
References offset, and trisycl::detail::parallel_for().

Referenced by trisycl::handler::parallel_for().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.3.8 parallel_for_iterate()

```
template<typename Range , typename ParallelForFunctor >
void trisycl::detail::parallel_for_iterate (
             Range r,
             ParallelForFunctor & f )
```

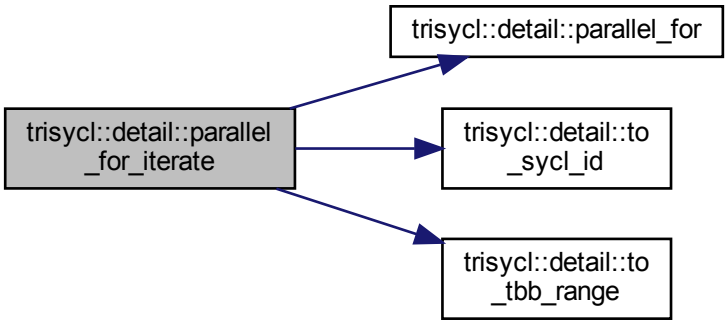
```
#include <include/triSYCL/parallelism/detail/parallelism_tbb.hpp>
```

Definition at line 70 of file parallelism_tbb.hpp.

```
00071 {
00072   tbb::parallel_for(to_tbb_range(r),
00073                     [&](auto idx) { f(to_sycl_id(idx)); },
00074                     tbb::simple_partitioner{});
00075 }
```

References trisycl::detail::parallel_for(), trisycl::detail::to_sycl_id(), and trisycl::detail::to_tbb_range().

Here is the call graph for this function:



9.19.3.9 parallel_for_workgroup()

```
template<int Dimensions = 1, typename ParallelForFunctor >
void trisycl::detail::parallel_for_workgroup (
             nd_range< Dimensions > r,
             ParallelForFunctor f )
```

```
#include <include/triSYCL/parallelism/detail/parallelism.hpp>
```

Implement the loop on the work-groups.

Definition at line 250 of file parallelism.hpp.

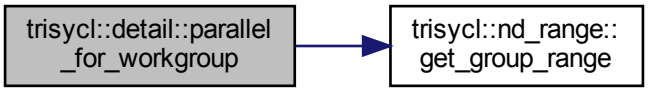
```
00251                                                 {
00252   // In a sequential execution there is only one index processed at a time
00253   group<Dimensions> g { r };
00254
00255   // First iterate on all the work-groups
00256   parallel_for_iterate<Dimensions,
00257                        range<Dimensions>,
```

```
00258                       ParallelForFunctor,
00259                       group<Dimensions» {
00260     r.get_group_range(),
00261     f,
00262     g };
00263 }
```

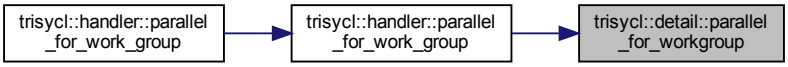
References trisycl::nd_range< Dimensions >::get_group_range().

Referenced by trisycl::handler::parallel_for_work_group().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.3.10 parallel_for_workitem()

```
template<int Dimensions, typename T_Item , typename ParallelForFunctor >
void trisycl::detail::parallel_for_workitem (
             const group< Dimensions > & g,
             ParallelForFunctor f )
```

`#include <include/triSYCL/parallelism/detail/parallelism.hpp>`

Implement the loop on the work-items inside a work-group.

Todo Better type the functor

Definition at line 271 of file parallelism.hpp.

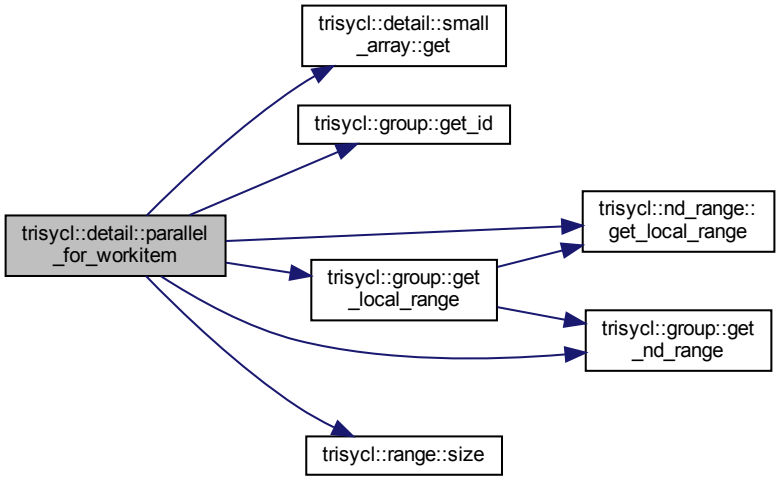
```
00272                                                 {
00273 #if defined(_OPENMP) && (!defined(TRISYCL_NO_BARRIER) && !defined(_MSC_VER))
00274   /* To implement barriers with OpenMP, one thread is created for each
00275 work-item in the group and thus an OpenMP barrier has the same effect
00276 of an OpenCL barrier executed by the work-items in a workgroup
00277
00278 The issue is that the parallel_for_workitem() execution is slow even
00279 when nd_item::barrier() is not used
00280 */
00281   range<Dimensions> l_r = g.get_nd_range().get_local_range();
00282   id<Dimensions> id_l_r { l_r };
00283
00284   auto tot = l_r.size();
00285
00286   if constexpr (Dimensions == 1) {
00287 #pragma omp parallel for collapse(1) schedule(static) num_threads(tot)
00288     for (size_t i = 0; i < l_r.get(0); ++i) {
00289       T_Item index{g.get_nd_range()};
00290       index.set_local(i);
00291       index.set_global(index.get_local_id() + id_l_r * g.get_id());
00292       f(index);
00293     }
00294   } else if constexpr (Dimensions == 2) {
00295 #pragma omp parallel for collapse(2) schedule(static) num_threads(tot)
00296     for (size_t i = 0; i < l_r.get(0); ++i) {
00297       for (size_t j = 0; j < l_r.get(1); ++j) {
00298         T_Item index{g.get_nd_range()};
00299         index.set_local({i,j});
00300         index.set_global(index.get_local_id() + id_l_r * g.get_id());
00301         f(index);
00302       }
00303     }
00304   } else if constexpr (Dimensions == 3) {
00305 #pragma omp parallel for collapse(3) schedule(static) num_threads(tot)
00306     for (size_t i = 0; i < l_r.get(0); ++i)
00307       for (size_t j = 0; j < l_r.get(1); ++j)
00308         for (size_t k = 0; k < l_r.get(2); ++k) {
00309           T_Item index{g.get_nd_range()};
00310           index.set_local({i,j,k});
00311           index.set_global(index.get_local_id() + id_l_r * g.get_id());
00312           f(index);
00313         }
00314   }
00315 #elif defined(_OPENMP) && (defined(TRISYCL_NO_BARRIER) && !defined(_MSC_VER))
00316   range<Dimensions> l_r = g.get_nd_range().get_local_range();
00317   id<Dimensions> id_l_r { l_r };
00318
00319   if constexpr (Dimensions == 1) {
00320 #pragma omp parallel for simd collapse(1)
00321     for (size_t i = 0; i < l_r.get(0); ++i) {
00322       T_Item index{g.get_nd_range()};
00323       index.set_local(i);
00324       index.set_global(index.get_local_id() + id_l_r * g.get_id());
00325       f(index);
00326     }
00327   } else if constexpr (Dimensions == 2) {
00328 #pragma omp parallel for simd collapse(2)
00329     for (size_t i = 0; i < l_r.get(0); ++i) {
00330       for (size_t j = 0; j < l_r.get(1); ++j) {
00331         T_Item index{g.get_nd_range()};
00332         index.set_local({i,j});
00333         index.set_global(index.get_local_id() + id_l_r * g.get_id());
00334         f(index);
00335       }
00336     }
00337   } else if constexpr (Dimensions == 3) {
00338 #pragma omp parallel for simd collapse(3)
00339     for (size_t i = 0; i < l_r.get(0); ++i)
00340       for (size_t j = 0; j < l_r.get(1); ++j)
00341         for (size_t k = 0; k < l_r.get(2); ++k) {
00342           T_Item index{g.get_nd_range()};
00343           index.set_local({i,j,k});
00344           index.set_global(index.get_local_id() + id_l_r * g.get_id());
00345           f(index);
00346         }
00347   }
00348 #else
00349   // In a sequential execution there is only one index processed at a time
00350   T_Item index { g.get_nd_range() };
00351   // To iterate on the local work-item
00352   id<Dimensions> local;
00353
00354   // Reconstruct the item from its group and local id
00355   auto reconstruct_item = [&] (id<Dimensions> l) {
00356     //local.display();
00357     //l.display();
```

```
00358     // Reconstruct the global item
00359     index.set_local(local);
00360     // \todo Some strength reduction here
00361     index.set_global(local + id<Dimensions>(g.get_local_range())*g.get_id());
00362     // Call the user kernel at last
00363     f(index);
00364   };
00365
00366   // Then iterate on all the work-items of the work-group
00367   parallel_for_iterate<Dimensions,
00368                        range<Dimensions>,
00369                        decltype(reconstruct_item),
00370                        id<Dimensions>> {
00371     g.get_local_range(),
00372     reconstruct_item,
00373     local };
00374 #endif
00375 }
```

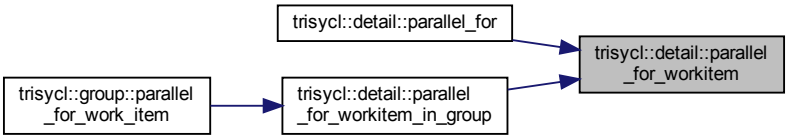
References trisycl::detail::small_array< BasicType, FinalType, Dims >::get(), trisycl::group< Dimensions >::get_id(), trisycl::group< Dimensions >::get_local_range(), trisycl::nd_range< Dimensions >::get_local_range(), trisycl::group< Dimensions > i, and trisycl::range< Dimensions >::size().

Referenced by trisycl::detail::parallel_for(), and trisycl::detail::parallel_for_workitem_in_group().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.3.11 parallel_for_workitem_in_group()

```
template<int Dimensions = 1, typename ParallelForFunctor >
void trisycl::detail::parallel_for_workitem_in_group (
             const group< Dimensions > & g,
             ParallelForFunctor f )
```

```
#include <include/triSYCL/group.hpp>
```

Implement the loop on the work-items inside a work-group.

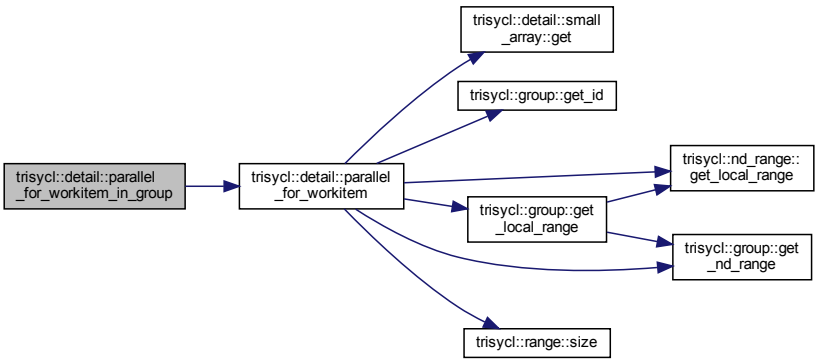
Definition at line 455 of file parallelism.hpp.

```
00456                                                                {
00457   parallel_for_workitem<Dimensions,
00458                         h_item<Dimensions>,
00459                         ParallelForFunctor>(g, f);
00460 }
```

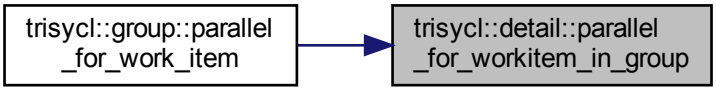
References trisycl::detail::parallel_for_workitem().

Referenced by trisycl::group< Dimensions >::parallel_for_work_item().

Here is the call graph for this function:



Here is the caller graph for this function:



9.19.3.12 range()

```
template<typename...  BasicType>
trisycl::range (
            BasicType...  Args ) -> range< sizeof...(Args)>
```

```
#include <include/triSYCL/range.hpp>
```

User-defined deduction guides to deduce the rank of a range from the number of arguments.

For example range { 4, 5, 7 } will define an range<3> { 4, 5, 7 }

9.19.3.13 to_sycl_id() [1/3]

```
static auto trisycl::detail::to_sycl_id (
            const tbb::blocked_range2d< size_t > & in )  [inline], [static]
```

```
#include <include/triSYCL/parallelism/detail/parallelism_tbb.hpp>
```

Definition at line 54 of file parallelism_tbb.hpp.

```
00055 {
00056   const auto cols = in.cols().begin();
00057   const auto rows = in.rows().begin();
00058   return make_id(rows, cols);
00059 }
```

9.19.3.14 to_sycl_id() [2/3]

```
static auto trisycl::detail::to_sycl_id (
            const tbb::blocked_range3d< size_t > & in )  [inline], [static]
```

```
#include <include/triSYCL/parallelism/detail/parallelism_tbb.hpp>
```

Definition at line 61 of file parallelism_tbb.hpp.

```
00062 {
00063   const auto pages = in.pages().begin();
00064   const auto cols = in.cols().begin();
00065   const auto rows = in.rows().begin();
00066   return make_id(rows, cols, pages);
00067 }
```

9.19.3.15 to_sycl_id() [3/3]

```
static auto trisycl::detail::to_sycl_id (
            const tbb::blocked_range< size_t > & in )  [inline], [static]
```

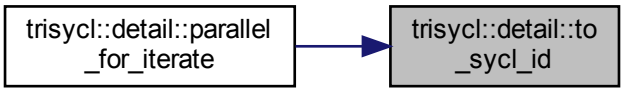
```
#include <include/triSYCL/parallelism/detail/parallelism_tbb.hpp>
```

Definition at line 49 of file parallelism_tbb.hpp.

```
00050 {
00051   return make_id(in.begin());
00052 }
```

Referenced by trisycl::detail::parallel_for_iterate().

Here is the caller graph for this function:



9.19.3.16 to_tbb_range() [1/3]

```
static auto trisycl::detail::to_tbb_range (
            const range< 1 > & r )  [inline], [static]
```

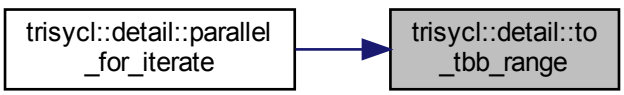
```
#include <include/triSYCL/parallelism/detail/parallelism_tbb.hpp>
```

Definition at line 34 of file parallelism_tbb.hpp.

```
00035 {
00036   return tbb::blocked_range<size_t>(0, r[0]);
00037 }
```

Referenced by trisycl::detail::parallel_for_iterate().

Here is the caller graph for this function:



9.19.3.17 to_tbb_range() [2/3]

```
static auto trisycl::detail::to_tbb_range (
            const range< 2 > & r )  [inline], [static]
```

#include <include/triSYCL/parallelism/detail/parallelism_tbb.hpp>

Definition at line 39 of file parallelism_tbb.hpp.

```
00040 {
00041   return tbb::blocked_range2d<size_t>(0, r[0], 0, r[1]);
00042 }
```

9.19.3.18 to_tbb_range() [3/3]

```
static auto trisycl::detail::to_tbb_range (
            const range< 3 > & r )  [inline], [static]
```

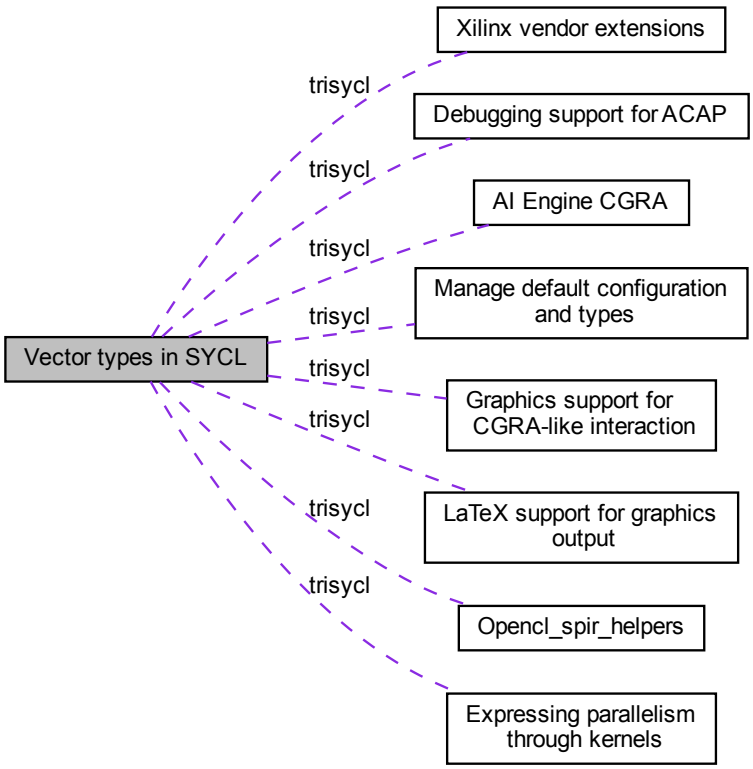
#include <include/triSYCL/parallelism/detail/parallelism_tbb.hpp>

Definition at line 44 of file parallelism_tbb.hpp.

```
00045 {
00046   return tbb::blocked_range3d<size_t>(0, r[0], 0, r[1], 0, r[2]);
00047 }
```

9.20 Vector types in SYCL

Collaboration diagram for Vector types in SYCL:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace std

Classes

- struct trisycl::elem

  Series of values to specify named swizzle indexes. More...
- class trisycl::vec< DataType, 1 >
- class trisycl::vec< DataType, 2 >
- class trisycl::vec< DataType, 3 >
- class trisycl::vec< DataType, 4 >
- class trisycl::vec< DataType, 8 >
- class trisycl::vec< DataType, 16 >

Macros

- #define TRISYCL_DECLARE_S(x)

  Accessors to access hex indexed elements of a vector.
- #define TRISYCL_GEN_SWIZ2(str, idx0, idx1)
- #define TRISYCL_GEN_SWIZ3(str, idx0, idx1, idx2)
- #define TRISYCL_GEN_SWIZ4(str, idx0, idx1, idx2, idx3)
- #define TRISYCL_DEFINE_VEC_TYPE_SIZE(type, size, actual_type) using type##size = vec<actual_type, size>;

  A macro to define type alias, such as for type=uchar, size=4 and actual_type=unsigned char, uchar4 is equivalent to vec<unsigned char, 4>
- #define TRISYCL_DEFINE_VEC_TYPE(type, actual_type)

  Declare the vector types of a type for all the sizes.

Typedefs

- template<typename DataType , int numElements>
  using trisycl::__swizzled_vec__ = vec< DataType, numElements >

Variables

- static constexpr int trisycl::elem::x = 0
- static constexpr int trisycl::elem::y = 1
- static constexpr int trisycl::elem::z = 2
- static constexpr int trisycl::elem::w = 3
- static constexpr int trisycl::elem::r = 0
- static constexpr int trisycl::elem::g = 1
- static constexpr int trisycl::elem::b = 2
- static constexpr int trisycl::elem::a = 3
- static constexpr int trisycl::elem::s0 = 0
- static constexpr int trisycl::elem::s1 = 1
- static constexpr int trisycl::elem::s2 = 2
- static constexpr int trisycl::elem::s3 = 3

- static constexpr int trisycl::elem::s4 = 4
- static constexpr int trisycl::elem::s5 = 5
- static constexpr int trisycl::elem::s6 = 6
- static constexpr int trisycl::elem::s7 = 7
- static constexpr int trisycl::elem::s8 = 8
- static constexpr int trisycl::elem::s9 = 9
- static constexpr int trisycl::elem::sA = 10
- static constexpr int trisycl::elem::sB = 11
- static constexpr int trisycl::elem::sC = 12
- static constexpr int trisycl::elem::sD = 13
- static constexpr int trisycl::elem::sE = 14
- static constexpr int trisycl::elem::sF = 15

9.20.1 Detailed Description

9.20.2 Class Documentation

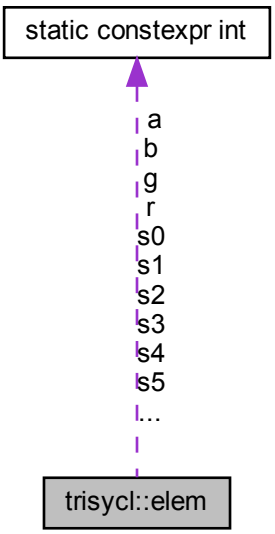
9.20.2.1 struct trisycl::elem

Series of values to specify named swizzle indexes.

Used when calling the swizzle member function template

Definition at line 27 of file vec.hpp.

Collaboration diagram for trisycl::elem:



Static Public Attributes

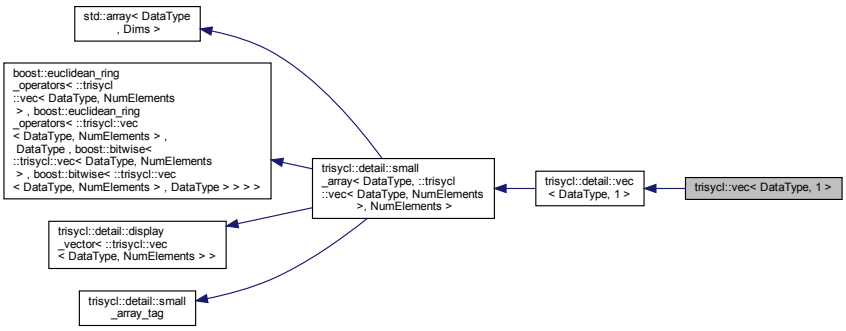
- static constexpr int x = 0
- static constexpr int y = 1
- static constexpr int z = 2
- static constexpr int w = 3
- static constexpr int r = 0
- static constexpr int g = 1
- static constexpr int b = 2
- static constexpr int a = 3
- static constexpr int s0 = 0
- static constexpr int s1 = 1
- static constexpr int s2 = 2
- static constexpr int s3 = 3
- static constexpr int s4 = 4
- static constexpr int s5 = 5
- static constexpr int s6 = 6
- static constexpr int s7 = 7
- static constexpr int s8 = 8
- static constexpr int s9 = 9
- static constexpr int sA = 10
- static constexpr int sB = 11
- static constexpr int sC = 12
- static constexpr int sD = 13
- static constexpr int sE = 14
- static constexpr int sF = 15

9.20.2.2 class trisycl::vec< DataType, 1 >

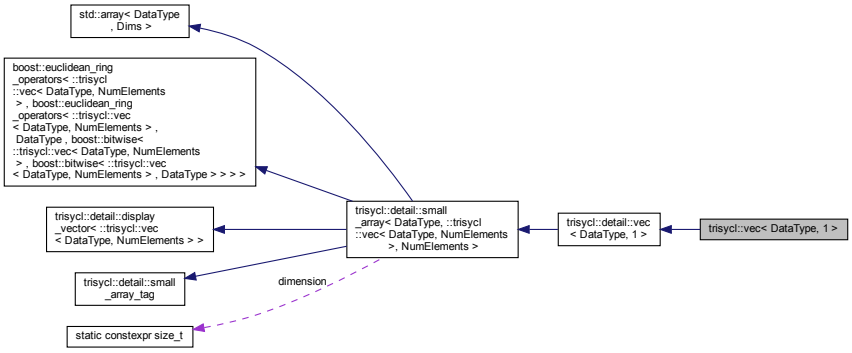
template<typename DataType>
class trisycl::vec< DataType, 1 >

Definition at line 99 of file vec.hpp.

Inheritance diagram for trisycl::vec< DataType, 1 >:



Collaboration diagram for trisycl::vec< DataType, 1 >:



Public Member Functions

- const DataType & x () const

  Accessors to the first element of a vector.
- DataType & x ()
- TRISYCL_DECLARE_S (0)
- operator DataType () const
- operator DataType ()

Private Types

- using base_vec = detail::vec< DataType, 1 >

Additional Inherited Members

9.20.2.2.1 Member Typedef Documentation

9.20.2.2.1.1 base_vec `template<typename DataType >`

```
using trisycl::vec< DataType, 1 >::base_vec = detail::vec<DataType, 1>  [private]
```

Definition at line 101 of file vec.hpp.

9.20.2.2.2 Member Function Documentation

9.20.2.2.2.1 operator DataType() `[1/2]` `template<typename DataType >`

```
trisycl::vec< DataType, 1 >::operator DataType ( )  [inline]
```

Definition at line 124 of file vec.hpp.

```
00124                         {
00125     return (*this)[0];
00126   }
```

9.20.2.2.2 operator DataType() [2/2] template<typename DataType >

trisycl::vec< DataType, 1 >::operator DataType ( ) const [inline]

Definition at line 120 of file vec.hpp.

```
00120                                {
00121     return (*this)[0];
00122   }
```

9.20.2.2.3 TRISYCL_DECLARE_S() template<typename DataType >

trisycl::vec< DataType, 1 >::TRISYCL_DECLARE_S (
0)

9.20.2.2.4 x() [1/2] template<typename DataType >

DataType & trisycl::vec< DataType, 1 >::x ( ) [inline]

Definition at line 114 of file vec.hpp.

```
00114                 {
00115     return (*this)[0];
00116   }
```

9.20.2.2.5 x() [2/2] template<typename DataType >

const DataType & trisycl::vec< DataType, 1 >::x ( ) const [inline]

Accessors to the first element of a vector.

Definition at line 110 of file vec.hpp.

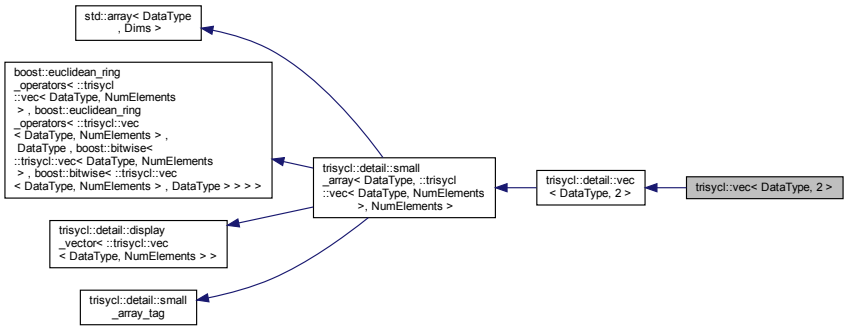
```
00110                                {
00111     return (*this)[0];
00112   }
```

9.20.2.3 class trisycl::vec< DataType, 2 >

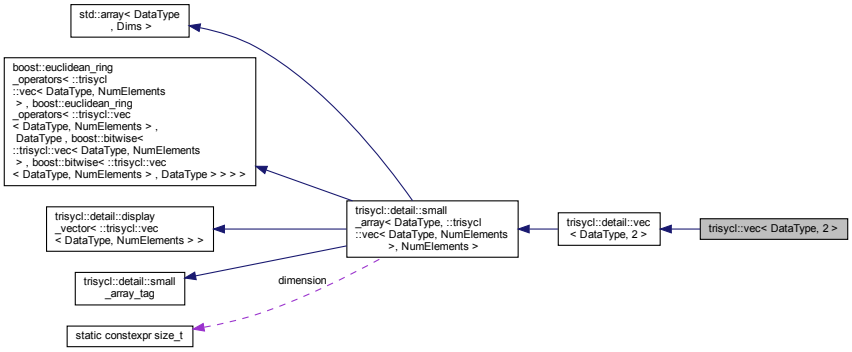
template<typename DataType>
class trisycl::vec< DataType, 2 >

Definition at line 130 of file vec.hpp.

Inheritance diagram for trisycl::vec< DataType, 2 >:



Collaboration diagram for trisycl::vec< DataType, 2 >:



Public Member Functions

- const DataType & x () const

  Accessors to the first element of a vector.
- DataType & x ()
- const DataType & y () const

  Accessors to the second element of a vector.
- DataType & y ()
- TRISYCL_DECLARE_S (0)
- TRISYCL_DECLARE_S (1)
- const __swizzled_vec__< DataType, 1 > lo () const
- __swizzled_vec__< DataType, 1 > lo ()
- const __swizzled_vec__< DataType, 1 > hi () const
- __swizzled_vec__< DataType, 1 > hi ()
- const __swizzled_vec__< DataType, 1 > odd () const
- __swizzled_vec__< DataType, 1 > odd ()
- const __swizzled_vec__< DataType, 1 > even () const
- __swizzled_vec__< DataType, 1 > even ()

Private Types

- using base_vec = detail::vec< DataType, 2 >

Additional Inherited Members

9.20.2.3.1 Member Typedef Documentation

9.20.2.3.1.1 base_vec `template<typename DataType >`

```
using trisycl::vec< DataType, 2 >::base_vec = detail::vec<DataType, 2>  [private]
```

Definition at line 132 of file vec.hpp.

9.20.2.3.2 Member Function Documentation

9.20.2.3.2.1 even() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 1 > trisycl::vec< DataType, 2 >::even ( ) [inline]

Definition at line 190 of file vec.hpp.

```
00190                                       {
00191     return base_vec::swizzle(elem::s0);
00192   }
```

References trisycl::elem::s0.

9.20.2.3.2.2 even() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 1 > trisycl::vec< DataType, 2 >::even ( ) const [inline]

Definition at line 186 of file vec.hpp.

```
00186                                             {
00187     return base_vec::swizzle(elem::s0);
00188   }
```

References trisycl::elem::s0.

9.20.2.3.2.3 hi() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 1 > trisycl::vec< DataType, 2 >::hi ( ) [inline]

Definition at line 174 of file vec.hpp.

```
00174                                       {
00175     return base_vec::swizzle(elem::s1);
00176   }
```

References trisycl::elem::s1.

9.20.2.3.2.4 hi() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 1 > trisycl::vec< DataType, 2 >::hi ( ) const [inline]

Definition at line 170 of file vec.hpp.

```
00170                                             {
00171     return base_vec::swizzle(elem::s1);
00172   }
```

References trisycl::elem::s1.

9.20.2.3.2.5 lo() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 1 > trisycl::vec< DataType, 2 >::lo ( ) [inline]

Definition at line 166 of file vec.hpp.

```
00166                                       {
00167     return base_vec::swizzle(elem::s0);
00168   }
```

References trisycl::elem::s0.

9.20.2.3.2.6 lo() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 1 > trisycl::vec< DataType, 2 >::lo ( ) const [inline]

Definition at line 162 of file vec.hpp.

```
00162                                                {
00163     return base_vec::swizzle(elem::s0);
00164   }
```

References trisycl::elem::s0.

9.20.2.3.2.7 odd() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 1 > trisycl::vec< DataType, 2 >::odd ( ) [inline]

Definition at line 182 of file vec.hpp.

```
00182                                          {
00183     return base_vec::swizzle(elem::s1);
00184   }
```

References trisycl::elem::s1.

9.20.2.3.2.8 odd() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 1 > trisycl::vec< DataType, 2 >::odd ( ) const [inline]

Definition at line 178 of file vec.hpp.

```
00178                                                {
00179     return base_vec::swizzle(elem::s1);
00180   }
```

References trisycl::elem::s1.

9.20.2.3.2.9 TRISYCL_DECLARE_S() [1/2] template<typename DataType >

trisycl::vec< DataType, 2 >::TRISYCL_DECLARE_S (
0)

9.20.2.3.2.10 TRISYCL_DECLARE_S() [2/2] template<typename DataType >

trisycl::vec< DataType, 2 >::TRISYCL_DECLARE_S (
1)

9.20.2.3.2.11 x() [1/2] template<typename DataType >

DataType & trisycl::vec< DataType, 2 >::x ( ) [inline]

Definition at line 145 of file vec.hpp.

```
00145                {
00146     return (*this)[0];
00147   }
```

9.20.2.3.2.12 x() [2/2] template<typename DataType >

const DataType & trisycl::vec< DataType, 2 >::x ( ) const [inline]

Accessors to the first element of a vector.

Definition at line 141 of file vec.hpp.

```
00141                                {
00142     return (*this)[0];
00143   }
```

9.20.2.3.2.13 y() [1/2] template<typename DataType >

DataType & trisycl::vec< DataType, 2 >::y ( ) [inline]

Definition at line 155 of file vec.hpp.

```
00155                {
00156     return (*this)[1];
00157   }
```

9.20.2.3.2.14 y() [2/2] template<typename DataType >

const DataType & trisycl::vec< DataType, 2 >::y ( ) const [inline]

Accessors to the second element of a vector.

Definition at line 151 of file vec.hpp.

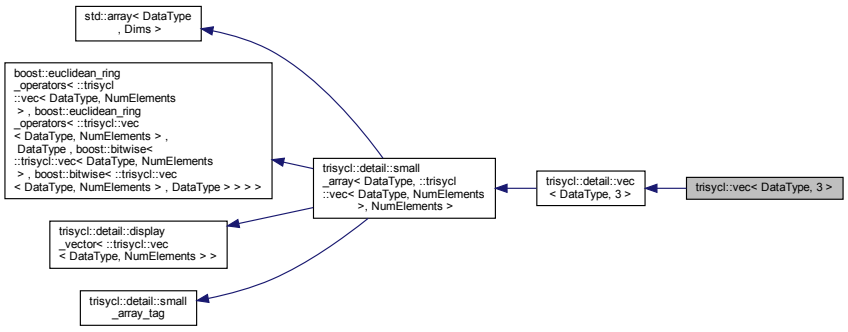
```
00151                                {
00152     return (*this)[1];
00153   }
```

9.20.2.4 class trisycl::vec< DataType, 3 >

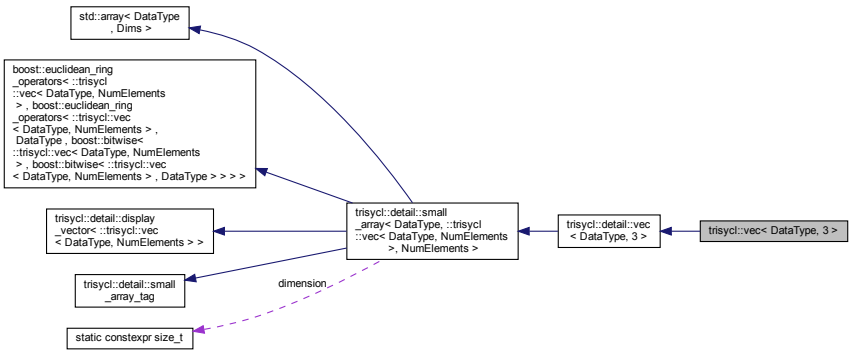
template<typename DataType>
class trisycl::vec< DataType, 3 >

Definition at line 197 of file vec.hpp.

Inheritance diagram for trisycl::vec< DataType, 3 >:



Collaboration diagram for trisycl::vec< DataType, 3 >:



Public Member Functions

- const DataType & x () const

  Accessors to the first element of a vector.
- DataType & x ()
- const DataType & y () const

  Accessors to the second element of a vector.
- DataType & y ()
- const DataType & z () const

  Accessors to the third element of a vector.
- DataType & z ()
- TRISYCL_DECLARE_S (0)
- TRISYCL_DECLARE_S (1)
- TRISYCL_DECLARE_S (2)
- const __swizzled_vec__< DataType, 2 > lo () const
- __swizzled_vec__< DataType, 2 > lo ()
- const __swizzled_vec__< DataType, 2 > hi () const
- __swizzled_vec__< DataType, 2 > hi ()
- const __swizzled_vec__< DataType, 2 > odd () const
- __swizzled_vec__< DataType, 2 > odd ()
- const __swizzled_vec__< DataType, 2 > even () const
- __swizzled_vec__< DataType, 2 > even ()

Private Types

- using base_vec = detail::vec< DataType, 3 >

Additional Inherited Members

9.20.2.4.1 Member Typedef Documentation

9.20.2.4.1.1 base_vec `template<typename DataType >`

```
using trisycl::vec< DataType, 3 >::base_vec = detail::vec<DataType, 3>  [private]
```

Definition at line 199 of file vec.hpp.

9.20.2.4.2 Member Function Documentation

9.20.2.4.2.1 even() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 3 >::even ( ) [inline]

Definition at line 268 of file vec.hpp.

```
00268                                       {
00269     return base_vec::swizzle(elem::s0, elem::s2);
00270   }
```

References trisycl::elem::s0, and trisycl::elem::s2.

9.20.2.4.2.2 even() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 3 >::even ( ) const [inline]

Definition at line 264 of file vec.hpp.

```
00264                                             {
00265     return base_vec::swizzle(elem::s0, elem::s2);
00266   }
```

References trisycl::elem::s0, and trisycl::elem::s2.

9.20.2.4.2.3 hi() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 3 >::hi ( ) [inline]

Definition at line 252 of file vec.hpp.

```
00252                                      {
00253     return base_vec::swizzle(elem::s2, elem::s2);
00254   }
```

References trisycl::elem::s2.

9.20.2.4.2.4 hi() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 3 >::hi ( ) const [inline]

Definition at line 248 of file vec.hpp.

```
00248                                            {
00249     return base_vec::swizzle(elem::s2, elem::s2);
00250   }
```

References trisycl::elem::s2.

9.20.2.4.2.5 lo() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 3 >::lo ( ) [inline]

Definition at line 244 of file vec.hpp.

```
00244                                       {
00245     return base_vec::swizzle(elem::s0, elem::s1);
00246   }
```

References trisycl::elem::s0, and trisycl::elem::s1.

9.20.2.4.2.6 lo() [2/2] template<typename DataType >

```
const __swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 3 >::lo ( ) const  [inline]
```

Definition at line 240 of file vec.hpp.

```
00240                                                {
00241     return base_vec::swizzle(elem::s0, elem::s1);
00242   }
```

References trisycl::elem::s0, and trisycl::elem::s1.

9.20.2.4.2.7 odd() [1/2] template<typename DataType >

```
__swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 3 >::odd ( )  [inline]
```

Definition at line 260 of file vec.hpp.

```
00260                                      {
00261     return base_vec::swizzle(elem::s1, elem::s1);
00262   }
```

References trisycl::elem::s1.

9.20.2.4.2.8 odd() [2/2] template<typename DataType >

```
const __swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 3 >::odd ( ) const  [inline]
```

Definition at line 256 of file vec.hpp.

```
00256                                                {
00257     return base_vec::swizzle(elem::s1, elem::s1);
00258   }
```

References trisycl::elem::s1.

9.20.2.4.2.9 TRISYCL_DECLARE_S() [1/3] template<typename DataType >

```
trisycl::vec< DataType, 3 >::TRISYCL_DECLARE_S (
            0  )
```

9.20.2.4.2.10 TRISYCL_DECLARE_S() [2/3] template<typename DataType >

```
trisycl::vec< DataType, 3 >::TRISYCL_DECLARE_S (
            1  )
```

9.20.2.4.2.11 TRISYCL_DECLARE_S() [3/3] template<typename DataType >

```
trisycl::vec< DataType, 3 >::TRISYCL_DECLARE_S (
            2  )
```

9.20.2.4.2.12 x() [1/2] template<typename DataType >

DataType & trisycl::vec< DataType, 3 >::x ( ) [inline]

Definition at line 212 of file vec.hpp.

```
00212                 {
00213     return (*this)[0];
00214   }
```

9.20.2.4.2.13 x() [2/2] template<typename DataType >

const DataType & trisycl::vec< DataType, 3 >::x ( ) const [inline]

Accessors to the first element of a vector.

Definition at line 208 of file vec.hpp.

```
00208                                 {
00209     return (*this)[0];
00210   }
```

9.20.2.4.2.14 y() [1/2] template<typename DataType >

DataType & trisycl::vec< DataType, 3 >::y ( ) [inline]

Definition at line 222 of file vec.hpp.

```
00222                 {
00223     return (*this)[1];
00224   }
```

9.20.2.4.2.15 y() [2/2] template<typename DataType >

const DataType & trisycl::vec< DataType, 3 >::y ( ) const [inline]

Accessors to the second element of a vector.

Definition at line 218 of file vec.hpp.

```
00218                                 {
00219     return (*this)[1];
00220   }
```

9.20.2.4.2.16 z() [1/2] template<typename DataType >

DataType & trisycl::vec< DataType, 3 >::z ( ) [inline]

Definition at line 232 of file vec.hpp.

```
00232                 {
00233     return (*this)[2];
00234   }
```

9.20.2.4.2.17 z() [2/2] template<typename DataType >

```
const DataType & trisycl::vec< DataType, 3 >::z ( ) const  [inline]
```

Accessors to the third element of a vector.

Definition at line 228 of file vec.hpp.

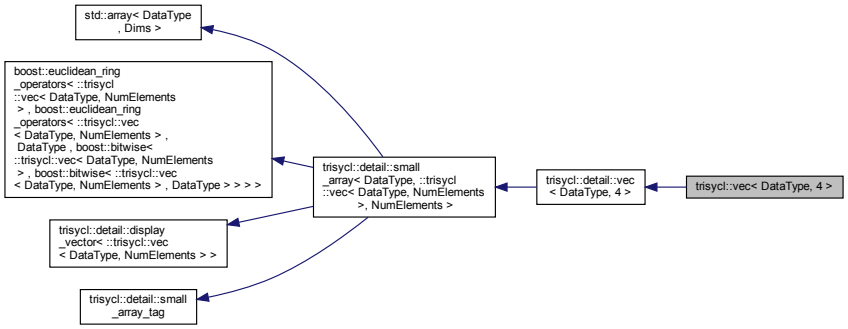
```
00228                                 {
00229       return (*this)[2];
00230     }
```

9.20.2.5 class trisycl::vec< DataType, 4 >

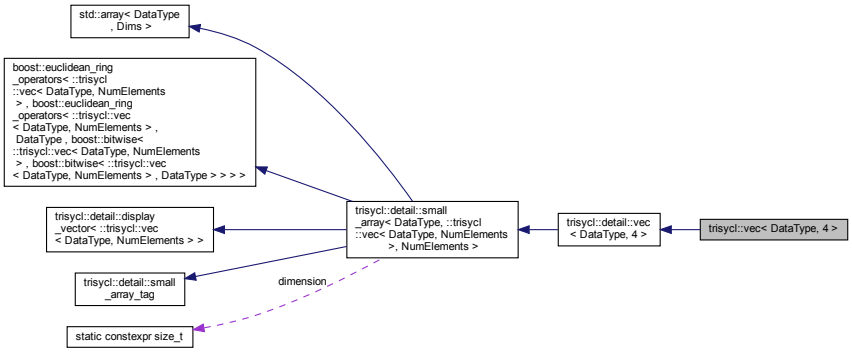
template<typename DataType>
class trisycl::vec< DataType, 4 >

Definition at line 275 of file vec.hpp.

Inheritance diagram for trisycl::vec< DataType, 4 >:



Collaboration diagram for trisycl::vec< DataType, 4 >:



Public Member Functions

- const DataType & x () const

  Accessors to the first element of a vector.
- DataType & x ()
- const DataType & y () const

  Accessors to the second element of a vector.
- DataType & y ()
- const DataType & z () const

  Accessors to the third element of a vector.
- DataType & z ()
- const DataType & w () const

  Accessors to the fourth element of a vector.
- DataType & w ()
- const DataType & r () const

  Accessors to the first element of a vector.
- DataType & r ()
- const DataType & g () const

  Accessors to the second element of a vector.
- DataType & g ()
- const DataType & b () const

  Accessors to the third element of a vector.
- DataType & b ()
- const DataType & a () const

  Accessors to the third element of a vector.
- DataType & a ()
- TRISYCL_DECLARE_S (0)
- TRISYCL_DECLARE_S (1)
- TRISYCL_DECLARE_S (2)
- TRISYCL_DECLARE_S (3)
- const __swizzled_vec__< DataType, 2 > lo () const
- __swizzled_vec__< DataType, 2 > lo ()
- const __swizzled_vec__< DataType, 2 > hi () const
- __swizzled_vec__< DataType, 2 > hi ()
- const __swizzled_vec__< DataType, 2 > odd () const
- __swizzled_vec__< DataType, 2 > odd ()
- const __swizzled_vec__< DataType, 2 > even () const
- __swizzled_vec__< DataType, 2 > even ()

Private Types

- using base_vec = detail::vec< DataType, 4 >

Additional Inherited Members

9.20.2.5.1 Member Typedef Documentation

9.20.2.5.1.1 base_vec `template<typename DataType >`

```
using trisycl::vec< DataType, 4 >::base_vec = detail::vec<DataType, 4>  [private]
```

Definition at line 277 of file vec.hpp.

9.20.2.5.2 Member Function Documentation

9.20.2.5.2.1 a() [1/2] template<typename DataType >

```
DataType & trisycl::vec< DataType, 4 >::a ( )  [inline]
```

Definition at line 360 of file vec.hpp.

```
00360                 {
00361     return (*this)[3];
00362   }
```

9.20.2.5.2.2 a() [2/2] template<typename DataType >

```
const DataType & trisycl::vec< DataType, 4 >::a ( ) const  [inline]
```

Accessors to the third element of a vector.

Definition at line 356 of file vec.hpp.

```
00356                                 {
00357       return (*this)[3];
00358   }
```

9.20.2.5.2.3 b() [1/2] template<typename DataType >

```
DataType & trisycl::vec< DataType, 4 >::b ( )  [inline]
```

Definition at line 350 of file vec.hpp.

```
00350                 {
00351     return (*this)[2];
00352   }
```

9.20.2.5.2.4 b() [2/2] template<typename DataType >

```
const DataType & trisycl::vec< DataType, 4 >::b ( ) const  [inline]
```

Accessors to the third element of a vector.

Definition at line 346 of file vec.hpp.

```
00346                                 {
00347       return (*this)[2];
00348   }
```

9.20.2.5.2.5 even() [1/2] template<typename DataType >

```
__swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 4 >::even ( )  [inline]
```

Definition at line 397 of file vec.hpp.

```
00397                                       {
00398     return base_vec::swizzle(elem::s0, elem::s2);
00399   }
```

References trisycl::elem::s0, and trisycl::elem::s2.

9.20.2.5.2.6 even() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 4 >::even ( ) const [inline]

Definition at line 393 of file vec.hpp.

```
00393                                                {
00394     return base_vec::swizzle(elem::s0, elem::s2);
00395   }
```

References trisycl::elem::s0, and trisycl::elem::s2.

9.20.2.5.2.7 g() [1/2] template<typename DataType >

DataType & trisycl::vec< DataType, 4 >::g ( ) [inline]

Definition at line 340 of file vec.hpp.

```
00340                {
00341     return (*this)[1];
00342   }
```

9.20.2.5.2.8 g() [2/2] template<typename DataType >

const DataType & trisycl::vec< DataType, 4 >::g ( ) const [inline]

Accessors to the second element of a vector.

Definition at line 336 of file vec.hpp.

```
00336                          {
00337       return (*this)[1];
00338   }
```

9.20.2.5.2.9 hi() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 4 >::hi ( ) [inline]

Definition at line 381 of file vec.hpp.

```
00381                                    {
00382     return base_vec::swizzle(elem::s2, elem::s3);
00383   }
```

References trisycl::elem::s2, and trisycl::elem::s3.

9.20.2.5.2.10 hi() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 4 >::hi ( ) const [inline]

Definition at line 377 of file vec.hpp.

```
00377                                              {
00378     return base_vec::swizzle(elem::s2, elem::s3);
00379   }
```

References trisycl::elem::s2, and trisycl::elem::s3.

9.20.2.5.2.11 lo() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 4 >::lo ( ) [inline]

Definition at line 373 of file vec.hpp.

```
00373                                       {
00374     return base_vec::swizzle(elem::s0, elem::s1);
00375   }
```

References trisycl::elem::s0, and trisycl::elem::s1.

9.20.2.5.2.12 lo() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 4 >::lo ( ) const [inline]

Definition at line 369 of file vec.hpp.

```
00369                                                 {
00370     return base_vec::swizzle(elem::s0, elem::s1);
00371   }
```

References trisycl::elem::s0, and trisycl::elem::s1.

9.20.2.5.2.13 odd() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 4 >::odd ( ) [inline]

Definition at line 389 of file vec.hpp.

```
00389                                       {
00390     return base_vec::swizzle(elem::s1, elem::s3);
00391   }
```

References trisycl::elem::s1, and trisycl::elem::s3.

9.20.2.5.2.14 odd() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 2 > trisycl::vec< DataType, 4 >::odd ( ) const [inline]

Definition at line 385 of file vec.hpp.

```
00385                                                 {
00386     return base_vec::swizzle(elem::s1, elem::s3);
00387   }
```

References trisycl::elem::s1, and trisycl::elem::s3.

9.20.2.5.2.15 r() [1/2] template<typename DataType >

DataType & trisycl::vec< DataType, 4 >::r ( ) [inline]

Definition at line 330 of file vec.hpp.

```
00330                 {
00331     return (*this)[0];
00332   }
```

9.20.2.5.2.16 r() [2/2] template<typename DataType >

```
const DataType & trisycl::vec< DataType, 4 >::r ( ) const  [inline]
```

Accessors to the first element of a vector.

Definition at line 326 of file vec.hpp.

```
00326                                 {
00327       return (*this)[0];
00328     }
```

9.20.2.5.2.17 TRISYCL_DECLARE_S() [1/4] template<typename DataType >

```
trisycl::vec< DataType, 4 >::TRISYCL_DECLARE_S (
            0  )
```

9.20.2.5.2.18 TRISYCL_DECLARE_S() [2/4] template<typename DataType >

```
trisycl::vec< DataType, 4 >::TRISYCL_DECLARE_S (
            1  )
```

9.20.2.5.2.19 TRISYCL_DECLARE_S() [3/4] template<typename DataType >

```
trisycl::vec< DataType, 4 >::TRISYCL_DECLARE_S (
            2  )
```

9.20.2.5.2.20 TRISYCL_DECLARE_S() [4/4] template<typename DataType >

```
trisycl::vec< DataType, 4 >::TRISYCL_DECLARE_S (
            3  )
```

9.20.2.5.2.21 w() [1/2] template<typename DataType >

```
DataType & trisycl::vec< DataType, 4 >::w ( )  [inline]
```

Definition at line 320 of file vec.hpp.

```
00320                 {
00321     return (*this)[3];
00322   }
```

9.20.2.5.2.22 w() [2/2] template<typename DataType >

```
const DataType & trisycl::vec< DataType, 4 >::w ( ) const  [inline]
```

Accessors to the fourth element of a vector.

Definition at line 316 of file vec.hpp.

```
00316                                 {
00317       return (*this)[3];
00318     }
```

9.20.2.5.2.23 x() [1/2] template<typename DataType >

DataType & trisycl::vec< DataType, 4 >::x ( ) [inline]

Definition at line 290 of file vec.hpp.

```
00290                 {
00291     return (*this)[0];
00292   }
```

9.20.2.5.2.24 x() [2/2] template<typename DataType >

const DataType & trisycl::vec< DataType, 4 >::x ( ) const [inline]

Accessors to the first element of a vector.

Definition at line 286 of file vec.hpp.

```
00286                       {
00287     return (*this)[0];
00288   }
```

9.20.2.5.2.25 y() [1/2] template<typename DataType >

DataType & trisycl::vec< DataType, 4 >::y ( ) [inline]

Definition at line 300 of file vec.hpp.

```
00300                 {
00301     return (*this)[1];
00302   }
```

9.20.2.5.2.26 y() [2/2] template<typename DataType >

const DataType & trisycl::vec< DataType, 4 >::y ( ) const [inline]

Accessors to the second element of a vector.

Definition at line 296 of file vec.hpp.

```
00296                       {
00297     return (*this)[1];
00298   }
```

9.20.2.5.2.27 z() [1/2] template<typename DataType >

DataType & trisycl::vec< DataType, 4 >::z ( ) [inline]

Definition at line 310 of file vec.hpp.

```
00310                 {
00311     return (*this)[2];
00312   }
```

9.20.2.5.2.28 z() [2/2] template<typename DataType >

```
const DataType & trisycl::vec< DataType, 4 >::z ( ) const  [inline]
```

Accessors to the third element of a vector.

Definition at line 306 of file vec.hpp.

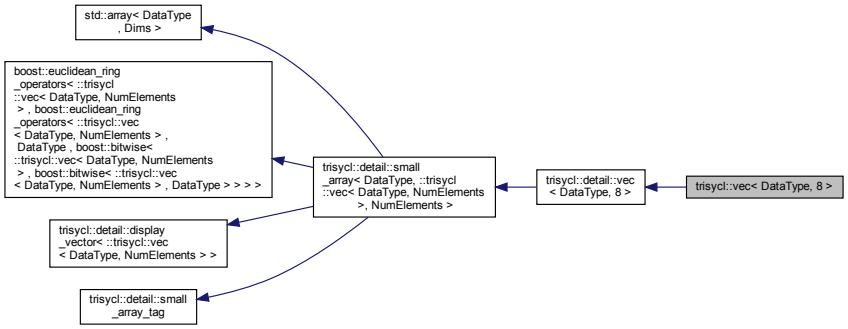
```
00306                                {
00307       return (*this)[2];
00308   }
```

9.20.2.6 class trisycl::vec< DataType, 8 >

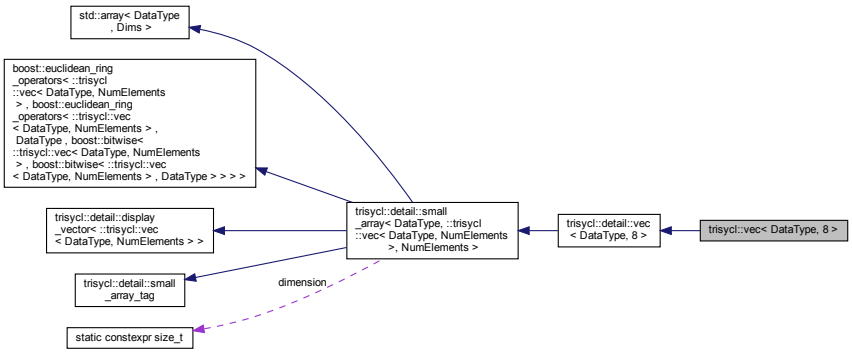
template<typename DataType>
class trisycl::vec< DataType, 8 >

Definition at line 409 of file vec.hpp.

Inheritance diagram for trisycl::vec< DataType, 8 >:



Collaboration diagram for trisycl::vec< DataType, 8 >:



Public Member Functions

- TRISYCL_DECLARE_S (0)
- TRISYCL_DECLARE_S (1)
- TRISYCL_DECLARE_S (2)
- TRISYCL_DECLARE_S (3)
- TRISYCL_DECLARE_S (4)
- TRISYCL_DECLARE_S (5)
- TRISYCL_DECLARE_S (6)
- TRISYCL_DECLARE_S (7)
- TRISYCL_DECLARE_S (8)
- const __swizzled_vec__< DataType, 4 > lo () const
- __swizzled_vec__< DataType, 4 > lo ()
- const __swizzled_vec__< DataType, 4 > hi () const
- __swizzled_vec__< DataType, 4 > hi ()
- const __swizzled_vec__< DataType, 4 > odd () const
- __swizzled_vec__< DataType, 4 > odd ()
- const __swizzled_vec__< DataType, 4 > even () const
- __swizzled_vec__< DataType, 4 > even ()

Private Types

- using base_vec = detail::vec< DataType, 8 >

Additional Inherited Members

9.20.2.6.1 Member Typedef Documentation

9.20.2.6.1.1 base_vec `template<typename DataType >`

```
using trisycl::vec< DataType, 8 >::base_vec = detail::vec<DataType, 8>  [private]
```

Definition at line 411 of file vec.hpp.

9.20.2.6.2 Member Function Documentation

9.20.2.6.2.1 even() [1/2] `template<typename DataType >`

```
__swizzled_vec__< DataType, 4 > trisycl::vec< DataType, 8 >::even ( )  [inline]
```

Definition at line 456 of file vec.hpp.

```
00456                                   {
00457     return base_vec::swizzle(elem::s0, elem::s2, elem::s4, elem::s6);
00458   }
```

References trisycl::elem::s0, trisycl::elem::s2, trisycl::elem::s4, and trisycl::elem::s6.

9.20.2.6.2.2 even() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 4 > trisycl::vec< DataType, 8 >::even ( ) const [inline]

Definition at line 452 of file vec.hpp.

```
00452                                                {
00453     return base_vec::swizzle(elem::s0, elem::s2, elem::s4, elem::s6);
00454   }
```

References trisycl::elem::s0, trisycl::elem::s2, trisycl::elem::s4, and trisycl::elem::s6.

9.20.2.6.2.3 hi() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 4 > trisycl::vec< DataType, 8 >::hi ( ) [inline]

Definition at line 440 of file vec.hpp.

```
00440                                       {
00441     return base_vec::swizzle(elem::s4, elem::s5, elem::s6, elem::s7);
00442   }
```

References trisycl::elem::s4, trisycl::elem::s5, trisycl::elem::s6, and trisycl::elem::s7.

9.20.2.6.2.4 hi() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 4 > trisycl::vec< DataType, 8 >::hi ( ) const [inline]

Definition at line 436 of file vec.hpp.

```
00436                                                {
00437     return base_vec::swizzle(elem::s4, elem::s5, elem::s6, elem::s7);
00438   }
```

References trisycl::elem::s4, trisycl::elem::s5, trisycl::elem::s6, and trisycl::elem::s7.

9.20.2.6.2.5 lo() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 4 > trisycl::vec< DataType, 8 >::lo ( ) [inline]

Definition at line 432 of file vec.hpp.

```
00432                                       {
00433     return base_vec::swizzle(elem::s0, elem::s1, elem::s2, elem::s3);
00434   }
```

References trisycl::elem::s0, trisycl::elem::s1, trisycl::elem::s2, and trisycl::elem::s3.

9.20.2.6.2.6 lo() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 4 > trisycl::vec< DataType, 8 >::lo ( ) const [inline]

Definition at line 428 of file vec.hpp.

```
00428                                                {
00429     return base_vec::swizzle(elem::s0, elem::s1, elem::s2, elem::s3);
00430   }
```

References trisycl::elem::s0, trisycl::elem::s1, trisycl::elem::s2, and trisycl::elem::s3.

9.20.2.6.2.7 odd() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 4 > trisycl::vec< DataType, 8 >::odd ( ) [inline]

Definition at line 448 of file vec.hpp.

```
00448                                       {
00449     return base_vec::swizzle(elem::s1, elem::s3, elem::s5, elem::s7);
00450   }
```

References trisycl::elem::s1, trisycl::elem::s3, trisycl::elem::s5, and trisycl::elem::s7.

9.20.2.6.2.8 odd() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 4 > trisycl::vec< DataType, 8 >::odd ( ) const [inline]

Definition at line 444 of file vec.hpp.

```
00444                                             {
00445     return base_vec::swizzle(elem::s1, elem::s3, elem::s5, elem::s7);
00446   }
```

References trisycl::elem::s1, trisycl::elem::s3, trisycl::elem::s5, and trisycl::elem::s7.

9.20.2.6.2.9 TRISYCL_DECLARE_S() [1/9] template<typename DataType >

trisycl::vec< DataType, 8 >::TRISYCL_DECLARE_S (
0)

9.20.2.6.2.10 TRISYCL_DECLARE_S() [2/9] template<typename DataType >

trisycl::vec< DataType, 8 >::TRISYCL_DECLARE_S (
1)

9.20.2.6.2.11 TRISYCL_DECLARE_S() [3/9] template<typename DataType >

trisycl::vec< DataType, 8 >::TRISYCL_DECLARE_S (
2)

9.20.2.6.2.12 TRISYCL_DECLARE_S() [4/9] template<typename DataType >

trisycl::vec< DataType, 8 >::TRISYCL_DECLARE_S (
3)

9.20.2.6.2.13 TRISYCL_DECLARE_S() [5/9] template<typename DataType >

trisycl::vec< DataType, 8 >::TRISYCL_DECLARE_S (
4)

9.20.2.6.2.14 TRISYCL_DECLARE_S() [6/9] template<typename DataType >

```
trisycl::vec< DataType, 8 >::TRISYCL_DECLARE_S (
            5  )
```

9.20.2.6.2.15 TRISYCL_DECLARE_S() [7/9] template<typename DataType >

```
trisycl::vec< DataType, 8 >::TRISYCL_DECLARE_S (
            6  )
```

9.20.2.6.2.16 TRISYCL_DECLARE_S() [8/9] template<typename DataType >

```
trisycl::vec< DataType, 8 >::TRISYCL_DECLARE_S (
            7  )
```

9.20.2.6.2.17 TRISYCL_DECLARE_S() [9/9] template<typename DataType >

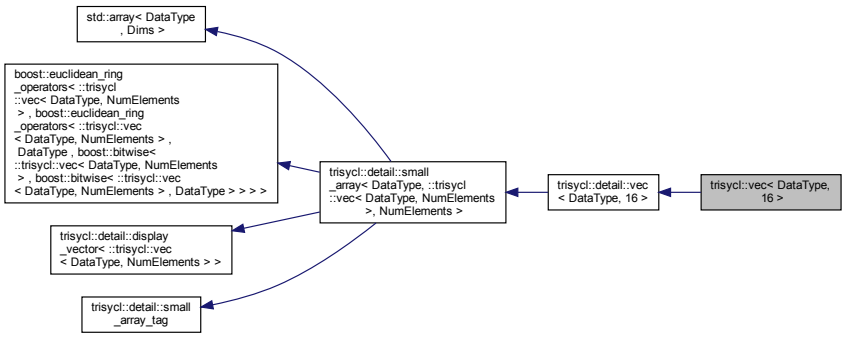
```
trisycl::vec< DataType, 8 >::TRISYCL_DECLARE_S (
            8  )
```

9.20.2.7 class trisycl::vec< DataType, 16 >

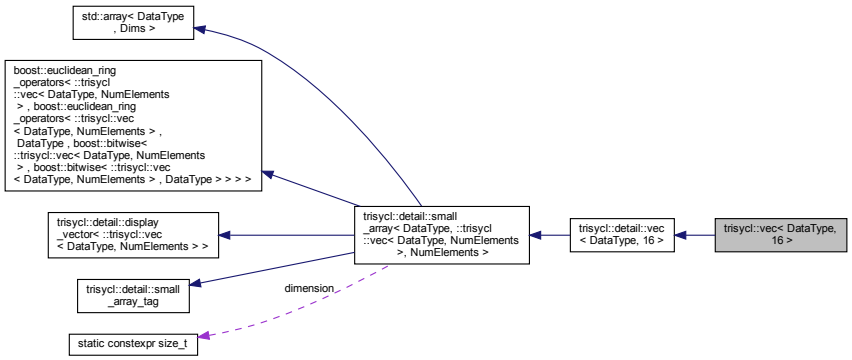
template<typename DataType>
class trisycl::vec< DataType, 16 >

Definition at line 463 of file vec.hpp.

Inheritance diagram for trisycl::vec< DataType, 16 >:



Collaboration diagram for trisycl::vec< DataType, 16 >:



Public Member Functions

- TRISYCL_DECLARE_S (0)
- TRISYCL_DECLARE_S (1)
- TRISYCL_DECLARE_S (2)
- TRISYCL_DECLARE_S (3)
- TRISYCL_DECLARE_S (4)
- TRISYCL_DECLARE_S (5)
- TRISYCL_DECLARE_S (6)
- TRISYCL_DECLARE_S (7)
- TRISYCL_DECLARE_S (8)
- TRISYCL_DECLARE_S (9)
- TRISYCL_DECLARE_S (A)
- TRISYCL_DECLARE_S (B)
- TRISYCL_DECLARE_S (C)
- TRISYCL_DECLARE_S (D)
- TRISYCL_DECLARE_S (E)
- TRISYCL_DECLARE_S (F)
- const __swizzled_vec__< DataType, 8 > lo () const
- __swizzled_vec__< DataType, 8 > lo ()
- const __swizzled_vec__< DataType, 8 > hi () const
- __swizzled_vec__< DataType, 8 > hi ()
- const __swizzled_vec__< DataType, 8 > odd () const
- __swizzled_vec__< DataType, 8 > odd ()
- const __swizzled_vec__< DataType, 8 > even () const
- __swizzled_vec__< DataType, 8 > even ()

Private Types

- using base_vec = detail::vec< DataType, 16 >

Additional Inherited Members

9.20.2.7.1 Member Typedef Documentation

9.20.2.7.1.1 base_vec template<typename DataType >

using trisycl::vec< DataType, 16 >::base_vec = detail::vec<DataType, 16> [private]

Definition at line 465 of file vec.hpp.

9.20.2.7.2 Member Function Documentation

9.20.2.7.2.1 even() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 8 > trisycl::vec< DataType, 16 >::even ( ) [inline]

Definition at line 524 of file vec.hpp.

```
00524                                        {
00525     return base_vec::swizzle(elem::s0, elem::s2, elem::s4, elem::s6, elem::s8,
00526                              elem::sA, elem::sC, elem::sE);
00527   }
```

References trisycl::elem::s0, trisycl::elem::s2, trisycl::elem::s4, trisycl::elem::s6, trisycl::elem::s8, trisycl::elem::sA, trisycl::elem::sC, and trisycl::elem::sE.

9.20.2.7.2.2 even() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 8 > trisycl::vec< DataType, 16 >::even ( ) const [inline]

Definition at line 519 of file vec.hpp.

```
00519                                              {
00520     return base_vec::swizzle(elem::s0, elem::s2, elem::s4, elem::s6, elem::s8,
00521                              elem::sA, elem::sC, elem::sE);
00522   }
```

References trisycl::elem::s0, trisycl::elem::s2, trisycl::elem::s4, trisycl::elem::s6, trisycl::elem::s8, trisycl::elem::sA, trisycl::elem::sC, and trisycl::elem::sE.

9.20.2.7.2.3 hi() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 8 > trisycl::vec< DataType, 16 >::hi ( ) [inline]

Definition at line 504 of file vec.hpp.

```
00504                                       {
00505     return base_vec::swizzle(elem::s8, elem::s9, elem::sA, elem::sB, elem::sC,
00506                              elem::sD, elem::sE, elem::sF);
00507   }
```

References trisycl::elem::s8, trisycl::elem::s9, trisycl::elem::sA, trisycl::elem::sB, trisycl::elem::sC, trisycl::elem::sD, trisycl::elem::sE, and trisycl::elem::sF.

9.20.2.7.2.4 hi() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 8 > trisycl::vec< DataType, 16 >::hi ( ) const [inline]

Definition at line 499 of file vec.hpp.

```
00499                                             {
00500     return base_vec::swizzle(elem::s8, elem::s9, elem::sA, elem::sB, elem::sC,
00501                              elem::sD, elem::sE, elem::sF);
00502   }
```

References trisycl::elem::s8, trisycl::elem::s9, trisycl::elem::sA, trisycl::elem::sB, trisycl::elem::sC, trisycl::elem::sD, trisycl::elem::sE, and trisycl::elem::sF.

9.20.2.7.2.5 lo() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 8 > trisycl::vec< DataType, 16 >::lo ( ) [inline]

Definition at line 494 of file vec.hpp.

```
00494                                       {
00495     return base_vec::swizzle(elem::s0, elem::s1, elem::s2, elem::s3, elem::s4,
00496                              elem::s5, elem::s6, elem::s7);
00497   }
```

References trisycl::elem::s0, trisycl::elem::s1, trisycl::elem::s2, trisycl::elem::s3, trisycl::elem::s4, trisycl::elem::s5, trisycl::elem::s6, and trisycl::elem::s7.

9.20.2.7.2.6 lo() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 8 > trisycl::vec< DataType, 16 >::lo ( ) const [inline]

Definition at line 489 of file vec.hpp.

```
00489                                             {
00490     return base_vec::swizzle(elem::s0, elem::s1, elem::s2, elem::s3, elem::s4,
00491                              elem::s5, elem::s6, elem::s7);
00492   }
```

References trisycl::elem::s0, trisycl::elem::s1, trisycl::elem::s2, trisycl::elem::s3, trisycl::elem::s4, trisycl::elem::s5, trisycl::elem::s6, and trisycl::elem::s7.

9.20.2.7.2.7 odd() [1/2] template<typename DataType >

__swizzled_vec__< DataType, 8 > trisycl::vec< DataType, 16 >::odd ( ) [inline]

Definition at line 514 of file vec.hpp.

```
00514                                       {
00515     return base_vec::swizzle(elem::s1, elem::s3, elem::s5, elem::s7, elem::s9,
00516                              elem::sB, elem::sD, elem::sF);
00517   }
```

References trisycl::elem::s1, trisycl::elem::s3, trisycl::elem::s5, trisycl::elem::s7, trisycl::elem::s9, trisycl::elem::sB, trisycl::elem::sD, and trisycl::elem::sF.

9.20.2.7.2.8 odd() [2/2] template<typename DataType >

const __swizzled_vec__< DataType, 8 > trisycl::vec< DataType, 16 >::odd ( ) const [inline]

Definition at line 509 of file vec.hpp.

```
00509                                             {
00510     return base_vec::swizzle(elem::s1, elem::s3, elem::s5, elem::s7, elem::s9,
00511                              elem::sB, elem::sD, elem::sF);
00512   }
```

References trisycl::elem::s1, trisycl::elem::s3, trisycl::elem::s5, trisycl::elem::s7, trisycl::elem::s9, trisycl::elem::sB, trisycl::elem::sD, and trisycl::elem::sF.

9.20.2.7.2.9 TRISYCL_DECLARE_S() [1/16] template<typename DataType >

trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
0)

9.20.2.7.2.10 TRISYCL_DECLARE_S() [2/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            1  )
```

9.20.2.7.2.11 TRISYCL_DECLARE_S() [3/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            2  )
```

9.20.2.7.2.12 TRISYCL_DECLARE_S() [4/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            3  )
```

9.20.2.7.2.13 TRISYCL_DECLARE_S() [5/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            4  )
```

9.20.2.7.2.14 TRISYCL_DECLARE_S() [6/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            5  )
```

9.20.2.7.2.15 TRISYCL_DECLARE_S() [7/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            6  )
```

9.20.2.7.2.16 TRISYCL_DECLARE_S() [8/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            7  )
```

9.20.2.7.2.17 TRISYCL_DECLARE_S() [9/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            8  )
```

9.20.2.7.2.18 TRISYCL_DECLARE_S() [10/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            9  )
```

9.20.2.7.2.19 TRISYCL_DECLARE_S() [11/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            A  )
```

9.20.2.7.2.20 TRISYCL_DECLARE_S() [12/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            B  )
```

9.20.2.7.2.21 TRISYCL_DECLARE_S() [13/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            C  )
```

9.20.2.7.2.22 TRISYCL_DECLARE_S() [14/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            D  )
```

9.20.2.7.2.23 TRISYCL_DECLARE_S() [15/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            E  )
```

9.20.2.7.2.24 TRISYCL_DECLARE_S() [16/16] template<typename DataType >

```
trisycl::vec< DataType, 16 >::TRISYCL_DECLARE_S (
            F  )
```

9.20.3 Macro Definition Documentation

9.20.3.1 TRISYCL_DECLARE_S

```
#define TRISYCL_DECLARE_S(
              x )
```

```
#include <include/triSYCL/vec.hpp>
```

Value:

```
  const DataType& s##x()const {                    \
    return (*this)[0x##x];                          \
  }                                                 \
                                                    \
  DataType& s##x() {                                \
    return (*this)[0x##x];                          \
  }
```

Accessors to access hex indexed elements of a vector.

Definition at line 67 of file vec.hpp.

9.20.3.2 TRISYCL_DEFINE_VEC_TYPE

```
#define TRISYCL_DEFINE_VEC_TYPE(
              type,
              actual_type )
```

```
#include <include/triSYCL/vec.hpp>
```

Value:

```
  TRISYCL_DEFINE_VEC_TYPE_SIZE(type, 1, actual_type)          \
  TRISYCL_DEFINE_VEC_TYPE_SIZE(type, 2, actual_type)          \
  TRISYCL_DEFINE_VEC_TYPE_SIZE(type, 3, actual_type)          \
  TRISYCL_DEFINE_VEC_TYPE_SIZE(type, 4, actual_type)          \
  TRISYCL_DEFINE_VEC_TYPE_SIZE(type, 8, actual_type)          \
  TRISYCL_DEFINE_VEC_TYPE_SIZE(type, 16, actual_type)
```

Declare the vector types of a type for all the sizes.

Definition at line 540 of file vec.hpp.

9.20.3.3 TRISYCL_DEFINE_VEC_TYPE_SIZE

```
#define TRISYCL_DEFINE_VEC_TYPE_SIZE(
              type,
              size,
              actual_type )  using type##size = vec<actual_type, size>;
```

```
#include <include/triSYCL/vec.hpp>
```

A macro to define type alias, such as for type=uchar, size=4 and actual_type=unsigned char, uchar4 is equivalent to vec<unsigned char, 4>

Definition at line 536 of file vec.hpp.

9.20.3.4 TRISYCL_GEN_SWIZ2

```
#define TRISYCL_GEN_SWIZ2(
              str,
              idx0,
              idx1 )
```

```
#include <include/triSYCL/vec.hpp>
```

Value:

```
  const __swizzled_vec__<DataType, 2> str()const {   \
     return base_vec::swizzle(idx0, idx1);            \
  }                                                   \
  __swizzled_vec__<DataType, 2> str() {               \
    return base_vec::swizzle(idx0, idx1);             \
  }
```

Definition at line 76 of file vec.hpp.

9.20.3.5 TRISYCL_GEN_SWIZ3

```
#define TRISYCL_GEN_SWIZ3(
              str,
              idx0,
              idx1,
              idx2 )
```

```
#include <include/triSYCL/vec.hpp>
```

Value:

```
  const __swizzled_vec__<DataType, 3> str()const {   \
    return base_vec::swizzle(idx0, idx1, idx2);       \
  }                                                   \
  __swizzled_vec__<DataType, 3> str() {               \
    return base_vec::swizzle(idx0, idx1, idx2);       \
  }
```

Definition at line 83 of file vec.hpp.

9.20.3.6 TRISYCL_GEN_SWIZ4

```
#define TRISYCL_GEN_SWIZ4(
              str,
              idx0,
              idx1,
              idx2,
              idx3 )
```

```
#include <include/triSYCL/vec.hpp>
```

Value:

```
  const __swizzled_vec__<DataType, 4> str()const {   \
    return base_vec::swizzle(idx0, idx1, idx2, idx3); \
  }                                                   \
  __swizzled_vec__<DataType, 4> str() {               \
    return base_vec::swizzle(idx0, idx1, idx2, idx3); \
  }
```

Definition at line 90 of file vec.hpp.

9.20.4 Typedef Documentation

9.20.4.1 __swizzled_vec__

```
template<typename DataType , int numElements>
using trisycl::__swizzled_vec__ = typedef vec<DataType, numElements>
```

```
#include <include/triSYCL/vec.hpp>
```

Definition at line 55 of file vec.hpp.

9.20.5 Variable Documentation

9.20.5.1 a

```
constexpr int trisycl::elem::a = 3  [static], [constexpr]
```

Definition at line 35 of file vec.hpp.

9.20.5.2 b

```
constexpr int trisycl::elem::b = 2  [static], [constexpr]
```

Definition at line 34 of file vec.hpp.

9.20.5.3 g

```
constexpr int trisycl::elem::g = 1  [static], [constexpr]
```

Definition at line 33 of file vec.hpp.

9.20.5.4 r

```
constexpr int trisycl::elem::r = 0  [static], [constexpr]
```

Definition at line 32 of file vec.hpp.

9.20.5.5 s0

```
constexpr int trisycl::elem::s0 = 0  [static], [constexpr]
```

Definition at line 36 of file vec.hpp.

Referenced by trisycl::vec< DataType, 2 >::even(), trisycl::vec< DataType, 3 >::even(), trisycl::vec< DataType, 4 >::even(), trisycl::vec< DataType, 8 >::even(), trisycl::vec< DataType, 16 >::even(), trisycl::vec< DataType, 2 >::lo(), trisycl::vec< DataType, 3 >::lo(), trisycl::vec< DataType, 4 >::lo(), trisycl::vec< DataType, 8 >::lo(), and trisycl::vec< DataType, 16 >

9.20.5.6 s1

```
constexpr int trisycl::elem::s1 = 1  [static], [constexpr]
```

Definition at line 37 of file vec.hpp.

Referenced by trisycl::vec< DataType, 2 >::hi(), trisycl::vec< DataType, 3 >::lo(), trisycl::vec< DataType, 4 >::lo(), trisycl::vec< DataType, 8 >::lo(), trisycl::vec< DataType, 16 >::lo(), trisycl::vec< DataType, 2 >::odd(), trisycl::vec< DataType, 3 >::
trisycl::vec< DataType, 4 >::odd(), trisycl::vec< DataType, 8 >::odd(), and trisycl::vec< DataType, 16 >::odd().

9.20.5.7 s2

```
constexpr int trisycl::elem::s2 = 2  [static], [constexpr]
```

Definition at line 38 of file vec.hpp.

Referenced by trisycl::vec< DataType, 3 >::even(), trisycl::vec< DataType, 4 >::even(), trisycl::vec< DataType, 8 >::even(), trisycl::vec< DataType, 16 >::even(), trisycl::vec< DataType, 3 >::hi(), trisycl::vec< DataType, 4 >::hi(), trisycl::vec< DataType, 8 >
and trisycl::vec< DataType, 16 >::lo().

9.20.5.8 s3

```
constexpr int trisycl::elem::s3 = 3  [static], [constexpr]
```

Definition at line 39 of file vec.hpp.

Referenced by trisycl::vec< DataType, 4 >::hi(), trisycl::vec< DataType, 8 >::lo(), trisycl::vec< DataType, 16 >::lo(), trisycl::vec< DataType, 4 >::odd(), trisycl::vec< DataType, 8 >::odd(), and trisycl::vec< DataType, 16 >::odd().

9.20.5.9 s4

```
constexpr int trisycl::elem::s4 = 4  [static], [constexpr]
```

Definition at line 40 of file vec.hpp.

Referenced by trisycl::vec< DataType, 8 >::even(), trisycl::vec< DataType, 16 >::even(), trisycl::vec< DataType, 8 >::hi(), and trisycl::vec< DataType, 16 >::lo().

9.20.5.10 s5

```
constexpr int trisycl::elem::s5 = 5  [static], [constexpr]
```

Definition at line 41 of file vec.hpp.

Referenced by trisycl::vec< DataType, 8 >::hi(), trisycl::vec< DataType, 16 >::lo(), trisycl::vec< DataType, 8 >::odd(), and trisycl::vec< DataType, 16 >::odd().

9.20.5.11 s6

```
constexpr int trisycl::elem::s6 = 6  [static], [constexpr]
```

Definition at line 42 of file vec.hpp.

Referenced by trisycl::vec< DataType, 8 >::even(), trisycl::vec< DataType, 16 >::even(), trisycl::vec< DataType, 8 >::hi(), and trisycl::vec< DataType, 16 >::lo().

9.20.5.12 s7

```
constexpr int trisycl::elem::s7 = 7  [static], [constexpr]
```

Definition at line 43 of file vec.hpp.

Referenced by trisycl::vec< DataType, 8 >::hi(), trisycl::vec< DataType, 16 >::lo(), trisycl::vec< DataType, 8 >::odd(), and trisycl::vec< DataType, 16 >::odd().

9.20.5.13 s8

```
constexpr int trisycl::elem::s8 = 8  [static], [constexpr]
```

Definition at line 44 of file vec.hpp.

Referenced by trisycl::vec< DataType, 16 >::even(), and trisycl::vec< DataType, 16 >::hi().

9.20.5.14 s9

```
constexpr int trisycl::elem::s9 = 9  [static], [constexpr]
```

Definition at line 45 of file vec.hpp.

Referenced by trisycl::vec< DataType, 16 >::hi(), and trisycl::vec< DataType, 16 >::odd().

9.20.5.15 sA

```
constexpr int trisycl::elem::sA = 10  [static], [constexpr]
```

Definition at line 46 of file vec.hpp.

Referenced by trisycl::vec< DataType, 16 >::even(), and trisycl::vec< DataType, 16 >::hi().

9.20.5.16 sB

```
constexpr int trisycl::elem::sB = 11  [static], [constexpr]
```

Definition at line 47 of file vec.hpp.

Referenced by trisycl::vec< DataType, 16 >::hi(), and trisycl::vec< DataType, 16 >::odd().

9.20.5.17 sC

```
constexpr int trisycl::elem::sC = 12  [static], [constexpr]
```

Definition at line 48 of file vec.hpp.

Referenced by trisycl::vec< DataType, 16 >::even(), and trisycl::vec< DataType, 16 >::hi().

9.20.5.18 sD

```
constexpr int trisycl::elem::sD = 13  [static], [constexpr]
```

Definition at line 49 of file vec.hpp.

Referenced by trisycl::vec< DataType, 16 >::hi(), and trisycl::vec< DataType, 16 >::odd().

9.20.5.19 sE

```
constexpr int trisycl::elem::sE = 14  [static], [constexpr]
```

Definition at line 50 of file vec.hpp.

Referenced by trisycl::vec< DataType, 16 >::even(), and trisycl::vec< DataType, 16 >::hi().

9.20.5.20 sF

```
constexpr int trisycl::elem::sF = 15  [static], [constexpr]
```

Definition at line 51 of file vec.hpp.

Referenced by trisycl::vec< DataType, 16 >::hi(), and trisycl::vec< DataType, 16 >::odd().

9.20.5.21 w

```
constexpr int trisycl::elem::w = 3  [static], [constexpr]
```

Definition at line 31 of file vec.hpp.

9.20.5.22 x

```
constexpr int trisycl::elem::x = 0  [static], [constexpr]
```

Definition at line 28 of file vec.hpp.

9.20.5.23 y

```
constexpr int trisycl::elem::y = 1  [static], [constexpr]
```

Definition at line 29 of file vec.hpp.

9.20.5.24 z

```
constexpr int trisycl::elem::z = 2  [static], [constexpr]
```

Definition at line 30 of file vec.hpp.

Chapter 10

Namespace Documentation

10.1 cl Namespace Reference

Namespaces

- namespace sycl

  The official SYCL 1.2.1 specification exposes the API in the cl::sycl namespace.

10.2 cl::sycl Namespace Reference

The official SYCL 1.2.1 specification exposes the API in the cl::sycl namespace.

10.2.1 Detailed Description

The official SYCL 1.2.1 specification exposes the API in the cl::sycl namespace.

10.3 std Namespace Reference

Classes

- struct hash< trisycl::buffer< T, Dimensions, Allocator > >
- struct hash< trisycl::context >
- struct hash< trisycl::device >
- struct hash< trisycl::event >
- struct hash< trisycl::kernel >
- struct hash< trisycl::platform >
- struct hash< trisycl::queue >
- struct tuple_element< I, T >

  Each tuple element type is the same, the one of any small_array heir element.
- struct tuple_element< I, trisycl::vec< T, Size > >

  The std::tuple_element specialization for vec.
- struct tuple_size< T >

  Export the dimension as its tuple size.
- struct tuple_size< trisycl::vec< T, Size > >

  The std::tuple_size specialization for vec.

Functions

- template<std::size_t I, typename T >
  requires trisycl::detail::is_small_array<T>
  decltype(auto) get (T &&e)

  The element of the tuple is the matching element of the small_array heir.
- template<int I, typename T , int Size>
  auto get (const trisycl::vec< T, Size > &v)

  Provide a get<> accessor to have C++17 structured binding working.

10.3.1 Function Documentation

10.3.1.1 get() [1/2]

```
template<int I, typename T , int Size>
auto std::get (
             const trisycl::vec< T, Size > & v )
```

Provide a get<> accessor to have C++17 structured binding working.

Note that it works without this with current std::array based implementation on the host device but we prefer to have a more realistic const binding

Definition at line 596 of file vec.hpp.

```
00596                                        {
00597   static_assert(I < Size, "The vec<> has not enough elements!");
00598   return v[I];
00599 }
```

10.3.1.2 get() [2/2]

```
template<std::size_t I, typename T >
requires trisycl::detail::is_small_array<T>
decltype(auto) std::get (
             T && e )
```

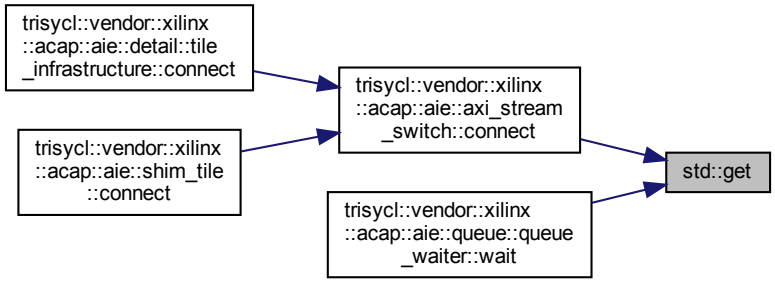
The element of the tuple is the matching element of the small_array heir.

Definition at line 402 of file small_array.hpp.

```
00403           {
00404   return e.get(I);
```

Referenced by trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::connect(), and trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter::wait().

Here is the caller graph for this function:



10.4 sycl Namespace Reference

Expose the SYCL API directly in the sycl namespace.

10.4.1 Detailed Description

Expose the SYCL API directly in the sycl namespace.

10.5 trisycl Namespace Reference

The vector type to be used as SYCL vector.

Namespaces

- namespace access
  Describe the type of access by kernels.
- namespace detail
  Some global triSYCL configuration.
- namespace drt
  The device-side runtime.
- namespace extension
- namespace facade
- namespace info
- namespace mixin
- namespace native
- namespace property
- namespace sycl_2_2
- namespace trisycl
- namespace vendor
- namespace views

Classes

- class accessor

  The accessor abstracts the way buffer or pipe data are accessed inside a kernel in a multidimensional variable length array way. More...
- class accessor< DataType, 1, AccessMode, access::target::blocking_pipe >

  The pipe accessor abstracts the way pipe data are accessed inside a kernel. More...
- class accessor< DataType, 1, AccessMode, access::target::pipe >

  The pipe accessor abstracts the way pipe data are accessed inside a kernel. More...
- class accessor_error

  Error regarding the trisycl::accessor objects defined. More...
- struct async_exception

  An error stored in an exception_list for asynchronous errors. More...
- class buffer

  A SYCL buffer is a multidimensional variable length array (à la C99 VLA or even Fortran before) that is used to store data to work on. More...
- class cl_exception

  Returns the OpenCL error code encapsulated in the exception. More...
- class compile_program_error

  Error while compiling the SYCL kernel to a SYCL device. More...
- class context

  SYCL context. More...
- class device

  SYCL device. More...
- class device_error

  The SYCL device will trigger this exception on error. More...
- class device_selector

  The SYCL heuristics to select a device. More...
- class device_type_selector

  A device selector by device_type. More...
- class device_typename_selector

  Select a device by template device_type parameter. More...
- struct elem

  Series of values to specify named swizzle indexes. More...
- struct error_handler

  User supplied error handler to call a user-provided function when an error happens from a SYCL object that was constructed with this error handler. More...
- class event

  SYCL event.
- class event_error

  Error regarding associated trisycl::event objects. More...
- class exception

  Encapsulate a SYCL error information. More...
- struct exception_list

  Exception list to store several exceptions. More...
- class feature_not_supported

  Exception thrown when an optional feature or extension is used in a kernel but its not available on the device the SYCL kernel is being enqueued on. More...
- struct group

  A group index used in a parallel_for_workitem to specify a work_group. More...
- struct h_item

`h_item<int dimensions>` *identifies an instance of a* `group::parallel_for_work_item` *function object executing at each point in a local* `range<int dimensions>` *passed to a* `parallel_for_work_item` *call or to the corresponding* `parallel_for_work_group` *call if no range is passed to the* `parallel_for_work_item` *call. More...*

- class half
- class handler

  Command group handler class. More...
- class host_accessor

  SYCL 2020 host_accessor. More...
- class id

  Define a multi-dimensional index, used for example to locate a work item. More...
- struct image
- class invalid_object_error

  Error regarding any memory objects being used inside the kernel. More...
- class invalid_parameter_error

  Error regarding parameters to the SYCL kernel, it may apply to any captured parameters to the kernel lambda. More...
- struct is_wrapper
- class item

  A SYCL item stores information on a work-item with some more context such as the definition range and offset. More...
- class kernel

  SYCL kernel. More...
- class kernel_error

  Error that occurred before or while enqueuing the SYCL kernel. More...
- class link_program_error

  Error while linking the SYCL kernel to a SYCL device. More...
- class memory_allocation_error

  Error on memory allocation on the SYCL device for a SYCL kernel. More...
- struct nd_item

  A SYCL nd_item stores information on a work-item within a work-group, with some more context such as the definition ranges. More...
- struct nd_range

  A ND-range, made by a global and local range, to specify work-group and work-item organization. More...
- class nd_range_error

  Error regarding the trisycl::nd_range specified for the SYCL kernel. More...
- class non_cl_error

  Exception for an OpenCL operation requested in a non OpenCL area. More...
- class pipe

  SYCL pipe object. More...
- class pipe_error

  A failing pipe error will trigger this exception on error. More...
- class platform

  Abstract the OpenCL platform. More...
- class platform_error

  The SYCL platform will trigger this exception on error. More...
- class profiling_error

  The SYCL runtime will trigger this error if there is an error when profiling info is enabled. More...
- class program

  SYCL program.
- class property_list

  SYCL property list.
- class queue

SYCL queue, similar to the OpenCL queue concept. More...

- class range

  A SYCL range defines a multi-dimensional index range that can be used to define launch parallel computation extent or buffer sizes. More...
- class runtime_error
- class vec

  forward declaration to use in the detail class
- class vec< DataType, 1 >
- class vec< DataType, 16 >
- class vec< DataType, 2 >
- class vec< DataType, 3 >
- class vec< DataType, 4 >
- class vec< DataType, 8 >

Typedefs

- template<typename T >
  using constant = detail::addr_space< T, access::address_space::constant_space >

  Declare a variable to be in the OpenCL constant address space.
- template<typename T >
  using constant_ptr = constant< T * >

  Declare a variable to be in the OpenCL constant address space.
- template<typename T >
  using generic = detail::addr_space< T, access::address_space::generic_space >

  Declare a variable to be in the OpenCL 2 generic address space.
- template<typename T >
  using global = detail::addr_space< T, access::address_space::global_space >

  Declare a variable to be in the OpenCL global address space.
- template<typename T >
  using global_ptr = global< T * >

  Declare a variable to be in the OpenCL global address space.
- template<typename T >
  using local = detail::addr_space< T, access::address_space::local_space >

  Declare a variable to be in the OpenCL local address space.
- template<typename T >
  using local_ptr = local< T * >

  Declare a variable to be in the OpenCL local address space.
- template<typename T >
  using priv = detail::addr_space< T, access::address_space::private_space >

  Declare a variable to be in the OpenCL private address space.
- template<typename T >
  using private_ptr = priv< T * >

  Declare a variable to be in the OpenCL private address space.
- template<typename Pointer , access::address_space AS>
  using multi_ptr = detail::address_space_ptr< Pointer, AS >

  A pointer that can be statically associated to any address-space.
- template<typename T >
  using buffer_allocator = std::allocator< T >

  The allocator objects give the programmer some control on how the memory is allocated inside SYCL.
- template<typename T >
  using image_allocator = std::allocator< T >

  The allocator used for the `image` inside SYCL.

- template<typename T >
  using map_allocator = std::allocator< T >

  The allocator used to map the memory at the same place.
- template<class T , class Alloc = std::allocator<T>>
  using vector_class = std::vector< T, Alloc >
- using string_class = std::string
- template<class R , class... ArgTypes>
  using function_class = std::function< R(ArgTypes...)>
- using mutex_class = std::mutex
- template<class T , class D = std::default_delete<T>>
  using unique_ptr_class = std::unique_ptr< T[ ], D >
- template<class T >
  using shared_ptr_class = std::shared_ptr< T >
- template<class T >
  using weak_ptr_class = std::weak_ptr< T >
- template<class T >
  using hash_class = std::hash< T >
- using exception_ptr_class = std::exception_ptr
- using default_selector = device_typename_selector< info::device_type::automatic >

  Devices selected by heuristics of the system.
- using gpu_selector = device_typename_selector< info::device_type::gpu >

  Select devices according to device type info::device::device_type::gpu from all the available OpenCL devices.
- using cpu_selector = device_typename_selector< info::device_type::cpu >

  Select devices according to device type info::device::device_type::cpu from all the available devices and heuristics.
- using host_selector = device_typename_selector< info::device_type::host >

  Selects the SYCL host CPU device that does not require an OpenCL runtime.
- using exception_ptr = std::exception_ptr

  A shared pointer to an exception as in C++ specification.
- using async_handler = function_class< void, exception_list >
- template<typename DataType , int numElements>
  using __swizzled_vec__ = vec< DataType, numElements >

Enumerations

- enum class program_state { none , compiled , linked }
- enum class rounding_mode {
  automatic , rte , rtz , rtp ,
  rtn }

  Rounding mode for vector conversions.

Functions

- template<typename Accessor >
  static auto & get_pipe_detail (Accessor &a)

  Top-level function to break circular dependencies on the the types to get the pipe implementation.
- template<typename T , access::address_space AS>
  multi_ptr< T, AS > make_ptr (multi_ptr< T, AS > pointer)

  Construct a trisycl::multi_ptr<> with the right type.
- template<typename T , access::address_space AS>
  multi_ptr< T, AS > make_ptr (T *pointer)

  Construct a trisycl::multi_ptr<> with the right type.

- template<std::forward_iterator FIB, std::forward_iterator FIE>
  buffer (FIB begin, FIE end) -> buffer< typename std::iterator_traits< FIB >::value_type, 1 >

  A deduction guide to infer the buffer type from the read-write forward iterators into a copy-in on construction and copy-back on destruction.
- buffer (auto &host_data) -> buffer< std::ranges::range_value_t< decltype(host_data)>, 1 >

  A deduction guide to infer the buffer type from the read-write range.
- template<> auto context::get_info< info::context::reference_count > () const
- template<> auto context::get_info< info::context::platform > () const
- template<> auto context::get_info< info::context::devices > () const
- template<> auto event::get_info< info::event::command_execution_status > () const
- template<> auto event::get_info< info::event::reference_count > () const
- template<> auto event::get_profiling_info< info::event_profiling::command_submit > () const
- template<> auto event::get_profiling_info< info::event_profiling::command_start > () const
- template<> auto event::get_profiling_info< info::event_profiling::command_end > () const
- template<typename... BasicType>
  id (BasicType... Args) -> id< sizeof...(Args)>

  User-defined deduction guides to deduce the rank of an id from the number of arguments.
- TRISYCL_MATH_WRAP (abs) TRISYCL_MATH_WRAP(atan) TRISYCL_MATH_WRAP3ss(clamp) TRISYCL_MATH_WRAP2s(fmax) template< typename T
- int size auto fmax (const vec< T, size > &x, const vec< T, size > &y)
- TRISYCL_MATH_WRAP2s (fmin) template< typename T
- int size auto fmin (const vec< T, size > &x, const vec< T, size > &y)
- TRISYCL_MATH_WRAP2s (frexp) template< typename T > T max(const T &x
- template<typename T , int size>
  auto max (const vec< T, size > &x, const vec< T, size > &y)
- template<typename T >
  T min (const T &x, const T &y, const T &z)
- template<typename T , int size>
  auto min (const vec< T, size > &x, const vec< T, size > &y)
- TRISYCL_MATH_WRAP2s (modf) TRISYCL_MATH_WRAP3s(remquo) TRISYCL_MATH_WRAP2(rotate) template< typename T
- int size auto cross (const vec< T, size > &x, const vec< T, size > &y)
- template<typename T , int size>
  auto length (const vec< T, size > &x)
- template<typename T , int size>
  auto dot (const vec< T, size > &x, const vec< T, size > &y)
- template<typename T , int size>
  auto normalize (const vec< T, size > &x)
- template<typename T >
  T sign (const T &val)
- template<typename T , int size>
  auto sign (const vec< T, size > &x)
- template<> auto platform::get_info< info::platform::extensions > () const
- template<> auto queue::get_info< info::queue::device > () const
- template<> auto queue::get_info< info::queue::context > () const
- template<> auto queue::get_info< info::queue::reference_count > () const
- template<typename... BasicType>
  range (BasicType... Args) -> range< sizeof...(Args)>

  User-defined deduction guides to deduce the rank of a range from the number of arguments.

Variables

- const T & y
- const T const T & z

10.5.1 Detailed Description

The vector type to be used as SYCL vector.

Small SYCL vector class.

The exception pointer type to be used as SYCL exception pointer.

The hash type to be used as SYCL hash.

The weak pointer type to be used as SYCL weak pointer.

The shared pointer type to be used as SYCL shared pointer.

The unique pointer type to be used as SYCL unique pointer.

The mutex type to be used as SYCL mutex.

The functional type to be used as SYCL function.

The string type to be used as SYCL string.

10.5.2 Typedef Documentation

10.5.2.1 exception_ptr_class

```
using trisycl::exception_ptr_class = typedef std::exception_ptr
```

Definition at line 126 of file default_classes.hpp.

10.5.2.2 function_class

```
template<class R , class...  ArgTypes>
using trisycl::function_class = typedef std::function<R(ArgTypes...)>
```

Definition at line 50 of file default_classes.hpp.

10.5.2.3 hash_class

```
template<class T >
using trisycl::hash_class = typedef std::hash<T>
```

Definition at line 114 of file default_classes.hpp.

10.5.2.4 mutex_class

```
using trisycl::mutex_class = typedef std::mutex
```

Definition at line 62 of file default_classes.hpp.

10.5.2.5 shared_ptr_class

```
template<class T >
using trisycl::shared_ptr_class = typedef std::shared_ptr<T>
```

Definition at line 88 of file default_classes.hpp.

10.5.2.6 string_class

```
using trisycl::string_class = typedef std::string
```

Definition at line 37 of file default_classes.hpp.

10.5.2.7 unique_ptr_class

```
template<class T , class D = std::default_delete<T>>
using trisycl::unique_ptr_class = typedef std::unique_ptr<T[], D>
```

Definition at line 75 of file default_classes.hpp.

10.5.2.8 vector_class

```
template<class T , class Alloc = std::allocator<T>>
using trisycl::vector_class = typedef std::vector<T, Alloc>
```

Definition at line 25 of file default_classes.hpp.

10.5.2.9 weak_ptr_class

```
template<class T >
using trisycl::weak_ptr_class = typedef std::weak_ptr<T>
```

Definition at line 101 of file default_classes.hpp.

10.5.3 Enumeration Type Documentation

10.5.3.1 program_state

```
enum class trisycl::program_state  [strong]
```

Enumerator

none	
compiled	
linked	

Definition at line 24 of file program.hpp.

```
00024                           {
00025   none,
00026   compiled,
00027   linked
00028 };
```

10.5.3.2 rounding_mode

```
enum class trisycl::rounding_mode  [strong]
```

Rounding mode for vector conversions.

Enumerator

automatic	
rte	Default rounding mode, rtz for integers, rte for floating-point.
rtz	Round to nearest even.
rtp	Round towards zero.
rtn	Round towards positive infinity. Round towards negative infinity

Definition at line 15 of file rounding_mode.hpp.

```
00015                           {
00016   automatic, /// Default rounding mode, rtz for integers, rte for floating-point
00017   rte,       /// Round to nearest even
00018   rtz,       /// Round towards zero
00019   rtp,       /// Round towards positive infinity
00020   rtn        /// Round towards negative infinity
00021 };
```

10.5.4 Function Documentation

10.5.4.1 cross()

```
int size auto trisycl::cross (
             const vec< T, size > & x,
             const vec< T, size > & y )
```

Definition at line 296 of file math.hpp.

```
00302                                                          {
00303   static_assert((size == 4 || size == 3),
00304                 "vec type should be 3 or 4 dimensions");
00305   if constexpr (size > 3) {
00306     return vec<T, size> {
00307             (x[1] * y[2]) - (x[2] * y[1]),
```

```
00308             (x[2] * y[0]) - (x[0] * y[2]),
00309             (x[0] * y[1]) - (x[1] * y[0]),
00310             0.0};
00311   } else {
```

References x, and y.

10.5.4.2 dot()

```
template<typename T , int size>
auto trisycl::dot (
             const vec< T, size > & x,
             const vec< T, size > & y )
```

Definition at line 322 of file math.hpp.

```
00323                                                 {0}));
00324 }
```

10.5.4.3 event::get_info< info::event::command_execution_status >()

```
template<>
auto trisycl::event::get_info< info::event::command_execution_status > ( ) const  [inline]
```

Definition at line 100 of file event.hpp.

```
00100                                                                 {
00101   return implementation->get_command_execution_status();
00102 }
```

10.5.4.4 event::get_info< info::event::reference_count >()

```
template<>
auto trisycl::event::get_info< info::event::reference_count > ( ) const  [inline]
```

Definition at line 105 of file event.hpp.

```
00105                                                 {
00106   return implementation->get_reference_count();
00107 }
```

10.5.4.5 event::get_profiling_info< info::event_profiling::command_end >()

```
template<>
auto trisycl::event::get_profiling_info< info::event_profiling::command_end > ( ) const  [inline]
```

Definition at line 120 of file event.hpp.

```
00120                                                                 {
00121   return implementation->get_profiling_info(info::event_profiling::command_end);
00122 }
```

10.5.4.6 event::get_profiling_info< info::event_profiling::command_start >()

```
template<>
auto trisycl::event::get_profiling_info< info::event_profiling::command_start > ( ) const
[inline]
```

Definition at line 115 of file event.hpp.

```
00115                                                                          {
00116   return implementation->get_profiling_info(info::event_profiling::command_start);
00117 }
```

10.5.4.7 event::get_profiling_info< info::event_profiling::command_submit >()

```
template<>
auto trisycl::event::get_profiling_info< info::event_profiling::command_submit > ( ) const
[inline]
```

Definition at line 110 of file event.hpp.

```
00110                                                                          {
00111   return implementation->get_profiling_info(info::event_profiling::command_submit);
00112 }
```

10.5.4.8 fmax()

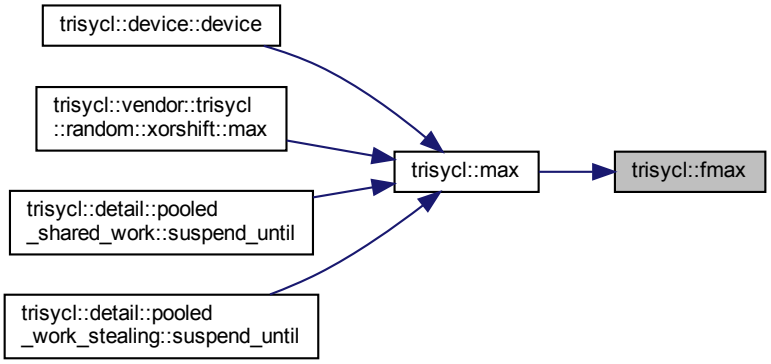
```
int size auto trisycl::fmax (
             const vec< T, size > & x,
             const vec< T, size > & y )
```

Definition at line 164 of file math.hpp.

```
00171                                {
```

Referenced by max().

Here is the caller graph for this function:



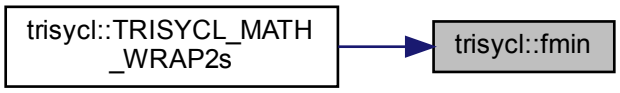
10.5.4.9 fmin()

```
int size auto trisycl::fmin (
             const vec< T, size > & x,
             const vec< T, size > & y )
```

Definition at line 180 of file math.hpp.
```
00187                                {
```

Referenced by TRISYCL_MATH_WRAP2s().

Here is the caller graph for this function:



10.5.4.10 length()

```
template<typename T , int size>
auto trisycl::length (
             const vec< T, size > & x )
```

Definition at line 315 of file math.hpp.
```
00321                                  {
```

10.5.4.11 max()

```
template<typename T , int size>
auto trisycl::max (
             const vec< T, size > & x,
             const vec< T, size > & y )
```

Definition at line 226 of file math.hpp.
```
00233                                {
```

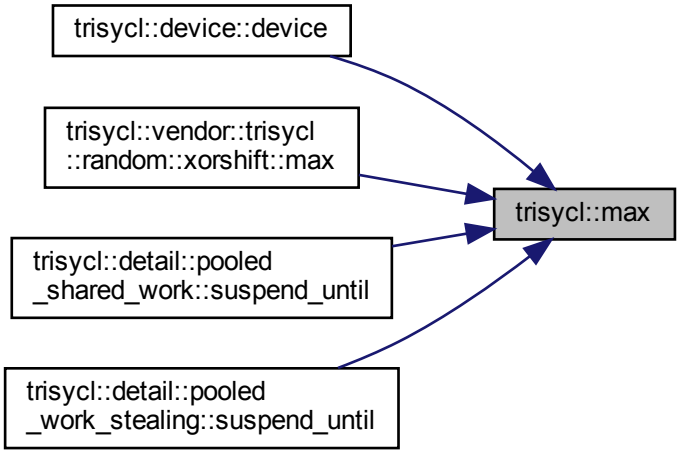
References fmax(), x, and y.

Referenced by trisycl::device::device(), trisycl::vendor::trisycl::random::xorshift< Nbits >::max(), trisycl::detail::pooled_shared_work::s
and trisycl::detail::pooled_work_stealing::suspend_until().

Here is the call graph for this function:



Here is the caller graph for this function:



10.5.4.12 min() [1/2]

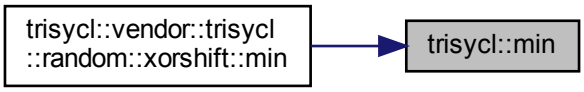
```
template<typename T >
T trisycl::min (
             const T & x,
             const T & y,
             const T & z )
```

Definition at line 235 of file math.hpp.

```
00241                                        {
```

Referenced by trisycl::vendor::trisycl::random::xorshift< Nbits >::min().

Here is the caller graph for this function:



10.5.4.13 min() [2/2]

```
template<typename T , int size>
auto trisycl::min (
             const vec< T, size > & x,
             const vec< T, size > & y )
```

Definition at line 246 of file math.hpp.

```
00253                                {
```

10.5.4.14 normalize()

```
template<typename T , int size>
auto trisycl::normalize (
             const vec< T, size > & x )
```

Definition at line 328 of file math.hpp.

```
00328                                                          {
00329   return std::inner_product(x.begin(), x.end(), y.begin(), T{0});
00330 }
```

References x, and y.

10.5.4.15 sign() [1/2]

```
template<typename T >
T trisycl::sign (
             const T & val )
```

Definition at line 337 of file math.hpp.

```
00343                       {
```

10.5.4.16 sign() [2/2]

```
template<typename T , int size>
auto trisycl::sign (
             const vec< T, size > & x )
```

Definition at line 344 of file math.hpp.

```
00346                       : -1;
```

10.5.4.17 TRISYCL_MATH_WRAP()

```
trisycl::TRISYCL_MATH_WRAP (
             abs  )
```

10.5.4.18 TRISYCL_MATH_WRAP2s() [1/3]

```
trisycl::TRISYCL_MATH_WRAP2s (
             fmin  )
```

References x, and y.

10.5.4.19 TRISYCL_MATH_WRAP2s() [2/3]

```
trisycl::TRISYCL_MATH_WRAP2s (
             frexp  ) const &
```

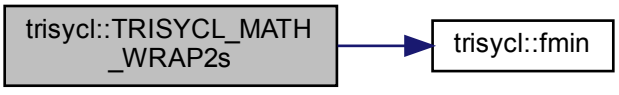
References x, and y.

10.5.4.20 TRISYCL_MATH_WRAP2s() [3/3]

```
trisycl::TRISYCL_MATH_WRAP2s (
             modf  )
```

References fmin(), x, and y.

Here is the call graph for this function:



10.5.5 Variable Documentation

10.5.5.1 y

```
const T& trisycl::y
```

Definition at line 207 of file math.hpp.

Referenced by trisycl::detail::vec< DataType, NumElements >::apply_binary_functor(), trisycl::vendor::xilinx::acap::aie::geography< trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >::check(), cross(), trisycl::vendor::xilinx::acap::aie::detail::device< La trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::for_each_memory_base(), trisycl::vendor::xilinx::acap::aie::de trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::for_each_tile_base(), trisycl::vendor::xilinx::acap::aie::detail::d trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_neighborhood(), trisycl::vendor::xilinx::acap::aie::detail::devic trisycl::vendor::xilinx::graphics::frame_grid::frame_grid(), trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::get_casca trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::get_cascade_stream_out(), trisycl::vendor::xilinx::graphics::frame_ trisycl::vendor::xilinx::graphics::image_grid::image_grid(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_cascade_end(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_cascade_start(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is trisycl::vendor::xilinx::acap::aie::layout::full::is_noc_tile(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_north_row(), trisycl::vendor::xilinx::acap::aie::layout::full::is_pl_tile(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_south_row(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_y_valid(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_y_v trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_id(), max(), trisycl::vendor::xilinx::acap::aie::device< Layout >::mem(), trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::mem(), trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memor trisycl::vendor::xilinx::acap::aie::geography< Layout >::memory_module_dx_coordinate(), trisycl::vendor::xilinx::acap::aie::geography normalize(), trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::start(), trisycl::vendor::xilinx::acap::aie::de trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::tile(), TRISYCL_MATH_WRAP2s(), trisycl::vendor::xilinx::graphics::image_ trisycl::vendor::xilinx::graphics::application::update_tile_data_image(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::validat trisycl::vendor::xilinx::acap::aie::geography< Layout >::validate_y(), and trisycl::detail::vec< DataType, NumElements >::zip().

10.5.5.2 z

```
const T const T& trisycl::z
```

Initial value:

```
{
  return std::max(x, std::max(y, z))
```

Definition at line 207 of file math.hpp.

Referenced by trisycl::detail::vec< DataType, NumElements >::zip().

10.6 trisycl::access Namespace Reference

Describe the type of access by kernels.

Enumerations

- enum class mode {
  read = 42 , write , read_write , discard_write ,
  discard_read_write , atomic }

  This describes the type of the access mode to be used via accessor.
- enum class target {
  global_buffer = 2014 , constant_buffer , local , image ,
  host_buffer , host_image , image_array , pipe ,
  blocking_pipe }

  The target enumeration describes the type of object to be accessed via the accessor.
- enum class fence_space : char { local_space , global_space , global_and_local }

  Precise the address space a barrier needs to act on.
- enum class address_space {
  global_space , local_space , constant_space , private_space ,
  generic_space }

  Enumerate the different OpenCL 2 address spaces.

10.6.1 Detailed Description

Describe the type of access by kernels.

Todo This values should be normalized to allow separate compilation with different implementations?

10.6.2 Enumeration Type Documentation

10.6.2.1 address_space

```
enum class trisycl::access::address_space  [strong]
```

Enumerate the different OpenCL 2 address spaces.

Enumerator

global_space	
local_space	
constant_space	
private_space	
generic_space	

Definition at line 28 of file address_space.hpp.

```
00028                           {
00029   global_space,
00030   local_space,
00031   constant_space,
00032   private_space,
00033   generic_space,
00034 };
```

10.6.2.2 fence_space

```
enum class trisycl::access::fence_space :  char  [strong]
```

Precise the address space a barrier needs to act on.

Enumerator

local_space	
global_space	
global_and_local	

Definition at line 62 of file access.hpp.

```
00062                           :  char {
00063     local_space,
00064     global_space,
00065     global_and_local
00066   };
```

10.6.2.3 mode

```
enum class trisycl::access::mode  [strong]
```

This describes the type of the access mode to be used via accessor.

Enumerator

read	Read-only access. Insist on the fact that read_write != read + write
write	Write-only access, but previous content *not* discarded.
read_write	Read and write access.
discard_write	Write-only access and previous content discarded.
discard_read_write	Read and write access and previous content discarded.
atomic	Atomic access.

Definition at line 32 of file access.hpp.

```
00032                  {
00033     read = 42, /**< Read-only access.  Insist on the fact that
00034 read_write != read + write */
00035     write, ///< Write-only access, but previous content *not* discarded
00036     read_write, ///< Read and write access
00037     discard_write, ///< Write-only access and previous content discarded
00038     discard_read_write, /**< Read and write access and previous
00039 content discarded*/
00040     atomic ///< Atomic access
00041   };
```

10.6.2.4 target

```
enum class trisycl::access::target  [strong]
```

The target enumeration describes the type of object to be accessed via the accessor.

Enumerator

global_buffer	
constant_buffer	
local	
image	
host_buffer	
host_image	
image_array	
pipe	
blocking_pipe	

Definition at line 47 of file access.hpp.

```
00047                     {
00048     global_buffer = 2014, //< Just pick a random number...
00049     constant_buffer,
00050     local,
00051     image,
00052     host_buffer,
00053     host_image,
00054     image_array,
00055     pipe,
00056     blocking_pipe
00057   };
```

10.7 trisycl::detail Namespace Reference

Some global triSYCL configuration.

Namespaces

- namespace spir
- namespace sycl_2_2

Classes

- class accessor

  The buffer accessor abstracts the way buffer data are accessed inside a kernel in a multidimensional variable length array way. More...
- class accessor< T, Dimensions, Mode, access::target::local >

  The local accessor specialization abstracts the way local memory is allocated to a kernel to be shared between work-items of the same work-group. More...
- class accessor_base

  The buffer accessor base is here mainly to be accessed from the task without dependency on the buffer data types. More...
- struct address_space_array

  Implementation of an array variable with an OpenCL address space. More...
- struct address_space_base

  Implementation of the base infrastructure to wrap something in an OpenCL address space. More...
- struct address_space_fundamental

  Implementation of a fundamental type with an OpenCL address space. More...
- struct address_space_object

Implementation of an object type with an OpenCL address space. More...

- struct address_space_ptr

  Implementation for an OpenCL address space pointer. More...
- struct address_space_variable

  Implementation of a variable with an OpenCL address space. More...
- class alignment
- class alignment< ::trisycl::vec< DataType, NumElements > >
- struct bool_pack
- class buffer

  A SYCL buffer is a multidimensional variable length array (à la C99 VLA or even Fortran before) that is used to store data to work on. More...
- struct buffer_base

  Factorize some template independent buffer aspects in a base class. More...
- class buffer_waiter

  A helper class to wait for the final buffer destruction if the conditions for blocking are met. More...
- class cache

  A simple thread safe cache mechanism to cache std::shared_ptr of values indexed by keys.
- struct container_element_aspect

  A mix-in to add some container element aspects. More...
- class context
- struct debug

  Class used to trace the construction, copy-construction, move-construction and destruction of classes that inherit from it. More...
- class device

  An abstract class representing various models of SYCL devices. More...
- struct display_vector

  Class used to display a vector-like type of classes that inherit from it. More...
- struct event
- struct expand_to_vector

  Allows optional expansion of a 1-element tuple to a V::dimension tuple to replicate scalar values in vector initialization. More...
- struct expand_to_vector< V, Tuple, true >

  Specialization in the case we ask for expansion. More...
- class fiber_pool
- class host_context

  SYCL host context. More...
- class host_device

  SYCL host device.
- class host_event
- class host_platform

  SYCL host platform. More...
- class host_queue

  Some implementation details about the SYCL queue.
- struct is_range

  A type trait to check if a type it a sycl::range or not.
- struct is_range< range< Dimensions > >

  Return true if template instantiating match a sycl::range.
- class kernel

  Abstract SYCL kernel. More...
- struct ocl_type

  Generate a type with some real OpenCL 2 attribute if we are on an OpenCL device. More...
- struct ocl_type< T, access::address_space::constant_space >

Add an attribute for __constant address space. More...

- struct ocl_type< T, access::address_space::global_space >

  Add an attribute for __global address space. More...
- struct ocl_type< T, access::address_space::local_space >

  Add an attribute for __local address space. More...
- struct ocl_type< T, access::address_space::private_space >

  Add an attribute for __private address space. More...
- class opencl_context

  SYCL OpenCL context.
- class opencl_device

  SYCL OpenCL device.
- class opencl_event
- class opencl_kernel

  An abstraction of the OpenCL kernel.
- class opencl_platform

  SYCL OpenCL platform. More...
- class opencl_queue

  Some implementation details about the SYCL queue.
- struct overloaded

  Helper type for the visitor to do pattern matching on invocables https://en.cppreference.com/w/cpp/utility/variant/visit. More...
- struct parallel_for_iterate

  A recursive multi-dimensional iterator that ends up calling f. More...
- struct parallel_for_iterate< 0, Range, ParallelForFunctor, Id >

  Stop the recursion when level reaches 0 by simply calling the kernel functor with the constructed id. More...
- class platform

  An abstract class representing various models of SYCL platforms. More...
- class pooled_shared_work
- class pooled_work_stealing
- class program
- class property
- struct queue

  Some implementation details about the SYCL queue.
- struct shared_ptr_implementation

  Provide an implementation as shared_ptr with total ordering and hashing to be used with algorithms and in (un)ordered containers.
- struct singleton

  Provide a singleton factory.
- struct small_array

  Define a multi-dimensional index, used for example to locate a work item or a buffer element. More...
- struct small_array_sycl

  A small array to implement SYCL objects like id, range, item... with 1, 2 or 3 elements (or even more if detail::allow_any_dimension is true), with the implicit constructors. More...
- struct small_array_tag

  Tag to define simply inheritance from small_array. More...
- struct task

  The abstraction to represent SYCL tasks executing inside command_group.
- class vec

  forward declaration to use in the detail class

Concepts

- concept is_small_array

  Concept describing a class inheriting from small_array.

Typedefs

- template<typename T , access::address_space AS>
  using addr_space = typename std::conditional< std::is_pointer< T >::value, address_space_ptr< T, AS >, typename std::conditional< std::is_class< T >::value, address_space_object< T, AS >, typename std::conditional< std::is_array< T >::value, address_space_array< T, AS >, address_space_fundamental< T, AS > >::type >::type >::type

  Dispatch the address space implementation according to the requested type.
- template<bool... v>
  using all_true = std::is_same< bool_pack< true, v... >, bool_pack< v..., true > >
- template<typename DataType , int numElements>
  using __swizzled_base_vec__ = vec< DataType, numElements >

Functions

- template<typename BufferDetail >
  static std::shared_ptr< detail::task > buffer_add_to_task (BufferDetail buf, handler *command_group_handler, bool is_write_mode)

  Proxy function to avoid some circular type recursion.
- static std::shared_ptr< detail::task > add_buffer_to_task (handler *command_group_handler, std::shared_ptr< detail::buffer_base > b, bool is_write_mode)
- template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
  auto waiter (detail::buffer< T, Dimensions > *b)

  Helper function to create a new buffer_waiter.
- template<typename V , typename Tuple , std::size_t... Is>
  std::array< typename V::element_type, V::dimension > tuple_to_array_iterate (Tuple t, std::index_sequence< Is... >)

  Helper to construct an array from initializer elements provided as a tuple.
- template<typename V , typename Tuple >
  auto tuple_to_array (Tuple t)

  Construct an array from initializer elements provided as a tuple.
- template<typename V , typename Tuple >
  auto expand (Tuple t)

  Create the array data of V from a tuple of initializer.
- template<typename KernelName , typename Functor >
  auto trace_kernel (Functor f)

  Wrap a kernel functor in some tracing messages to have start/stop information when TRISYCL_TRACE_KERNEL macro is defined.
- template<typename KernelName , typename Functor >
  void instantiate_kernel (Functor f) noexcept

  Instantiate the kernel code.
- TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX set_kernel_task_marker (detail::task &) noexcept

  Set the task to be used to launch the kernel.
- template<typename KernelName , typename Kernel >
  void launch_device_kernel (detail::task &t, Kernel k) noexcept

Gather in a single function what is necessary for the device compiler to outline the kernel or to set up the kernel and serialize the arguments.

- template<typename Range , typename Id >
  size_t constexpr linear_id (Range range, Id id, Id offset={})

  Compute a linearized array access used in the OpenCL 2 world.
- auto & assign_from_tuple (auto &dest, auto &&tuple)

  Assign arguments from a tuple.
- template<class... Ts>
  overloaded (Ts...) -> overloaded< Ts... >
- void unimplemented (const char *func, const char *file, int line)

  Display an "unimplemented" message.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void parallel_for_workitem_in_group (const group< Dimensions > &g, ParallelForFunctor f)

  Implement the loop on the work-items inside a work-group.
- static std::shared_ptr< detail::task > add_buffer_to_task (handler *command_group_handler, std::shared_ptr< detail::buffer_base > b, bool is_write_mode)

  Register a buffer as used by a task.
- template<int Dimensions = 1, typename ParallelForFunctor , typename Id >
  void parallel_for (range< Dimensions > r, ParallelForFunctor f, Id)

  Implementation of a data parallel computation with parallelism specified at launch time by a range<>.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void parallel_for (range< Dimensions > r, ParallelForFunctor f, item< Dimensions >)

  Implementation of a data parallel computation with parallelism specified at launch time by a range<>.
- template<typename F , typename R , typename A >
  auto capture_arg_v (R(F::*)(A))
- template<int Dimensions = 1, typename ParallelForFunctor >
  void parallel_for (range< Dimensions > r, ParallelForFunctor f)

  Calls the appropriate ternary parallel_for overload based on the index type of the kernel function object f.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void parallel_for_global_offset (range< Dimensions > global_size, id< Dimensions > offset, ParallelForFunctor f)

  Implementation of parallel_for with a range<> and an offset.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void parallel_for_workgroup (nd_range< Dimensions > r, ParallelForFunctor f)

  Implement the loop on the work-groups.
- template<int Dimensions, typename T_Item , typename ParallelForFunctor >
  void parallel_for_workitem (const group< Dimensions > &g, ParallelForFunctor f)

  Implement the loop on the work-items inside a work-group.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void parallel_for (nd_range< Dimensions > r, ParallelForFunctor f)

  Implement a variation of parallel_for to take into account a nd_range<>
- static auto to_tbb_range (const range< 1 > &r)
- static auto to_tbb_range (const range< 2 > &r)
- static auto to_tbb_range (const range< 3 > &r)
- static auto to_sycl_id (const tbb::blocked_range< size_t > &in)
- static auto to_sycl_id (const tbb::blocked_range2d< size_t > &in)
- static auto to_sycl_id (const tbb::blocked_range3d< size_t > &in)
- template<typename Range , typename ParallelForFunctor >
  void parallel_for_iterate (Range r, ParallelForFunctor &f)
- auto rangeify (auto r)

  Make an explicit range<1> when an integral is provided or pass the value unchanged.

Variables

- TRISYCL_WEAK_ATTRIB_PREFIX detail::cache< cl_context, detail::opencl_context > opencl_context::cache TRISYCL_WEAK_ATTRIB_SUFFIX
- template<typename T >
  constexpr auto alignment_v = alignment<T>::value
- auto constexpr underlying_value

  Get the enum value as an integral value of the underlying integer type.
- auto constexpr enum_distance

  Get the distance between 2 enum underlying values.
- auto constexpr enum_count

  Count the number of enums between 2 enum boundaries, including the boundaries.
- constexpr bool allow_any_dimension

  Map TRISYCL_ALLOW_ANY_DIMENSION to a constexpr variable.
- constexpr bool use_native_work_item

  Map TRISYCL_USE_OPENCL_ND_RANGE to a constexpr variable.
- auto prevent_arguments_from_optimization

  Avoid the interprocedural optimization to remove these arguments in the kernel instantiation by relying on some dummy static variables.
- template<typename T >
  constexpr auto is_range_v = is_range<T>::value

  A variable to check if a type is a sycl::range or not.

10.7.1 Detailed Description

Some global triSYCL configuration.

10.7.2 Typedef Documentation

10.7.2.1 __swizzled_base_vec__

```
template<typename DataType , int numElements>
using trisycl::detail::__swizzled_base_vec__ = typedef vec<DataType, numElements>
```

Definition at line 37 of file vec.hpp.

10.7.2.2 all_true

```
template<bool... v>
using trisycl::detail::all_true = typedef std::is_same<bool_pack<true, v...>, bool_pack<v...,
true> >
```

Definition at line 16 of file all_true.hpp.

10.7.3 Function Documentation

10.7.3.1 add_buffer_to_task()

```
static std::shared_ptr< detail::task > trisycl::detail::add_buffer_to_task (
            handler * command_group_handler,
            std::shared_ptr< detail::buffer_base > b,
            bool is_write_mode )  [static]
```

Register a buffer as used by a task.

This is a proxy function to avoid complicated type recursion.

Definition at line 532 of file handler.hpp.

```
00534                                        {
00535   command_group_handler->task->add_buffer(b, is_write_mode);
00536   return command_group_handler->task;
00537 }
```

References b, and trisycl::handler::task.

10.7.3.2 assign_from_tuple()

```
auto & trisycl::detail::assign_from_tuple (
            auto & dest,
            auto && tuple )
```

Assign arguments from a tuple.

Definition at line 14 of file metaprogramming.hpp.

```
00014                                                {
00015   auto expanded_assign = [&](auto&&...  elems) mutable {
00016     // Assign from each tuple element expanded
00017     dest = { std::forward<decltype(elems)>(elems)...  };
00018   };
00019   std::apply(expanded_assign, std::forward<decltype(tuple)>(tuple));
00020   // So we can use the result of the assignment expression if needed
00021   return dest;
00022 }
```

10.7.3.3 instantiate_kernel()

```
template<typename KernelName , typename Functor >
void trisycl::detail::instantiate_kernel (
            Functor f )  [noexcept]
```

Instantiate the kernel code.

To have a clear view of what a kernel is at the LLVM IR level even when a lot of optimizations are done (-O3), call the kernel from this function marked as `noinline`, otherwise the kernel code is inlined directly where it is used.

Actually some optimizations/code restructuring might duplicate the call to `instantiate_kernel` but the `noduplicate` attribute seems incompatible with `noinline` in Clang/LLVM 3.9.1, because the inlining happen when both attributes are used...

Definition at line 51 of file instantiate_kernel.hpp.

```
00051                                        {
00052   /* The outlining compiler is expected to do some massage here or
00053 around and to insert some calls to \c serialize_arg and so on */
00054   f();
00055 }
```

10.7.3.4 launch_device_kernel()

```
template<typename KernelName , typename Kernel >
void trisycl::detail::launch_device_kernel (
              detail::task & t,
              Kernel k )  [noexcept]
```

Gather in a single function what is necessary for the device compiler to outline the kernel or to set up the kernel and serialize the arguments.

The `noinline` attribute is used to force this to appear as a real function the device compiler can massage. Setup the task to be used to launch the kernel.

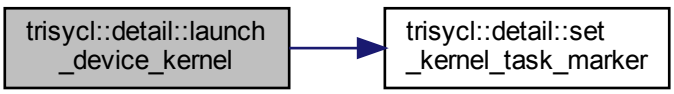
This is used by the outlining compiler.

Definition at line 91 of file instantiate_kernel.hpp.

```
00092                                                 {
00093 /** Setup the task to be used to launch the kernel.
00094
00095 This is used by the outlining compiler.  */
00096   detail::set_kernel_task_marker(t);
00097   /* A simplified version for the device just to be able to extract
00098 the kernel itself.
00099
00100 It is not a real outlining but a good approximation to be
00101 massaged by a compiler later.
00102
00103 Put the iteration space loops in the kernels for now, which makes
00104 sense for CPU emulation or FPGA pipelined execution.
00105 */
00106   instantiate_kernel<KernelName>(k);
00107 }
```

References set_kernel_task_marker().

Here is the call graph for this function:



10.7.3.5 rangeify()

```
auto trisycl::detail::rangeify (
              auto r )
```

Make an explicit range<1> when an integral is provided or pass the value unchanged.

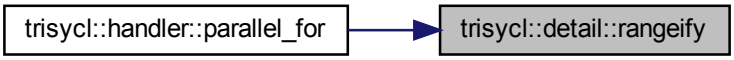
Definition at line 75 of file range.hpp.

```
00075                                 {
00076       if constexpr (std::integral<decltype(r)>)
00077         return range { r };
00078       else
```

```
00079         return r;
00080     };
```

Referenced by trisycl::handler::parallel_for().

Here is the caller graph for this function:



10.7.3.6 set_kernel_task_marker()

```
TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX trisycl::detail::set_kernel_task_↩
marker (
             detail::task &  )  [noexcept]
```

Set the task to be used to launch the kernel.

This is actually a dummy function so that the device compiler can just grab the task for passing it to the argument serialization functions from the device runtime.

It is better to use a weak linkage than having this function with extern linkage:

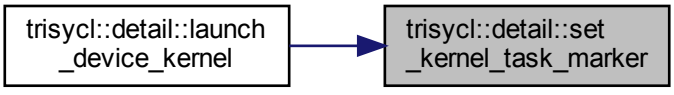
- if the device compiler is used the function is kept by the compiler because full link and LTO is not applied before the device compiler passes are applied. So there is no partial evaluation in the compiler according to what is returned by this function;
- if the device compiler is not used, the code still links and can be compiled.

Definition at line 77 of file instantiate_kernel.hpp.

```
00077                                                {
00078 }
```

Referenced by launch_device_kernel().

Here is the caller graph for this function:



10.7.4 Variable Documentation

10.7.4.1 alignment_v

```
template<typename T >
constexpr auto trisycl::detail::alignment_v = alignment<T>::value  [constexpr]
```

Definition at line 42 of file alignment_helper.hpp.

10.7.4.2 allow_any_dimension

```
constexpr bool trisycl::detail::allow_any_dimension  [constexpr]
```

Initial value:

```
=
    false
```

Map TRISYCL_ALLOW_ANY_DIMENSION to a constexpr variable.

Definition at line 33 of file global_config.hpp.

10.7.4.3 enum_count

```
auto constexpr trisycl::detail::enum_count  [constexpr]
```

Initial value:

```
= [] (auto enum_start, auto enum_end) {
  return 1 + enum_distance(enum_start, enum_end);
}
```

Count the number of enums between 2 enum boundaries, including the boundaries.

Definition at line 37 of file enum.hpp.

10.7.4.4 enum_distance

```
auto constexpr trisycl::detail::enum_distance  [constexpr]
```

Initial value:

```
= [] (auto enum_start, auto enum_end) {
  static_assert(std::is_same_v<decltype(enum_start), decltype(enum_end)>,
                "enum_start and auto enum_end should have the same type");
  return underlying_value(enum_end) - underlying_value(enum_start);
}
```

Get the distance between 2 enum underlying values.

Definition at line 28 of file enum.hpp.

10.7.4.5 is_range_v

```
template<typename T >
constexpr auto trisycl::detail::is_range_v = is_range<T>::value  [constexpr]
```

A variable to check if a type is a sycl::range or not.

Definition at line 90 of file range.hpp.

10.7.4.6 prevent_arguments_from_optimization

```
auto trisycl::detail::prevent_arguments_from_optimization  [inline]
```

Initial value:

```
= [] (auto & ...args) {

  [[gnu::used]] static auto keep = std::make_tuple(& args...);
}
```

Avoid the interprocedural optimization to remove these arguments in the kernel instantiation by relying on some dummy static variables.

Using [[gnu::used]] does not work on arguments but only on static variable, so use this function.

Definition at line 28 of file instantiate_kernel.hpp.

10.7.4.7 underlying_value

```
auto constexpr trisycl::detail::underlying_value  [constexpr]
```

Initial value:

```
= [] (auto e) {
  using value_t = std::underlying_type_t<decltype(e)>;
  return static_cast<value_t>(e);
}
```

Get the enum value as an integral value of the underlying integer type.

Definition at line 21 of file enum.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input(), and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::output().

10.7.4.8 use_native_work_item

```
constexpr bool trisycl::detail::use_native_work_item  [constexpr]
```

Initial value:

```
=

    false
```

Map TRISYCL_USE_OPENCL_ND_RANGE to a constexpr variable.

Definition at line 48 of file global_config.hpp.

Referenced by trisycl::handler::parallel_for().

10.8 trisycl::detail::spir Namespace Reference

Functions

- template<typename T , typename F >
  auto make_array_with_func (F unary_func)
- template<int Dimensions>
  nd_range< Dimensions > make_spir_nd_range ()
- template<int Dimensions>
  h_item< Dimensions > make_spir_h_item ()
- template<int Dimensions, typename Index_T >
  auto create_parallel_for_arg (Index_T index_or_range)

10.9 trisycl::detail::sycl_2_2 Namespace Reference

Classes

- class pipe

  Implement a pipe object. More...
- class pipe_accessor

  The accessor abstracts the way pipe data are accessed inside a kernel. More...
- class pipe_reservation

  The implementation of the pipe reservation station. More...
- struct reserve_id

  A private description of a reservation station. More...

10.10 trisycl::drt Namespace Reference

The device-side runtime.

Namespaces

- namespace code

  The binary code of the kernels.

Classes

- class accessor

  SYCL accessor seen from a device perspective. More...

Functions

- TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX serialize_arg (detail::task &task, std::size_t index, void ∗arg, std::size_t arg_size)

  Set an argument value of the kernel.
- TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX serialize_accessor_arg (detail::task &task, std::size_t index, void ∗arg, std::size_t arg_size)

  Set an accessor argument of the kernel.
- TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX set_kernel (detail::task &task, const char ∗kernel_name, const char ∗kernel_short_name)

  Set the kernel for a later invocation.

Variables

- TRISYCL_WEAK_ATTRIB_PREFIX std::map< std::string, std::vector< unsigned char > > TRISYCL_WEAK_ATTRIB_SUFFIX kernel_IR

  The code of the kernels arranged by name.

10.10.1 Detailed Description

The device-side runtime.

10.11 trisycl::drt::code Namespace Reference

The binary code of the kernels.

Classes

- struct program

  A binary program containing some kernels.

10.11.1 Detailed Description

The binary code of the kernels.

10.12 trisycl::extension Namespace Reference

Namespaces

- namespace ce

10.13 trisycl::extension::ce Namespace Reference

10.14 trisycl::facade Namespace Reference

Classes

- class accessor

  SYCL accessor facade providing multi-dimensional access features. More...
- class device

  SYCL device. More...

10.15 trisycl::info Namespace Reference

Classes

- struct param_traits

  Implement a meta-function from (T, value) to T' to express the return type value of an OpenCL function of kind (T, value)

Typedefs

- using device_fp_config = unsigned int
- using device_exec_capabilities = unsigned int
- using device_queue_properties = unsigned int
- using queue_profiling = bool

Enumerations

- enum class context : int { context::reference_count , context::platform , context::devices }

  Context information descriptors.
- enum class device_type : unsigned int {
  device_type::cpu , device_type::gpu , device_type::accelerator , device_type::custom ,
  device_type::automatic , device_type::host , device_type::opencl , device_type::all }

  Type of devices.
- enum class device : int {
  device::device_type , device::vendor_id , device::max_compute_units , device::max_work_item_dimensions
  ,
  device::max_work_item_sizes , device::max_work_group_size , device::preferred_vector_width_char ,
  device::preferred_vector_width_short ,
  device::preferred_vector_width_int , device::preferred_vector_width_long , device::preferred_vector_width_float
  , device::preferred_vector_width_double ,
  device::preferred_vector_width_half , device::native_vector_width_char , device::native_vector_width_short ,
  device::native_vector_width_int ,
  device::native_vector_width_long , device::native_vector_width_float , device::native_vector_width_double ,
  device::native_vector_width_half ,
  device::max_clock_frequency , device::address_bits , device::max_mem_alloc_size , device::image_support
  ,
  device::max_read_image_args , device::max_write_image_args , device::image2d_max_width , device::image2d_max_height
  ,

device::image3d_max_width , device::image3d_max_height , device::image3d_max_depth , device::image_max_buffer_size , device::image_max_array_size , device::max_samplers , device::max_parameter_size , device::mem_base_addr_align , device::half_fp_config , device::single_fp_config , device::double_fp_config , device::global_mem_cache_type , device::global_mem_cache_line_size , device::global_mem_cache_size , device::global_mem_size , device::max_constant_buffer_size , device::max_constant_args , device::local_mem_type , device::local_mem_size , device::error_correction_support , device::host_unified_memory , device::profiling_timer_resolution , device::is_endian_little , device::is_available , device::is_compiler_available , device::is_linker_available , device::execution_capabilities , device::queue_profiling , device::built_in_kernels , device::platform , device::name , device::vendor , device::driver_version , device::profile , device::version , device::opencl_c_version , device::extensions , device::printf_buffer_size , device::preferred_interop_user_sync , device::parent_device , device::partition_max_sub_devices , device::partition_properties , device::partition_affinity_domains , device::partition_type_property , device::partition_type_affinity_domain , device::reference_count }

Device information descriptors.

- enum class partition_property : int { partition_property::no_partition , partition_property::partition_equally , partition_property::partition_by_counts , partition_property::partition_by_affinity_domain }
- enum class partition_affinity_domain : int { partition_affinity_domain::not_applicable , partition_affinity_domain::numa , partition_affinity_domain::L4_cache , partition_affinity_domain::L3_cache , partition_affinity_domain::L2_cache , partition_affinity_domain::L1_cache , partition_affinity_domain::next_partitionable }
- enum class local_mem_type : int { local_mem_type::none , local_mem_type::local , local_mem_type::global }
- enum class fp_config : int { fp_config::denorm , fp_config::inf_nan , fp_config::round_to_nearest , fp_config::round_to_zero , fp_config::round_to_inf , fp_config::fma , fp_config::correctly_rounded_divide_sqrt , fp_config::soft_float }
- enum class global_mem_cache_type : int { global_mem_cache_type::none , global_mem_cache_type::read_only , global_mem_cache_type::write_only }
- enum class execution_capability : unsigned int { execution_capability::exec_kernel , execution_capability::exec_native_kernel }
- enum class event : unsigned int { command_execution_status , reference_count }
- enum class event_command_status : unsigned int { TRISYCL_OPENCL_ONLY =(= CL_SUBMITTED) , TRISYCL_OPENCL_ONLY =(= CL_RUNNING) , TRISYCL_OPENCL_ONLY =(= CL_COMPLETE) }
- enum class event_profiling : unsigned int { TRISYCL_OPENCL_ONLY =(= CL_PROFILING_COMMAND_SUBMIT) , TRISYCL_OPENCL_ONLY =(= CL_PROFILING_COMMAND_START) , TRISYCL_OPENCL_ONLY =(= CL_PROFILING_COMMAND_END) }
- enum class platform : unsigned int { platform::TRISYCL_OPENCL_ONLY =(= CL_PLATFORM_PROFILE) , platform::TRISYCL_OPENCL_ONLY =(= CL_PLATFORM_VERSION) , platform::TRISYCL_OPENCL_ONLY =(= CL_PLATFORM_NAME) , platform::TRISYCL_OPENCL_ONLY =(= CL_PLATFORM_VENDOR) , platform::TRISYCL_OPENCL_ONLY =(= CL_PLATFORM_EXTENSIONS) }

  Platform information descriptors.
- enum class queue : int { context , device , reference_count }

  Queue information descriptors.

Functions

- TRISYCL_INFO_PARAM_TRAITS (info::device::image2d_max_width, size_t)

Query the return type for get_info() on context stuff.

- TRISYCL_INFO_PARAM_TRAITS (info::device::image2d_max_height, size_t)
- TRISYCL_INFO_PARAM_TRAITS (info::device::image3d_max_width, size_t)
- TRISYCL_INFO_PARAM_TRAITS (info::device::image3d_max_height, size_t)
- TRISYCL_INFO_PARAM_TRAITS (info::device::image3d_max_depth, size_t)
- TRISYCL_INFO_PARAM_TRAITS (info::device::image_max_buffer_size, size_t)
- TRISYCL_INFO_PARAM_TRAITS (info::device::image_max_array_size, size_t)
- TRISYCL_INFO_PARAM_TRAITS (info::event_profiling::command_submit, trisycl::cl_ulong)
- TRISYCL_INFO_PARAM_TRAITS (info::event_profiling::command_start, trisycl::cl_ulong)
- TRISYCL_INFO_PARAM_TRAITS (info::event_profiling::command_end, trisycl::cl_ulong)

10.15.1 Typedef Documentation

10.15.1.1 queue_profiling

```
using trisycl::info::queue_profiling = typedef bool
```

Definition at line 18 of file queue.hpp.

10.15.2 Enumeration Type Documentation

10.15.2.1 event

```
enum class trisycl::info::event : unsigned int [strong]
```

Enumerator

command_execution_status	
reference_count	

Definition at line 6 of file event.hpp.

```
00006                 : unsigned int {
00007   command_execution_status,
00008   reference_count,
00009 };
```

10.15.2.2 event_command_status

```
enum class trisycl::info::event_command_status : unsigned int [strong]
```

Enumerator

TRISYCL_OPENCL_ONLY	
TRISYCL_OPENCL_ONLY	
TRISYCL_OPENCL_ONLY	

Definition at line 11 of file event.hpp.

```
00011                                : unsigned int {
00012     submitted TRISYCL_OPENCL_ONLY(= CL_SUBMITTED),
00013     running TRISYCL_OPENCL_ONLY(= CL_RUNNING),
00014     complete TRISYCL_OPENCL_ONLY(= CL_COMPLETE)
00015 };
```

10.15.2.3 event_profiling

```
enum class trisycl::info::event_profiling : unsigned int [strong]
```

Enumerator

TRISYCL_OPENCL_ONLY	
TRISYCL_OPENCL_ONLY	
TRISYCL_OPENCL_ONLY	

Definition at line 17 of file event.hpp.

```
00017                                : unsigned int {
00018   command_submit TRISYCL_OPENCL_ONLY(= CL_PROFILING_COMMAND_SUBMIT),
00019   command_start TRISYCL_OPENCL_ONLY(= CL_PROFILING_COMMAND_START),
00020   command_end TRISYCL_OPENCL_ONLY(= CL_PROFILING_COMMAND_END),
00021 };
```

10.15.2.4 queue

```
enum class trisycl::info::queue : int [strong]
```

Queue information descriptors.

Enumerator

context	
device	
reference_count	

Definition at line 22 of file queue.hpp.

```
00022                : int {
00023   context,
00024   device,
00025   reference_count,
00026 };
```

10.15.3 Function Documentation

10.15.3.1 TRISYCL_INFO_PARAM_TRAITS() [1/3]

```
trisycl::info::TRISYCL_INFO_PARAM_TRAITS (
             info::event_profiling::command_end ,
             trisycl::cl_ulong  )
```

10.15.3.2 TRISYCL_INFO_PARAM_TRAITS() [2/3]

```
trisycl::info::TRISYCL_INFO_PARAM_TRAITS (
             info::event_profiling::command_start ,
             trisycl::cl_ulong  )
```

10.15.3.3 TRISYCL_INFO_PARAM_TRAITS() [3/3]

```
trisycl::info::TRISYCL_INFO_PARAM_TRAITS (
             info::event_profiling::command_submit ,
             trisycl::cl_ulong  )
```

10.16 trisycl::mixin Namespace Reference

Classes

- class accessor

  SYCL accessor mixin providing multi-dimensional access features. *More...*

10.17 trisycl::native Namespace Reference

10.18 trisycl::property Namespace Reference

Namespaces

- namespace queue

10.19 trisycl::property::queue Namespace Reference

Classes

- class enable_profiling

10.20 trisycl::sycl_2_2 Namespace Reference

Classes

- class pipe

  A SYCL pipe. More...
- struct pipe_reservation

  The pipe reservation station allows to reserve an array-like view inside the pipe for ordered race-free access from various work-items for example. More...
- class static_pipe

  A SYCL static-scoped pipe equivalent to an OpenCL program-scoped pipe. More...

10.21 trisycl::trisycl Namespace Reference

Classes

- struct default_error_handler

10.21.1 Detailed Description

Todo Refactor when updating to latest specification

10.22 trisycl::vendor Namespace Reference

Namespaces

- namespace trisycl
- namespace xilinx

10.23 trisycl::vendor::trisycl Namespace Reference

Namespaces

- namespace pipe
- namespace random

  Random generators.
- namespace scope

10.24 trisycl::vendor::trisycl::pipe Namespace Reference

Classes

- class cout

  triSYCL extension for SYCL pipe object sending data to std::cout More...

10.25 trisycl::vendor::trisycl::random Namespace Reference

Random generators.

Classes

- struct xorshift

  A possible implementation of the xorshift RNGs from.

10.25.1 Detailed Description

Random generators.

10.26 trisycl::vendor::trisycl::scope Namespace Reference

Namespaces

- namespace detail

Classes

- class device

  A conceptual device implementing some device-scoped storage and based on a platform with platform-scoped storage. More...
- struct empty_device_scope

  An empty device scope to make compilation errors clearer. More...
- struct empty_platform_scope

  An empty platform scope to make compilation errors clearer. More...
- struct empty_queue_scope

  An empty queue scope to make compilation errors clearer. More...
- class platform

  A conceptual platform implementing some platform-scoped storage. More...
- class queue

  A conceptual queue implementing some queue-scoped storage constructed from a device with some device-scope storage. More...

10.27 trisycl::vendor::trisycl::scope::detail Namespace Reference

Classes

- class device

  A conceptual device implementing some device-scoped storage and based on a platform with platform-scoped storage. More...
- class platform

  A conceptual platform implementing some platform-scoped storage. More...
- class queue

  A conceptual queue implementing some queue-scoped storage constructed from a device with some device-scope storage. More...
- struct storage_type_trait
- struct storage_type_trait< ScopedType, typename ScopedType::storage_type >

Typedefs

- template<typename ScopedType >
  using storage_type_trait_t = typename storage_type_trait< ScopedType >::type

  Retrieve the type of the scoped storage of the object or `void` if none.

10.27.1 Typedef Documentation

10.27.1.1 storage_type_trait_t

```
template<typename ScopedType >
using trisycl::vendor::trisycl::scope::detail::storage_type_trait_t = typedef typename storage_type_trait<Sco
Type>::type
```

Retrieve the type of the scoped storage of the object or `void` if none.

Definition at line 30 of file util.hpp.

10.28 trisycl::vendor::xilinx Namespace Reference

Namespaces

- namespace acap
- namespace graphics
- namespace latex
- namespace partition

  Kind of array partition.

Classes

- struct partition_array

  Define an array class with partition feature. More...

Variables

- auto dataflow

  Apply dataflow execution on functions or loops.
- auto pipeline

  Execute loops in a pipelined manner.

10.28.1 Variable Documentation

10.28.1.1 dataflow

```
auto trisycl::vendor::xilinx::dataflow  [inline]
```

Initial value:

```
= [] (auto functor) noexcept {

  _ssdm_op_SpecDataflowPipeline(-1, "");
  functor();
}
```

Apply dataflow execution on functions or loops.

With this mode, Xilinx tools analyze the dataflow dependencies between sequential functions or loops and create channels (based on ping-pong RAMs or FIFOs) that allow consumer functions or loops to start operation before the producer functions or loops have completed.

This allows functions or loops to operate in parallel, which decreases latency and improves the throughput.

Parameters

in	*f*	is a function that functions or loops in f will be executed in a dataflow manner.

Definition at line 31 of file opt_decorate_func.hpp.

10.28.1.2 pipeline

```
auto trisycl::vendor::xilinx::pipeline  [inline]
```

Initial value:

```
= [] (auto functor) noexcept {

  _ssdm_op_SpecPipeline(1, 1, 0, 0, "");
```

```
  functor();
}
```

Execute loops in a pipelined manner.

A loop with pipeline mode processes a new input every clock cycle. This allows the operations of different iterations of the loop to be executed in a concurrent manner to reduce latency.

Parameters

in	*f*	is a function with an innermost loop to be executed in a pipeline way.

Definition at line 48 of file opt_decorate_func.hpp.

10.29 trisycl::vendor::xilinx::acap Namespace Reference

Namespaces

- namespace aie
- namespace debug

10.30 trisycl::vendor::xilinx::acap::aie Namespace Reference

Namespaces

- namespace detail
- namespace layout

Classes

- class accessor

  Create a SYCL-like device view of an AI Engine CGRA with some layout.
- class axi_packet
- class axi_stream_switch

  An AXI stream switch parameterized with some geographic layout. More...
- struct cascade_stream

  Implementation of the cascade stream infrastructure between AI Engine tiles. More...
- class communicator_port

  Abstract interface for a communication port.
- struct connection

  A connection between some AXI Stream Switch.
- class device

  Create a SYCL-like device view of an AI Engine CGRA with some layout.
- class dma

  Generic DMA.
- class dma_dsl

  Implement a small DSL to chain DMA & locking operations like allowed by the hardware. More...
- class fifo_channel

A FIFO channel connection.

- struct geography

  Some geographic information about the AI Engine array. More...
- struct lock_unit

  The lock infrastructure used by AI Engine memory modules and shim tiles. More...
- struct memory

  The AI Engine Memory Module infrastructure of a tile. More...
- struct memory_base

  The AI Engine Memory Module infrastructure of a tile common to all the tiles. More...
- class memory_infrastructure

  The AI Engine memory infrastructure common to all the tiles. More...
- struct port

  Abstraction of a communication port.
- class port_receiver

  A router input port directing to an AIE core input or a BLI input.
- struct program

  Define an AI Engine CGRA program with its code and memory per core. More...
- struct queue

  The AI Engine minimal queue. More...
- class receiving_dma

  Receiving DMA.
- class sending_dma

  Sending DMA.
- class shim_tile

  The AI Engine shim tile infrastructure. More...
- struct tile

  The AI Engine tile infrastructure defining the program of a tile. More...
- class tile_base

  The AI Engine tile infrastructure common to all the tiles. More...
- class tile_infrastructure

  The AI Engine tile infrastructure common to all the tiles. More...

10.31 trisycl::vendor::xilinx::acap::aie::detail Namespace Reference

Classes

- struct device

  Create a SYCL-like device view of an AI Engine CGRA with some layout.
- class tile_infrastructure

  The AI Engine tile infrastructure common to all the tiles. More...

10.32 trisycl::vendor::xilinx::acap::aie::layout Namespace Reference

Classes

- struct full

  Describe the layout of a "big" AI Engine array. More...
- struct one_pe

  Describe the layout of a tiny AI Engine with only 1 tile. More...
- struct size

  Fully specify the size of an AI Engine layout. More...
- struct small

  Describe the layout of a small AI Engine array with only a few PE. More...
- struct vc1902

  Describe the layout of a Versal VC1902 device.
- struct ve1152

Typedefs

- using xc10s10 = ve1152

  Alias to ve1152.
- using xcve1152 = ve1152

  Alias to ve1152.
- using xc10s80 = vc1902

  Alias to vc1902.
- using xcvc1902 = vc1902

  Alias to vc1902.

10.32.1 Typedef Documentation

10.32.1.1 xc10s10

```
typedef ve1152 trisycl::vendor::xilinx::acap::aie::layout::xc10s10
```

Alias to ve1152.

Definition at line 28 of file versal.hpp.

10.32.1.2 xc10s80

```
typedef vc1902 trisycl::vendor::xilinx::acap::aie::layout::xc10s80
```

Alias to vc1902.

Definition at line 39 of file versal.hpp.

10.32.1.3 xcvc1902

```
typedef vc1902 trisycl::vendor::xilinx::acap::aie::layout::xcvc1902
```

Alias to vc1902.

Definition at line 42 of file versal.hpp.

10.32.1.4 xcve1152

```
typedef ve1152 trisycl::vendor::xilinx::acap::aie::layout::xcve1152
```

Alias to ve1152.

Definition at line 31 of file versal.hpp.

10.33 trisycl::vendor::xilinx::acap::debug Namespace Reference

Classes

- class bsp_checker

  Verify that an SPMD program has a bulk-synchronous behaviour across all the tiles of the CGRA. More...

10.34 trisycl::vendor::xilinx::graphics Namespace Reference

Classes

- struct application

  A graphics application running in a separate thread to display images in a grid of tiles.
- struct frame_grid

  An application window displaying a grid of tiles.
- struct image_grid

  An application window displaying a grid of tiled images.
- struct palette

  Color palette to project a linear data space to an RGB color space.

Typedefs

- using rgb = std::array< std::uint8_t, 3 >

10.34.1 Typedef Documentation

10.34.1.1 rgb

```
typedef std::array< std::uint8_t, 3 > trisycl::vendor::xilinx::graphics::rgb
```

Definition at line 50 of file graphics.hpp.

10.35 trisycl::vendor::xilinx::latex Namespace Reference

Classes

- class context

10.36 trisycl::vendor::xilinx::partition Namespace Reference

Kind of array partition.

Classes

- struct block
  Represent a block partition.
- struct complete
  Represent a complete partition.
- struct cyclic
  Represent a cyclic partition.
- struct none
  Represent a none partition.

Enumerations

- enum class type {
  cyclic , block , complete , none ,
  cyclic , block , complete , none }
  Three real partition types: cyclic, block, complete.
- enum class type {
  cyclic , block , complete , none ,
  cyclic , block , complete , none }
  Three real partition types: cyclic, block, complete.

10.36.1 Detailed Description

Kind of array partition.

To be used when defining or declaring a vendor-supported partition array in kernel.

10.36.2 Enumeration Type Documentation

10.36.2.1 type [1/2]

```
enum class trisycl::vendor::xilinx::partition::type  [strong]
```

Three real partition types: cyclic, block, complete.

none represents non partitioned standard array.

Enumerator

cyclic	
block	
complete	
none	
cyclic	
block	
complete	
none	

Definition at line 36 of file partition_array.hpp.

```
00036                   {
00037     cyclic,
00038     block,
00039     complete,
00040     none
00041   };
```

10.36.2.2 type [2/2]

```
enum class trisycl::vendor::xilinx::partition::type  [strong]
```

Three real partition types: cyclic, block, complete.

none represents non partitioned standard array.

Enumerator

cyclic	
block	
complete	
none	
cyclic	
block	
complete	
none	

Definition at line 36 of file partition_array.hpp.

```
00036                   {
00037     cyclic,
00038     block,
00039     complete,
00040     none
00041   };
```

10.37 trisycl::views Namespace Reference

Variables

- auto constexpr enum_underlying_type

Construct a range of integral values iterating between 2 enum values, inclusive.

- auto constexpr enum_type

  Construct a range iterating between 2 enum values, inclusive.

10.37.1 Variable Documentation

10.37.1.1 enum_type

```
auto constexpr trisycl::views::enum_type  [constexpr]
```

Initial value:

```
= [] (auto begin, auto inclusive_end) {
  using begin_t = decltype(begin);
  return enum_underlying_type(begin, inclusive_end)
    | ranges::views::transform([] (auto i) { return static_cast<begin_t>(i); });
}
```

Construct a range iterating between 2 enum values, inclusive.

Parameters

begin	is the starting value
inclusive_end	is the value to stop at.

Note that since `inclusive_end` is included in the range, the range has at least 1 element.

Returns

a range of enum values from `begin` to the `inclusive_end`

Definition at line 56 of file ranges.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::tile_infrastructure().

10.37.1.2 enum_underlying_type

```
auto constexpr trisycl::views::enum_underlying_type  [constexpr]
```

Initial value:

```
= [] (auto begin, auto inclusive_end) {
  using begin_t = decltype(begin);
  using inclusive_end_t = decltype(inclusive_end);
  static_assert(std::is_same_v<begin_t, inclusive_end_t>,
                "begin and inclusive_end should be from the same enum");
  return ranges::iota_view { detail::underlying_value(begin),
                             detail::underlying_value(inclusive_end) + 1 };
}
```

Construct a range of integral values iterating between 2 enum values, inclusive.

Parameters

begin	is the starting value
inclusive_end	is the value to stop at.

Note that since `inclusive_end` is included in the range, the range has at least 1 element.

Returns

a range of integral values of the underlying type of the enum from `begin` to the `inclusive_end`

Definition at line 33 of file ranges.hpp.

Chapter 11

Concept Documentation

11.1 trisycl::detail::is_small_array Concept Reference

Concept describing a class inheriting from small_array.

`#include <small_array.hpp>`

11.1.1 Concept definition

```
template<typename T>
concept trisycl::detail::is_small_array =  std::derived_from<T, trisycl::detail::small_array_tag>
```

11.1.2 Detailed Description

Concept describing a class inheriting from small_array.

Definition at line 40 of file small_array.hpp.

Chapter 12

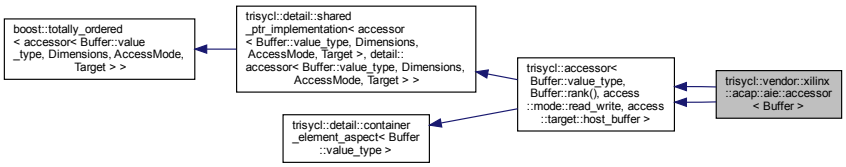
Class Documentation

12.1 trisycl::vendor::xilinx::acap::aie::accessor< Buffer > Class Template Reference

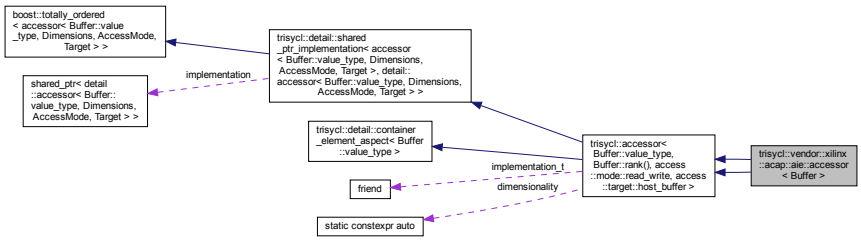
Create a SYCL-like device view of an AI Engine CGRA with some layout.

`#include <accessor.hpp>`

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::accessor< Buffer >:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::accessor< Buffer >:



Public Member Functions

- accessor (Buffer &b, auto &cgh)
- accessor (Buffer &b, auto &cgh)
- template<typename Allocator >
  accessor (buffer< DataType, Dimensions, Allocator > &target_buffer, handler &command_group_handler)

  Construct a buffer accessor from a buffer using a command group handler object from the command group scope.
- template<typename Allocator >
  accessor (buffer< DataType, Dimensions, Allocator > &target_buffer)

  Construct a buffer accessor from a buffer.
- template<typename Allocator >
  accessor (buffer< DataType, Dimensions, Allocator > &target_buffer, handler &command_group_handler, const range< Dimensions > &offset, const range< Dimensions > &range)

  Construct a buffer accessor from a buffer given a specific range for access permissions and an offset that provides the starting point for the access range using a command group handler object from the command group scope.
- accessor (const range< Dimensions > &allocation_size, handler &command_group_handler)

  Construct an accessor of dimension Dimensions with elements of type DataType using the passed range to specify the size in each dimension.
- template<typename Allocator >
  accessor (buffer< DataType, Dimensions, Allocator > &target_buffer, handler &command_group_handler)

  Construct a buffer accessor from a buffer using a command group handler object from the command group scope.
- template<typename Allocator >
  accessor (buffer< DataType, Dimensions, Allocator > &target_buffer)

  Construct a buffer accessor from a buffer.
- template<typename Allocator >
  accessor (buffer< DataType, Dimensions, Allocator > &target_buffer, handler &command_group_handler, const range< Dimensions > &offset, const range< Dimensions > &range)

  Construct a buffer accessor from a buffer given a specific range for access permissions and an offset that provides the starting point for the access range using a command group handler object from the command group scope.
- accessor (const range< Dimensions > &allocation_size, handler &command_group_handler)

  Construct an accessor of dimension Dimensions with elements of type DataType using the passed range to specify the size in each dimension.

Private Types

- using base = ::trisycl::accessor< typename Buffer::value_type, Buffer::rank(), access::mode::read_write, access::target::host_buffer >
- using base = ::trisycl::accessor< typename Buffer::value_type, Buffer::rank(), access::mode::read_write, access::target::host_buffer >

Additional Inherited Members

12.1.1 Detailed Description

template<typename Buffer>
class trisycl::vendor::xilinx::acap::aie::accessor< Buffer >

Create a SYCL-like device view of an AI Engine CGRA with some layout.

Parameters

Layout	is the layout description of the machine to instantiate with the physical size

Definition at line 28 of file accessor.hpp.

12.1.2 Member Typedef Documentation

12.1.2.1 base [1/2]

```
template<typename Buffer >
using trisycl::vendor::xilinx::acap::aie::accessor< Buffer >::base = ::trisycl::accessor<typename
Buffer::value_type, Buffer::rank(), access::mode::read_write, access::target::host_buffer>
[private]
```

Definition at line 33 of file accessor.hpp.

12.1.2.2 base [2/2]

```
template<typename Buffer >
using trisycl::vendor::xilinx::acap::aie::accessor< Buffer >::base = ::trisycl::accessor<typename
Buffer::value_type, Buffer::rank(), access::mode::read_write, access::target::host_buffer>
[private]
```

Definition at line 33 of file accessor.hpp.

12.1.3 Constructor & Destructor Documentation

12.1.3.1 accessor() [1/10]

```
template<typename Buffer >
trisycl::vendor::xilinx::acap::aie::accessor< Buffer >::accessor (
            Buffer & b,
            auto & cgh )  [inline]
```

Definition at line 40 of file accessor.hpp.

```
00041     : base { b } {}
```

12.1.3.2 accessor() [2/10]

```
template<typename Buffer >
trisycl::vendor::xilinx::acap::aie::accessor< Buffer >::accessor (
            Buffer & b,
            auto & cgh )  [inline]
```

Definition at line 40 of file accessor.hpp.

```
00041     :  base { b } {}
```

12.1.3.3 accessor() [3/10]

```
template<typename Buffer >
template<typename Allocator >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            buffer< DataType, Dimensions, Allocator > & target_buffer,
            handler & command_group_handler )  [inline]
```

Construct a buffer accessor from a buffer using a command group handler object from the command group scope.

Constructor only available for global_buffer or constant_buffer target.

access_target defines the form of access being obtained.

Todo Add template allocator type in all the accessor constructors in the specification or just use a more opaque Buffer type?

Todo fix specification where access mode should be target instead

Definition at line 104 of file accessor.hpp.

```
00105                                           :  implementation_t {
00106     new detail::accessor<DataType, Dimensions, AccessMode, Target> {
00107       target_buffer.implementation->implementation, command_group_handler }
00108   } {
00109     static_assert(Target == access::target::global_buffer
00110                   || Target == access::target::constant_buffer,
00111                   "access target should be global_buffer or constant_buffer "
00112                   "when a handler is used");
00113     // Now the implementation is created, register it
00114     implementation->register_accessor();
00115   }
```

12.1.3.4 accessor() [4/10]

```
template<typename Buffer >
template<typename Allocator >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            buffer< DataType, Dimensions, Allocator > & target_buffer )  [inline]
```

Construct a buffer accessor from a buffer.

Constructor only available for host_buffer target.

access_target defines the form of access being obtained.

Definition at line 125 of file accessor.hpp.

```
00126     :  implementation_t {
00127     new detail::accessor<DataType, Dimensions, AccessMode, Target> {
00128       target_buffer.implementation->implementation }
00129   } {
00130     static_assert(Target == access::target::host_buffer,
00131                   "without a handler, access target should be host_buffer");
00132   }
```

12.1.3.5 accessor() [5/10]

```
template<typename Buffer >
template<typename Allocator >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            buffer< DataType, Dimensions, Allocator > & target_buffer,
            handler & command_group_handler,
            const range< Dimensions > & offset,
            const range< Dimensions > & range )  [inline]
```

Construct a buffer accessor from a buffer given a specific range for access permissions and an offset that provides the starting point for the access range using a command group handler object from the command group scope.

This accessor limits the processing of the buffer to the [offset, offset+range[for every dimension. Any other parts of the buffer will be unaffected.

Constructor only available for access modes global_buffer, and constant_buffer (see Table "Buffer accessor constructors"). access_target defines the form of access being obtained.

This accessor is recommended for discard-write and discard read write access modes, when the unaffected parts of the processing should be retained.

Definition at line 153 of file accessor.hpp.

```
00156                                          {
00157     TRISYCL_UNIMPL;
00158   }
```

12.1.3.6 accessor() [6/10]

```
template<typename Buffer >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            const range< Dimensions > & allocation_size,
            handler & command_group_handler )  [inline]
```

Construct an accessor of dimension Dimensions with elements of type DataType using the passed range to specify the size in each dimension.

It needs as a parameter a command group handler object from the command group scope. Constructor only available if AccessMode is local, see Table 3.25.

Definition at line 169 of file accessor.hpp.

```
00171     :  implementation_t { new detail::accessor<DataType,
00172                                                Dimensions,
00173                                                AccessMode,
00174                                                access::target::local> {
00175       allocation_size, command_group_handler
00176         }
00177   }
00178   {
00179     static_assert(Target == access::target::local,
00180                   "This accessor constructor requires "
00181                   "access target be local");
00182   }
```

12.1.3.7 accessor() [7/10]

```
template<typename Buffer >
template<typename Allocator >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            buffer< DataType, Dimensions, Allocator > & target_buffer,
            handler & command_group_handler )  [inline]
```

Construct a buffer accessor from a buffer using a command group handler object from the command group scope.

Constructor only available for global_buffer or constant_buffer target.

access_target defines the form of access being obtained.

Todo Add template allocator type in all the accessor constructors in the specification or just use a more opaque Buffer type?

Todo fix specification where access mode should be target instead

Definition at line 104 of file accessor.hpp.

```
00105                                       :  implementation_t {
00106     new detail::accessor<DataType, Dimensions, AccessMode, Target> {
00107       target_buffer.implementation->implementation, command_group_handler }
00108   } {
00109     static_assert(Target == access::target::global_buffer
00110                   || Target == access::target::constant_buffer,
00111                   "access target should be global_buffer or constant_buffer "
00112                   "when a handler is used");
00113     // Now the implementation is created, register it
00114     implementation->register_accessor();
00115   }
```

12.1.3.8 accessor() [8/10]

```
template<typename Buffer >
template<typename Allocator >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            buffer< DataType, Dimensions, Allocator > & target_buffer )  [inline]
```

Construct a buffer accessor from a buffer.

Constructor only available for host_buffer target.

access_target defines the form of access being obtained.

Definition at line 125 of file accessor.hpp.

```
00126     :  implementation_t {
00127     new detail::accessor<DataType, Dimensions, AccessMode, Target> {
00128       target_buffer.implementation->implementation }
00129   } {
00130     static_assert(Target == access::target::host_buffer,
00131                   "without a handler, access target should be host_buffer");
00132   }
```

12.1.3.9 accessor() [9/10]

```
template<typename Buffer >
template<typename Allocator >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            buffer< DataType, Dimensions, Allocator > & target_buffer,
            handler & command_group_handler,
            const range< Dimensions > & offset,
            const range< Dimensions > & range )  [inline]
```

Construct a buffer accessor from a buffer given a specific range for access permissions and an offset that provides the starting point for the access range using a command group handler object from the command group scope.

This accessor limits the processing of the buffer to the [offset, offset+range[for every dimension. Any other parts of the buffer will be unaffected.

Constructor only available for access modes global_buffer, and constant_buffer (see Table "Buffer accessor constructors"). access_target defines the form of access being obtained.

This accessor is recommended for discard-write and discard read write access modes, when the unaffected parts of the processing should be retained.

Definition at line 153 of file accessor.hpp.

```
00156                                          {
00157     TRISYCL_UNIMPL;
00158   }
```

12.1.3.10 accessor() [10/10]

```
template<typename Buffer >
trisycl::accessor< DataType, Dimensions, AccessMode, Target >::accessor (
            const range< Dimensions > & allocation_size,
            handler & command_group_handler )  [inline]
```

Construct an accessor of dimension Dimensions with elements of type DataType using the passed range to specify the size in each dimension.

It needs as a parameter a command group handler object from the command group scope. Constructor only available if AccessMode is local, see Table 3.25.

Definition at line 169 of file accessor.hpp.

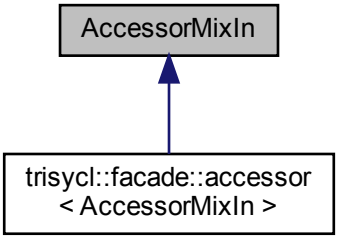
```
00171     :  implementation_t { new detail::accessor<DataType,
00172                                                Dimensions,
00173                                                AccessMode,
00174                                                access::target::local> {
00175       allocation_size, command_group_handler
00176         }
00177   }
00178   {
00179     static_assert(Target == access::target::local,
00180                   "This accessor constructor requires "
00181                   "access target be local");
00182   }
```

The documentation for this class was generated from the following files:

- include/sycl/vendor/Xilinx/acap/aie/accessor.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/accessor.hpp

12.2 AccessorMixIn Class Reference

Inheritance diagram for AccessorMixIn:



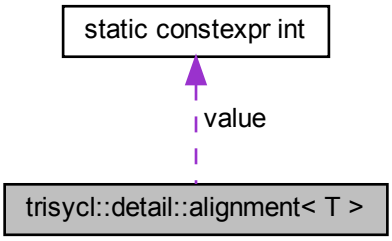
The documentation for this class was generated from the following file:

- include/triSYCL/accessor/facade/accessor.hpp

12.3 trisycl::detail::alignment< T > Class Template Reference

`#include <alignment_helper.hpp>`

Collaboration diagram for trisycl::detail::alignment< T >:



Static Public Attributes

- static constexpr int value = 0

12.3.1 Detailed Description

template<typename T>
class trisycl::detail::alignment< T >

Definition at line 24 of file alignment_helper.hpp.

12.3.2 Member Data Documentation

12.3.2.1 value

```
template<typename T >
constexpr int trisycl::detail::alignment< T >::value = 0  [static], [constexpr]
```

Definition at line 25 of file alignment_helper.hpp.

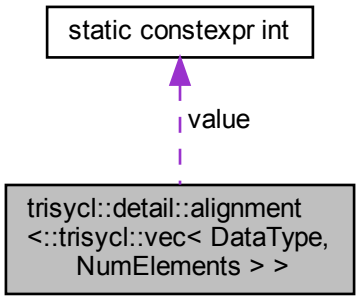
The documentation for this class was generated from the following file:

- include/triSYCL/detail/alignment_helper.hpp

12.4 trisycl::detail::alignment<::trisycl::vec< DataType, NumElements > > Class Template Reference

```
#include <alignment_helper.hpp>
```

Collaboration diagram for trisycl::detail::alignment<::trisycl::vec< DataType, NumElements > >:



Static Public Attributes

- static constexpr int value = get_align()

Static Private Member Functions

- static constexpr int get_align ()

12.4.1 Detailed Description

template<typename DataType, int NumElements>
class trisycl::detail::alignment<::trisycl::vec< DataType, NumElements > >

Definition at line 29 of file alignment_helper.hpp.

12.4.2 Member Function Documentation

12.4.2.1 get_align()

```
template<typename DataType , int NumElements>
static constexpr int trisycl::detail::alignment<::trisycl::vec< DataType, NumElements > >↩
::get_align ( )  [inline], [static], [constexpr], [private]
```

Definition at line 31 of file alignment_helper.hpp.

```
00031                                       {
00032       if  constexpr (NumElements == 3)
00033         return 4 * sizeof(DataType);
00034       else
00035         return NumElements * sizeof(DataType);
00036     }
```

12.4.3 Member Data Documentation

12.4.3.1 value

```
template<typename DataType , int NumElements>
constexpr int trisycl::detail::alignment<::trisycl::vec< DataType, NumElements > >::value =
get_align()  [static], [constexpr]
```

Definition at line 38 of file alignment_helper.hpp.

The documentation for this class was generated from the following file:

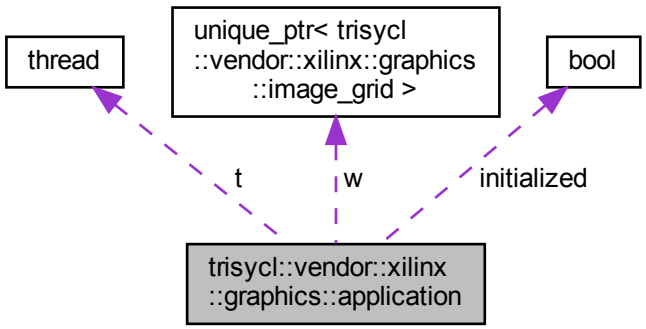
- include/triSYCL/detail/alignment_helper.hpp

12.5 trisycl::vendor::xilinx::graphics::application Struct Reference

A graphics application running in a separate thread to display images in a grid of tiles.

`#include <graphics.hpp>`

Collaboration diagram for trisycl::vendor::xilinx::graphics::application:



Public Member Functions

- auto & start (int &argc, char **&argv, int nx, int ny, int image_x, int image_y, int zoom)
  Start the graphics application.
- void wait ()
  Wait for the graphics window to end.
- bool is_done ()
  Test if the window has been closed.
- bool is_done_barrier ()
  Test if the window has been closed after synchronizing with a barrier.
- template<typename DataType , typename RangeValue >
  void update_tile_data_image (int x, int y, DataType data, RangeValue min_value, RangeValue max_value)
  Update the image of a tile.
- graphics::image_grid & image_grid ()
  Return the image_grid in this application.
- ~application ()
  The destructor waiting for graphics to end.
- auto & start (int &argc, char **&argv, int nx, int ny, int image_x, int image_y, int zoom)
  Start the graphics application.
- void wait ()
  Wait for the graphics window to end.
- bool is_done ()
  Test if the window has been closed.
- bool is_done_barrier ()
  Test if the window has been closed after synchronizing with a barrier.

- template<typename DataType , typename RangeValue >
  void update_tile_data_image (int x, int y, DataType data, RangeValue min_value, RangeValue max_value)

  Update the image of a tile.
- graphics::image_grid & image_grid ()

  Return the image_grid in this application.
- ~application ()

  The destructor waiting for graphics to end.

Public Attributes

- std::thread t
- std::unique_ptr< graphics::image_grid > w
- bool initialized = false

12.5.1 Detailed Description

A graphics application running in a separate thread to display images in a grid of tiles.

Definition at line 594 of file graphics.hpp.

12.5.2 Constructor & Destructor Documentation

12.5.2.1 ~application() [1/2]

```
trisycl::vendor::xilinx::graphics::application::~application ( )  [inline]
```

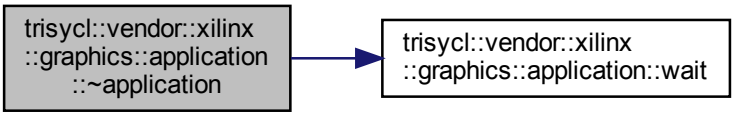
The destructor waiting for graphics to end.

Definition at line 703 of file graphics.hpp.

```
00703                 {
00704     // If the graphics thread is still running, wait for it to exit
00705     if (t.joinable())
00706       wait();
00707   }
```

References t, and wait().

Here is the call graph for this function:



12.5.2.2 ~application() [2/2]

```
trisycl::vendor::xilinx::graphics::application::~application ( )  [inline]
```

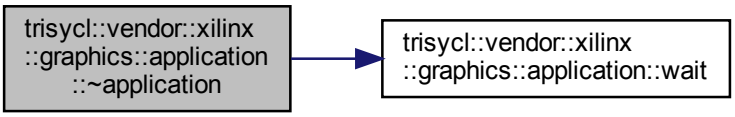
The destructor waiting for graphics to end.

Definition at line 703 of file graphics.hpp.

```
00703                  {
00704     // If the graphics thread is still running, wait for it to exit
00705     if (t.joinable())
00706       wait();
00707   }
```

References t, and wait().

Here is the call graph for this function:



12.5.3 Member Function Documentation

12.5.3.1 image_grid() [1/2]

```
graphics::image_grid & trisycl::vendor::xilinx::graphics::application::image_grid ( )  [inline]
```

Return the image_grid in this application.

Definition at line 697 of file graphics.hpp.

```
00697                                  {
00698     return *w;
00699   }
```

References w.

12.5.3.2 image_grid() [2/2]

```
graphics::image_grid & trisycl::vendor::xilinx::graphics::application::image_grid ( )  [inline]
```

Return the image_grid in this application.

Definition at line 697 of file graphics.hpp.

```
00697                                  {
00698     return *w;
00699   }
```

References w.

12.5.3.3 is_done() [1/2]

```
bool trisycl::vendor::xilinx::graphics::application::is_done ( )  [inline]
```

Test if the window has been closed.

Definition at line 660 of file graphics.hpp.

```
00660                  {
00661     return w->done;
00662   }
```

References w.

12.5.3.4 is_done() [2/2]

```
bool trisycl::vendor::xilinx::graphics::application::is_done ( )  [inline]
```

Test if the window has been closed.

Definition at line 660 of file graphics.hpp.

```
00660                  {
00661     return w->done;
00662   }
```

References w.

12.5.3.5 is_done_barrier() [1/2]

```
bool trisycl::vendor::xilinx::graphics::application::is_done_barrier ( )  [inline]
```

Test if the window has been closed after synchronizing with a barrier.

Definition at line 666 of file graphics.hpp.

```
00666                          {
00667     return w->is_done_barrier();
00668   }
```

References w.

12.5.3.6 is_done_barrier() [2/2]

```
bool trisycl::vendor::xilinx::graphics::application::is_done_barrier ( )  [inline]
```

Test if the window has been closed after synchronizing with a barrier.

Definition at line 666 of file graphics.hpp.

```
00666                          {
00667     return w->is_done_barrier();
00668   }
```

References w.

12.5.3.7 start() [1/2]

```
auto & trisycl::vendor::xilinx::graphics::application::start (
            int & argc,
            char **& argv,
            int nx,
            int ny,
            int image_x,
            int image_y,
            int zoom )  [inline]
```

Start the graphics application.

Parameters

in,out	*argc*	is the standard C program argument number
in,out	*argv*	is the standard C program argument array
in	*nx*	is the numer of tiles horizontally
in	*ny*	is the numer of tiles vertically
in	*image_x*	is the horizontal image pixel size
in	*image_y*	is the vertical image pixel size
in	*zoom*	is the zooming factor applied to image pixels, both horizontally and vertically

Returns

a reference on ∗this to allow operation chaining

Definition at line 618 of file graphics.hpp.

```
00619                                                                    {
00620     // To be sure not passing over the asynchronous graphics start
00621     std::promise<void> graphics_initialization;
00622
00623     /* Put all the graphics in its own thread.  Since
00624 Gtk::Application::create might modify argc and argv, capture by
00625 reference.  Since we have to wait for this thread, there should
00626 not be a read from freed memory issue.  */
00627     t = std::thread { [&]() mutable {
00628         // An application allowing several instance running at the same time
00629         auto a =
00630           Gtk::Application::create(argc, argv, "com.xilinx.trisycl.graphics",
00631                                    Gio::APPLICATION_NON_UNIQUE);
00632         /* Create the graphics object in this thread so the dispatcher
00633 is bound to this thread too */
00634         w.reset(new graphics::image_grid { nx, ny, image_x, image_y, zoom });
00635         w->set_close_action([&] {
00636             w->done = true;
00637           });
00638         // OK, the graphics system is in a usable state, unleash the main thread
00639         graphics_initialization.set_value();
00640         // Launch the graphics event loop handling with the graphics life
00641         a->run(*w);
00642
00643         // Advertise that the graphics is shutting down
00644         w->done = true;
00645       } };
00646     // Wait for the graphics to start
00647     graphics_initialization.get_future().get();
00648
00649     return *this;
00650   }
```

References a, t, and w.

12.5.3.8 start() [2/2]

```
auto & trisycl::vendor::xilinx::graphics::application::start (
            int & argc,
            char **& argv,
            int nx,
            int ny,
            int image_x,
            int image_y,
            int zoom )  [inline]
```

Start the graphics application.

Parameters

in,out	*argc*	is the standard C program argument number
in,out	*argv*	is the standard C program argument array
in	*nx*	is the numer of tiles horizontally
in	*ny*	is the numer of tiles vertically
in	*image_x*	is the horizontal image pixel size
in	*image_y*	is the vertical image pixel size
in	*zoom*	is the zooming factor applied to image pixels, both horizontally and vertically

Returns

a reference on ∗this to allow operation chaining

Definition at line 618 of file graphics.hpp.

```
00619                                                                    {
00620     // To be sure not passing over the asynchronous graphics start
00621     std::promise<void> graphics_initialization;
00622
00623     /* Put all the graphics in its own thread.  Since
00624 Gtk::Application::create might modify argc and argv, capture by
00625 reference.  Since we have to wait for this thread, there should
00626 not be a read from freed memory issue.  */
00627     t = std::thread { [&]() mutable {
00628         // An application allowing several instance running at the same time
00629         auto a =
00630           Gtk::Application::create(argc, argv, "com.xilinx.trisycl.graphics",
00631                                    Gio::APPLICATION_NON_UNIQUE);
00632         /* Create the graphics object in this thread so the dispatcher
00633 is bound to this thread too */
00634         w.reset(new graphics::image_grid { nx, ny, image_x, image_y, zoom });
00635         w->set_close_action([&] {
00636             w->done = true;
00637           });
00638         // OK, the graphics system is in a usable state, unleash the main thread
00639         graphics_initialization.set_value();
00640         // Launch the graphics event loop handling with the graphics life
00641         a->run(*w);
00642
00643         // Advertise that the graphics is shutting down
00644         w->done = true;
00645       } };
00646     // Wait for the graphics to start
00647     graphics_initialization.get_future().get();
00648
00649     return *this;
00650   }
```

References a, t, and w.

12.5.3.9 update_tile_data_image() [1/2]

```
template<typename DataType , typename RangeValue >
void trisycl::vendor::xilinx::graphics::application::update_tile_data_image (
            int x,
            int y,
            DataType data,
            RangeValue min_value,
            RangeValue max_value )  [inline]
```

Update the image of a tile.

Parameters

in	*x*	is the tile horizontal id
in	*y*	is the tile vertical id
in	*data*	is a 2D MDspan of extent at most image_y by image_x (only the pixels of the extents are drawn) or a pointer to a 2D linearized area of exactly image_y by image_x pixel values
in	*min_value*	is the value represented with minimum of graphics palette color
in	*max_value*	is the value represented with maximum of graphics palette color

Definition at line 688 of file graphics.hpp.

```
00691                                                 {
00692     w->update_tile_data_image(x, y, data, min_value, max_value);
00693   };
```

References w, x, and trisycl::y.

12.5.3.10 update_tile_data_image() [2/2]

```
template<typename DataType , typename RangeValue >
void trisycl::vendor::xilinx::graphics::application::update_tile_data_image (
            int x,
            int y,
            DataType data,
            RangeValue min_value,
            RangeValue max_value )  [inline]
```

Update the image of a tile.

Parameters

in	*x*	is the tile horizontal id
in	*y*	is the tile vertical id
in	*data*	is a 2D MDspan of extent at most image_y by image_x (only the pixels of the extents are drawn) or a pointer to a 2D linearized area of exactly image_y by image_x pixel values
in	*min_value*	is the value represented with minimum of graphics palette color
in	*max_value*	is the value represented with maximum of graphics palette color

Definition at line 688 of file graphics.hpp.

```
00691                                                   {
00692     w->update_tile_data_image(x, y, data, min_value, max_value);
00693   };
```

References w, x, and trisycl::y.

12.5.3.11 wait() [1/2]

```
void trisycl::vendor::xilinx::graphics::application::wait ( )  [inline]
```

Wait for the graphics window to end.

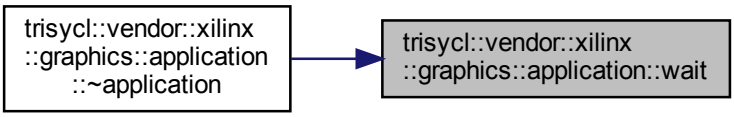
Definition at line 654 of file graphics.hpp.

```
00654                {
00655     t.join();
00656   }
```

References t.

Referenced by ~application().

Here is the caller graph for this function:



12.5.3.12 wait() [2/2]

```
void trisycl::vendor::xilinx::graphics::application::wait ( )  [inline]
```

Wait for the graphics window to end.

Definition at line 654 of file graphics.hpp.

```
00654                {
00655     t.join();
00656   }
```

References t.

12.5.4 Member Data Documentation

12.5.4.1 initialized

```
bool trisycl::vendor::xilinx::graphics::application::initialized = false
```

Definition at line 597 of file graphics.hpp.

12.5.4.2 t

```
std::thread trisycl::vendor::xilinx::graphics::application::t
```

Definition at line 595 of file graphics.hpp.

Referenced by start(), wait(), and ~application().

12.5.4.3 w

```
std::unique_ptr< graphics::image_grid > trisycl::vendor::xilinx::graphics::application::w
```

Definition at line 596 of file graphics.hpp.

Referenced by image_grid(), is_done(), is_done_barrier(), start(), and update_tile_data_image().

The documentation for this struct was generated from the following files:

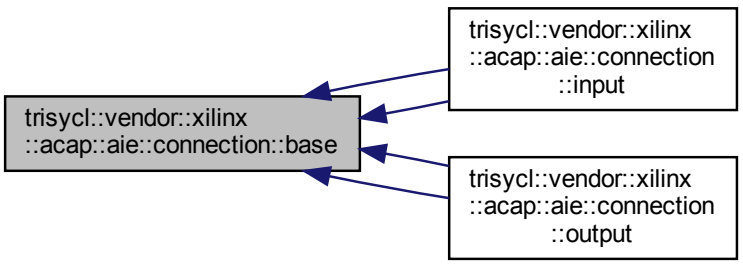
- include/sycl/vendor/Xilinx/graphics.hpp
- include/triSYCL/vendor/Xilinx/graphics.hpp

12.6 trisycl::vendor::xilinx::acap::aie::connection::base Struct Reference

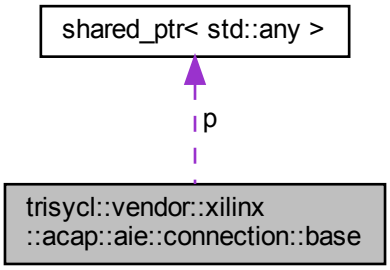
Base class of connection port.

```
#include <connection.hpp>
```

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::connection::base:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::connection::base:



Public Member Functions

- base (const connection &c)

  Construct a port from a connection.
- base ()=default
- template<typename T >
  auto pipe_of ()

  Access the type erased pipe as type T.
- base (const connection &c)

  Construct a port from a connection.
- base ()=default
- template<typename T >
  auto pipe_of ()

  Access the type erased pipe as type T.

Public Attributes

- std::shared_ptr< std::any > p

  Keep track of the communication media.

12.6.1 Detailed Description

Base class of connection port.

Definition at line 286 of file connection.hpp.

12.6.2 Constructor & Destructor Documentation

12.6.2.1 base() [1/4]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base (
            const connection & c )  [inline]
```

Construct a port from a connection.

Definition at line 292 of file connection.hpp.
```
00293       : p { c.p } {}
```

12.6.2.2 base() [2/4]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base ( )  [default]
```

12.6.2.3 base() [3/4]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base (
            const connection & c )  [inline]
```

Construct a port from a connection.

Definition at line 292 of file connection.hpp.
```
00293       : p { c.p } {}
```

12.6.2.4 base() [4/4]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base ( )  [default]
```

12.6.3 Member Function Documentation

12.6.3.1 pipe_of() [1/2]

```
template<typename T >
auto trisycl::vendor::xilinx::acap::aie::connection::base::pipe_of ( )  [inline]
```

Access the type erased pipe as type T.

Parameters

in	*T*	is the type of the data transmitted by the connection

Exceptions

trisycl::runtime_error	if the connection has not the correct type

Definition at line 307 of file connection.hpp.

```
00307                   {
00308       try {
00309         return std::any_cast<::trisycl::sycl_2_2::static_pipe<T, 4»(*p);
00310       } catch (std::bad_any_cast &) {
00311         throw ::trisycl::runtime_error {
00312           "The current connection is not of type "s
00313           + boost::typeindex::type_id<T>().pretty_name() };
00314       }
00315     }
```

12.6.3.2 pipe_of() [2/2]

```
template<typename T >
auto trisycl::vendor::xilinx::acap::aie::connection::base::pipe_of ( )  [inline]
```

Access the type erased pipe as type T.

Parameters

in	*T*	is the type of the data transmitted by the connection

Exceptions

trisycl::runtime_error	if the connection has not the correct type

Definition at line 307 of file connection.hpp.

```
00307                   {
00308       try {
00309         return std::any_cast<::trisycl::sycl_2_2::static_pipe<T, 4»(*p);
00310       } catch (std::bad_any_cast &) {
00311         throw ::trisycl::runtime_error {
00312           "The current connection is not of type "s
00313           + boost::typeindex::type_id<T>().pretty_name() };
00314       }
00315     }
```

12.6.4 Member Data Documentation

12.6.4.1 p

```
std::shared_ptr< std::any > trisycl::vendor::xilinx::acap::aie::connection::base::p
```

Keep track of the communication media.

Definition at line 288 of file connection.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::connection::input::in(), and trisycl::vendor::xilinx::acap::aie::connection::output::out().

The documentation for this struct was generated from the following files:

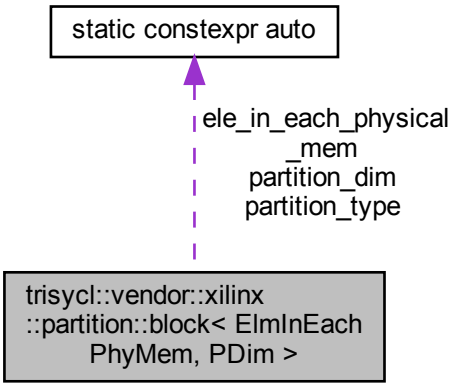
- include/sycl/vendor/Xilinx/acap/aie/connection.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/connection.hpp

12.7 trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim > Struct Template Reference

Represent a block partition.

`#include <partition_array.hpp>`

Collaboration diagram for trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim >:



Static Public Attributes

- static constexpr auto ele_in_each_physical_mem = ElmInEachPhyMem
- static constexpr auto partition_dim = PDim
- static constexpr auto partition_type = type::block

12.7.1 Detailed Description

template<std::size_t ElmInEachPhyMem = 1, std::size_t PDim = 1>
struct trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim >

Represent a block partition.

The single array would be partitioned into several small physical memories and can be accessed simultaneously, too. However, the first physical memory will be filled up first, then the next.

That is if we have a 4-element array which contains 4 integers 0, 1, 2, and 3. If we set factor to 2, and partition dimension to 1 for this cyclic partition array. Then, the contents of this array will be distributed to 2 physical memories: one contains 1, 2 and the other contains 3,4.

Parameters

ElmInEachPhyMem	is the number of elements in each physical memory that user wants to have.
PDim	is the dimension that user wants to apply block partition on. If PDim is 0, all dimensions will be partitioned with block order.

Todo Deal with multi-dimension array. Now, since we can only deal with 1-dim, PDim is set to 1 by default.

The single array would be partitioned into several small physical memories and can be accessed simultaneously, too. However, the first physical memory will be filled up first, then the next.

That is if we have a 4-element array which contains 4 integers 0, 1, 2, and 3. If we set factor to 2, and partition dimension to 1 for this cyclic partition array. Then, the contents of this array will be distributed to 2 physical memories: one contains 1, 2 and the other contains 3,4.

Parameters

ElmInEachPhyMem	is the number of elements in each physical memory that user wants to have.
PDim	is the dimension that user wants to apply block partition on. If PDim is 0, all dimensions will be partitioned with block order.

Todo Deal with multi-dimension array. Now, since we can only deal with 1-dim, PDim is set to 1 by default.

Definition at line 98 of file partition_array.hpp.

12.7.2 Member Data Documentation

12.7.2.1 ele_in_each_physical_mem

```
template<std::size_t ElmInEachPhyMem = 1, std::size_t PDim = 1>
static constexpr auto trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim >↩
::ele_in_each_physical_mem = ElmInEachPhyMem  [static], [constexpr]
```

Definition at line 99 of file partition_array.hpp.

12.7.2.2 partition_dim

```
template<std::size_t ElmInEachPhyMem = 1, std::size_t PDim = 1>
static constexpr auto trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim >↩
::partition_dim = PDim  [static], [constexpr]
```

Definition at line 100 of file partition_array.hpp.

12.7.2.3 partition_type

```
template<std::size_t ElmInEachPhyMem = 1, std::size_t PDim = 1>
static constexpr auto trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim >↩
::partition_type = type::block  [static], [constexpr]
```

Definition at line 101 of file partition_array.hpp.

The documentation for this struct was generated from the following files:

- include/sycl/vendor/Xilinx/fpga/partition_array.hpp
- include/triSYCL/vendor/Xilinx/fpga/partition_array.hpp

12.8 trisycl::detail::bool_pack< ... > Struct Template Reference

12.8.1 Detailed Description

template<bool...>
struct trisycl::detail::bool_pack< ... >

Definition at line 13 of file all_true.hpp.

The documentation for this struct was generated from the following file:

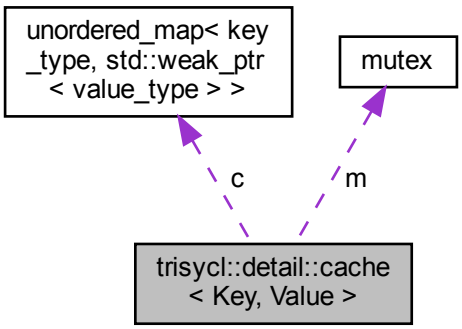
- include/triSYCL/detail/all_true.hpp

12.9 trisycl::detail::cache< Key, Value > Class Template Reference

A simple thread safe cache mechanism to cache std::shared_ptr of values indexed by keys.

```
#include <cache.hpp>
```

Collaboration diagram for trisycl::detail::cache< Key, Value >:



Public Types

- using key_type = Key

  The type of the keys used to indexed the cache.
- using value_type = Value

  The base type of the values stored in the cache.

Public Member Functions

- template<typename Functor >
  std::shared_ptr< value_type > get_or_register (const key_type &k, Functor &&create_element)

  Get a value stored in the cache if present or insert by calling a generator function.
- void remove (const key_type &k)

  Remove an entry from the cache.

Private Attributes

- std::unordered_map< key_type, std::weak_ptr< value_type > > c

  The caching storage.
- std::mutex m

  To make the cache thread-safe.

12.9.1 Detailed Description

template<typename Key, typename Value>
class trisycl::detail::cache< Key, Value >

A simple thread safe cache mechanism to cache std::shared_ptr of values indexed by keys.

Since internally only std::weak_ptr are stored, this does not prevent object deletion but it is up to the programmer not to use this cache to retrieve deleted objects.

Definition at line 27 of file cache.hpp.

12.9.2 Member Typedef Documentation

12.9.2.1 key_type

```
template<typename Key , typename Value >
using trisycl::detail::cache< Key, Value >::key_type = Key
```

The type of the keys used to indexed the cache.

Definition at line 32 of file cache.hpp.

12.9.2.2 value_type

```
template<typename Key , typename Value >
using trisycl::detail::cache< Key, Value >::value_type = Value
```

The base type of the values stored in the cache.

Definition at line 35 of file cache.hpp.

12.9.3 Member Function Documentation

12.9.3.1 get_or_register()

```
template<typename Key , typename Value >
template<typename Functor >
std::shared_ptr< value_type > trisycl::detail::cache< Key, Value >::get_or_register (
            const key_type & k,
            Functor && create_element )  [inline]
```

Get a value stored in the cache if present or insert by calling a generator function.

Parameters

in	*k*	is the key used to retrieve the value
in	*create_element*	is the function to be called if the key is not found in the cache to generate a value which is inserted for the key. This function has to produce a value convertible to a shared_ptr

Returns

a shared_ptr to the value retrieved or inserted

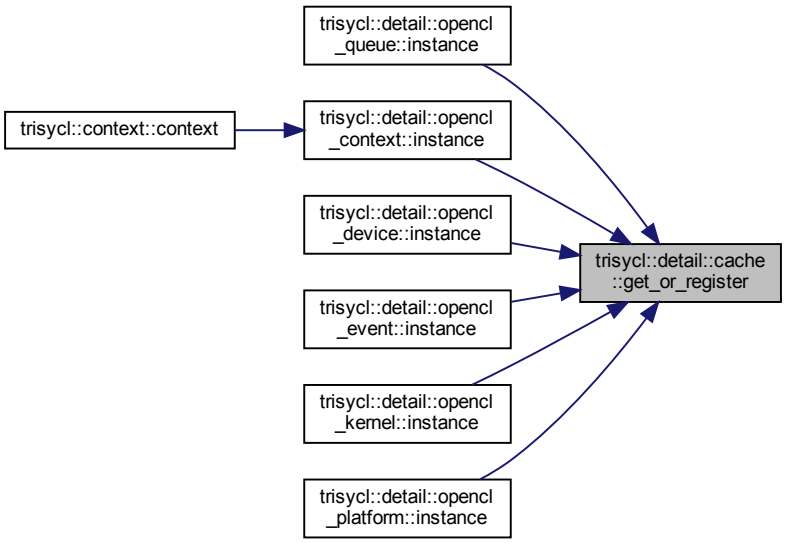
Definition at line 60 of file cache.hpp.

```
00061                                                                    {
00062     std::lock_guard<std::mutex> lg { m };
00063
00064     auto i = c.find(k);
00065     if (i != c.end())
00066       if (auto observe = i->second.lock())
00067         // Returns \c shared_ptr only if target object is still alive
00068         return observe;
00069
00070     // Otherwise create and insert a new element
00071     std::shared_ptr<value_type> e { create_element() };
00072     c.insert({ k, e });
00073     return e;
00074   }
```

References trisycl::detail::cache< Key, Value >::c, i, and trisycl::detail::cache< Key, Value >::m.

Referenced by trisycl::detail::opencl_queue::instance(), trisycl::detail::opencl_context::instance(), trisycl::detail::opencl_device::instanc
trisycl::detail::opencl_event::instance(), trisycl::detail::opencl_kernel::instance(), and trisycl::detail::opencl_platform::instance().

Here is the caller graph for this function:



12.9.3.2 remove()

```
template<typename Key , typename Value >
void trisycl::detail::cache< Key, Value >::remove (
            const key_type & k )  [inline]
```

Remove an entry from the cache.

Parameters

in	*k*	is the key associated to the value to remove from the cache

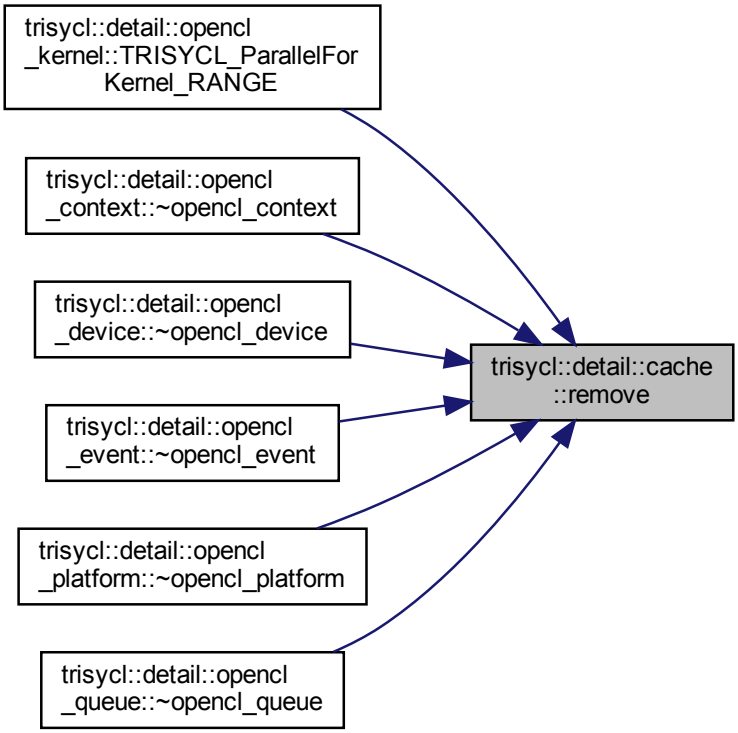
Definition at line 82 of file cache.hpp.

```
00082                                   {
00083     std::lock_guard<std::mutex> lg { m };
00084     c.erase(k);
00085   }
```

References trisycl::detail::cache< Key, Value >::c, and trisycl::detail::cache< Key, Value >::m.

Referenced by trisycl::detail::opencl_kernel::TRISYCL_ParallelForKernel_RANGE(), trisycl::detail::opencl_context::~opencl_context(), trisycl::detail::opencl_device::~opencl_device(), trisycl::detail::opencl_event::~opencl_event(), trisycl::detail::opencl_platform::~opencl_platform(), and trisycl::detail::opencl_queue::~opencl_queue().

Here is the caller graph for this function:



12.9.4 Member Data Documentation

12.9.4.1 c

```
template<typename Key , typename Value >
std::unordered_map<key_type, std::weak_ptr<value_type> > trisycl::detail::cache< Key, Value
>::c  [private]
```

The caching storage.

Definition at line 40 of file cache.hpp.

Referenced by trisycl::detail::cache< Key, Value >::get_or_register(), and trisycl::detail::cache< Key, Value >::remove().

12.9.4.2 m

```
template<typename Key , typename Value >
std::mutex trisycl::detail::cache< Key, Value >::m  [private]
```

To make the cache thread-safe.

Definition at line 43 of file cache.hpp.

Referenced by trisycl::detail::cache< Key, Value >::get_or_register(), and trisycl::detail::cache< Key, Value >::remove().

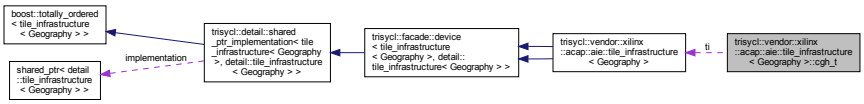
The documentation for this class was generated from the following file:

- include/triSYCL/detail/cache.hpp

12.10 trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t Struct Reference

```
#include <tile_infrastructure.hpp>
```

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t:



Public Member Functions

- cgh_t (tile_infrastructure t)
- template<typename Invocable >
  void single_task (Invocable &&t)
- cgh_t (tile_infrastructure t)
- template<typename Invocable >
  void single_task (Invocable &&t)

Public Attributes

- tile_infrastructure ti

12.10.1 Detailed Description

template<typename Geography>
struct trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t

Definition at line 100 of file tile_infrastructure.hpp.

12.10.2 Constructor & Destructor Documentation

12.10.2.1 cgh_t() [1/2]

```
template<typename Geography >
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t::cgh_t (
            tile_infrastructure t )  [inline]
```

Definition at line 102 of file tile_infrastructure.hpp.

```
00102 : ti { t } {}
```

12.10.2.2 cgh_t() [2/2]

```
template<typename Geography >
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t::cgh_t (
            tile_infrastructure t )  [inline]
```

Definition at line 102 of file tile_infrastructure.hpp.

```
00102 : ti { t } {}
```

12.10.3 Member Function Documentation

12.10.3.1 single_task() [1/2]

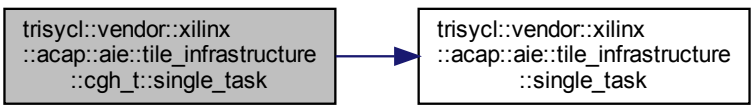
```
template<typename Geography >
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t::single_task
(
            Invocable && t )  [inline]
```

Definition at line 105 of file tile_infrastructure.hpp.

```
00105                                {
00106       ti.single_task([=] { t(); });
00107     }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::single_task(), and trisycl::vendor::xilinx::acap::aie::tile_

Here is the call graph for this function:



12.10.3.2 single_task() [2/2]

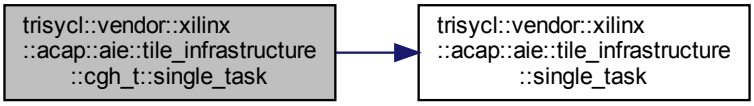
```
template<typename Geography >
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t::single_task
(
              Invocable && t )  [inline]
```

Definition at line 105 of file tile_infrastructure.hpp.

```
00105                                    {
00106       ti.single_task([=] { t(); });
00107     }
```

References trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::single_task(), and trisycl::vendor::xilinx::acap::aie::tile_

Here is the call graph for this function:



12.10.4 Member Data Documentation

12.10.4.1 ti

```
template<typename Geography >
tile_infrastructure trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >↩
::cgh_t::ti
```

Definition at line 101 of file tile_infrastructure.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t::single_task().

The documentation for this struct was generated from the following files:

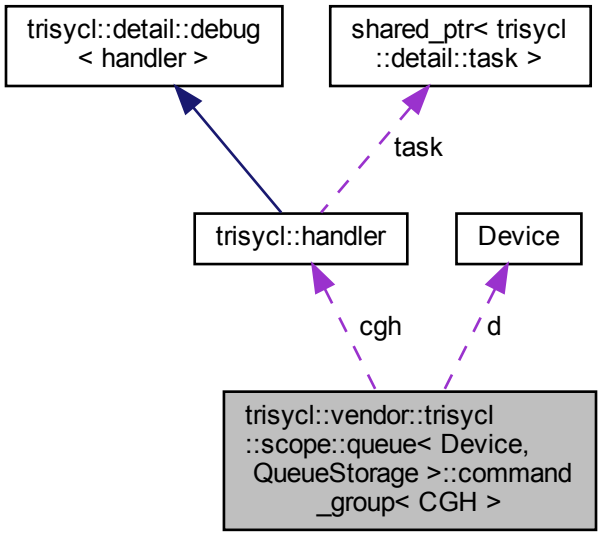
- include/sycl/vendor/Xilinx/acap/aie/tile_infrastructure.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/tile_infrastructure.hpp

12.11 trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH > Struct Template Reference

The command group redefining some member functions to be able to pass the scope and so on to the kernel.

`#include <queue.hpp>`

Collaboration diagram for trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >:



Public Member Functions

- command_group (::trisycl::handler &cgh, Device &d)
- operator::trisycl::handler & ()

  Add a conversion to `trisycl::handler&` so the usual methods such as `get_access` can still work.
- template<typename KernelName , typename Kernel >
  void single_task (Kernel k)

  Sequential kernel invocation.
- template<typename KernelName , int Rank, typename Kernel >
  void parallel_for (::trisycl::range< Rank > num_work_items, Kernel k)

  Parallel kernel invocation method on a range iteration space.
- auto & device_scope ()

  Provide access to the device scope from inside the kernel.
- template<typename PS = std::remove_reference_t<decltype(d.get_platform())>, typename E = std::enable_if_t<PS::has_some_↩ storage_p>>
  auto & platform_scope ()

  Provide access to the platform scope from inside the kernel, if any.
- command_group (::trisycl::handler &cgh, Device &d)
- operator::trisycl::handler & ()

Add a conversion to `trisycl::handler&` so the usual methods such as `get_access` can still work.

- template<typename KernelName , typename Kernel >
  void single_task (Kernel k)

  Sequential kernel invocation.
- template<typename KernelName , int Rank, typename Kernel >
  void parallel_for (::trisycl::range< Rank > num_work_items, Kernel k)

  Parallel kernel invocation method on a range iteration space.
- auto & device_scope ()

  Provide access to the device scope from inside the kernel.
- template<typename PS = std::remove_reference_t<decltype(d.get_platform())>, typename E = std::enable_if_t<PS::has_some_storage_p>>
  auto & platform_scope ()

  Provide access to the platform scope from inside the kernel, if any.

Public Attributes

- ::trisycl::handler & cgh

  The plain SYCL command-group handler.
- Device & d

12.11.1 Detailed Description

template<typename Device, typename QueueStorage = empty_queue_scope>
template<typename CGH>
struct trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >

The command group redefining some member functions to be able to pass the scope and so on to the kernel.

Definition at line 150 of file queue.hpp.

12.11.2 Constructor & Destructor Documentation

12.11.2.1 command_group() [1/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >::command_group (
            ::trisycl::handler & cgh,
            Device & d )  [inline]
```

Definition at line 156 of file queue.hpp.

```
00157       :  cgh { cgh }, d { d } {}
```

12.11.2.2 command_group() [2/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >::command↩
_group (
            ::trisycl::handler & cgh,
            Device & d )  [inline]
```

Definition at line 156 of file queue.hpp.

```
00157       :  cgh { cgh }, d { d } {}
```

12.11.3 Member Function Documentation

12.11.3.1 device_scope() [1/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >↩
::device_scope ( )  [inline]
```

Provide access to the device scope from inside the kernel.

Definition at line 188 of file queue.hpp.

```
00188 { return d.get_storage(); }
```

12.11.3.2 device_scope() [2/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >↩
::device_scope ( )  [inline]
```

Provide access to the device scope from inside the kernel.

Definition at line 188 of file queue.hpp.

```
00188 { return d.get_storage(); }
```

12.11.3.3 operator::trisycl::handler &() [1/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >::operator↩
::trisycl::handler & ( )  [inline]
```

Add a conversion to `trisycl::handler&` so the usual methods such as `get_access` can still work.

Definition at line 162 of file queue.hpp.

```
00162                                {
00163       return cgh;
00164     }
```

12.11.3.4 operator::trisycl::handler &() [2/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >::operator↩
::trisycl::handler & ( )  [inline]
```

Add a conversion to `trisycl::handler&` so the usual methods such as `get_access` can still work.

Definition at line 162 of file queue.hpp.

```
00162                                {
00163       return cgh;
00164     }
```

12.11.3.5 parallel_for() [1/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
template<typename KernelName , int Rank, typename Kernel >
void trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >↩
::parallel_for (
             ::trisycl::range< Rank > num_work_items,
             Kernel k )  [inline]
```

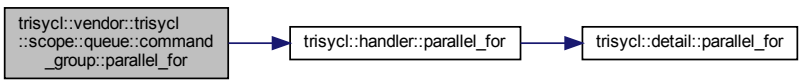
Parallel kernel invocation method on a range iteration space.

Definition at line 178 of file queue.hpp.

```
00179                                {
00180       cgh.parallel_for<KernelName>(num_work_items,
00181                                    [=, *this] (::trisycl::id<Rank> i) mutable {
00182                                      k(i, *this);
00183                                    });
00184     }
```

References i, and trisycl::handler::parallel_for().

Here is the call graph for this function:



12.11.3.6 parallel_for() [2/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
template<typename KernelName , int Rank, typename Kernel >
void trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >↩
::parallel_for (
             ::trisycl::range< Rank > num_work_items,
             Kernel k )  [inline]
```

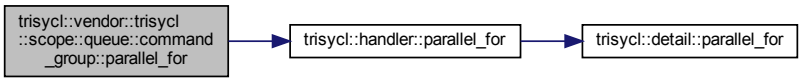
Parallel kernel invocation method on a range iteration space.

Definition at line 178 of file queue.hpp.

```
00179                                {
00180       cgh.parallel_for<KernelName>(num_work_items,
00181                                    [=, *this] (::trisycl::id<Rank> i) mutable {
00182                                      k(i, *this);
00183                                    });
00184     }
```

References i, and trisycl::handler::parallel_for().

Here is the call graph for this function:



12.11.3.7 platform_scope() [1/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
template<typename PS = std::remove_reference_t<decltype(d.get_platform())>, typename E =
std::enable_if_t<PS::has_some_storage_p>>
auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >↩
::platform_scope ( )  [inline]
```

Provide access to the platform scope from inside the kernel, if any.

Definition at line 194 of file queue.hpp.

```
00194 { return d.get_platform().get_storage(); }
```

12.11.3.8 platform_scope() [2/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
template<typename PS = std::remove_reference_t<decltype(d.get_platform())>, typename E =
std::enable_if_t<PS::has_some_storage_p>>
auto & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >↩
::platform_scope ( )  [inline]
```

Provide access to the platform scope from inside the kernel, if any.

Definition at line 194 of file queue.hpp.

```
00194 { return d.get_platform().get_storage(); }
```

12.11.3.9 single_task() [1/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
template<typename KernelName , typename Kernel >
void trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >↩
::single_task (
             Kernel k )  [inline]
```

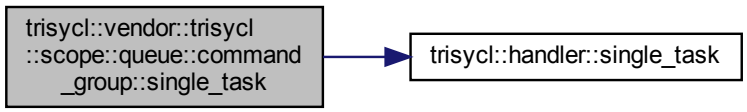
Sequential kernel invocation.

Definition at line 169 of file queue.hpp.

```
00169                                {
00170       cgh.single_task<KernelName>([=, *this] () mutable {
00171           k(*this);
00172         });
00173     }
```

References trisycl::handler::single_task().

Here is the call graph for this function:



12.11.3.10 single_task() [2/2]

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
template<typename KernelName , typename Kernel >
void trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >↩
::single_task (
             Kernel k )  [inline]
```

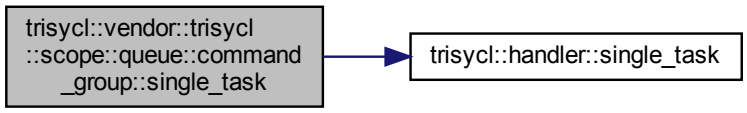
Sequential kernel invocation.

Definition at line 169 of file queue.hpp.

```
00169                                {
00170       cgh.single_task<KernelName>([=, *this] () mutable {
00171           k(*this);
00172         });
00173     }
```

References trisycl::handler::single_task().

Here is the call graph for this function:



12.11.4 Member Data Documentation

12.11.4.1 cgh

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
trisycl::handler & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group<
CGH >::cgh
```

The plain SYCL command-group handler.

Definition at line 153 of file queue.hpp.

12.11.4.2 d

```
template<typename Device , typename QueueStorage = empty_queue_scope>
template<typename CGH >
Device & trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH
>::d
```

Definition at line 154 of file queue.hpp.

The documentation for this struct was generated from the following files:

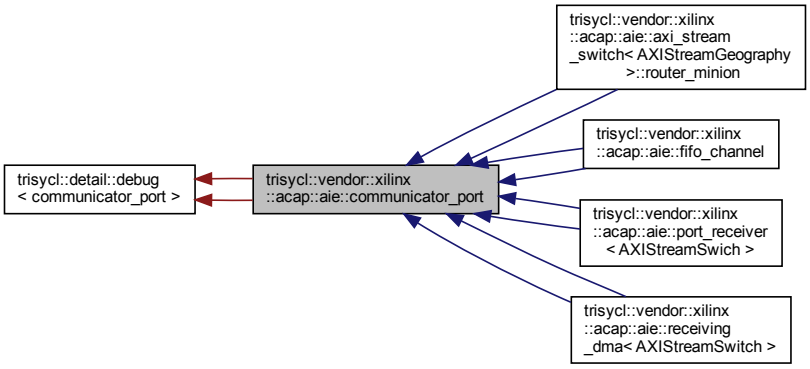
- include/sycl/vendor/triSYCL/scope/queue.hpp
- include/triSYCL/vendor/triSYCL/scope/queue.hpp

12.12 trisycl::vendor::xilinx::acap::aie::communicator_port Class Reference

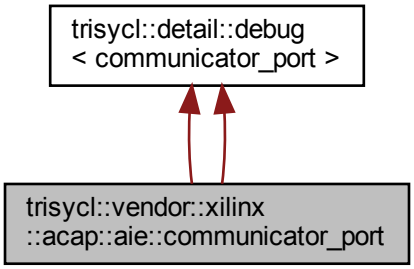
Abstract interface for a communication port.

`#include <connection.hpp>`

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::communicator_port:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::communicator_port:



Public Member Functions

- virtual void write (const axi_packet &)=0

  Enqueue a packet on the communicator input.
- auto & operator<< (const axi_packet::value_type &v)

  Alias to enqueue a packet on the communicator input.
- virtual bool try_write (const axi_packet &)=0

  Try to enqueue a packet to input.
- virtual axi_packet::value_type read ()=0

  Waiting read to a core input port.
- virtual bool try_read (axi_packet::value_type &)=0

  Non-blocking read from a communicator output port.
- template<typename T >
  auto & operator>> (T &v)

  Alias to the waiting read to a communicator output port.
- virtual ~communicator_port ()=default

  Nothing specific to do but need a virtual destructor to avoid slicing.
- virtual void write (const axi_packet &)=0

  Enqueue a packet on the communicator input.
- auto & operator<< (const axi_packet::value_type &v)

  Alias to enqueue a packet on the communicator input.
- virtual bool try_write (const axi_packet &)=0

  Try to enqueue a packet to input.
- virtual axi_packet::value_type read ()=0

  Waiting read to a core input port.
- virtual bool try_read (axi_packet::value_type &)=0

  Non-blocking read from a communicator output port.
- template<typename T >
  auto & operator>> (T &v)

  Alias to the waiting read to a communicator output port.
- virtual ~communicator_port ()=default

  Nothing specific to do but need a virtual destructor to avoid slicing.

Additional Inherited Members

12.12.1 Detailed Description

Abstract interface for a communication port.

For example a router output is actually implemented as an input to some other consumer (router, core...).

Definition at line 67 of file connection.hpp.

12.12.2 Constructor & Destructor Documentation

12.12.2.1 ~communicator_port() [1/2]

```
virtual trisycl::vendor::xilinx::acap::aie::communicator_port::~communicator_port ( )  [virtual],
[default]
```

Nothing specific to do but need a virtual destructor to avoid slicing.

12.12.2.2 ~communicator_port() [2/2]

```
virtual trisycl::vendor::xilinx::acap::aie::communicator_port::~communicator_port ( )  [virtual],
[default]
```

Nothing specific to do but need a virtual destructor to avoid slicing.

12.12.3 Member Function Documentation

12.12.3.1 operator<<() [1/2]

```
auto & trisycl::vendor::xilinx::acap::aie::communicator_port::operator<< (
            const axi_packet::value_type & v )  [inline]
```

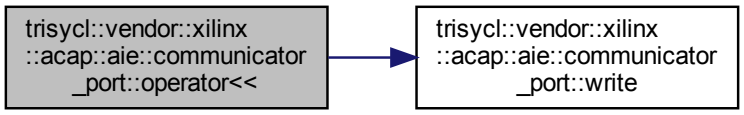
Alias to enqueue a packet on the communicator input.

Definition at line 76 of file connection.hpp.

```
00076                                                {
00077     write(v);
00078     return *this;
00079   }
```

References write().

Here is the call graph for this function:



12.12.3.2 operator<<() [2/2]

```
auto & trisycl::vendor::xilinx::acap::aie::communicator_port::operator<< (
            const axi_packet::value_type & v )  [inline]
```

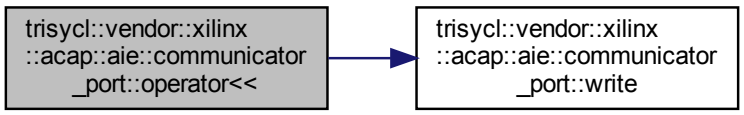
Alias to enqueue a packet on the communicator input.

Definition at line 76 of file connection.hpp.

```
00076                                                {
00077     write(v);
00078     return *this;
00079   }
```

References write().

Here is the call graph for this function:



12.12.3.3 operator>>() [1/2]

```
template<typename T >
auto & trisycl::vendor::xilinx::acap::aie::communicator_port::operator>> (
            T & v )  [inline]
```

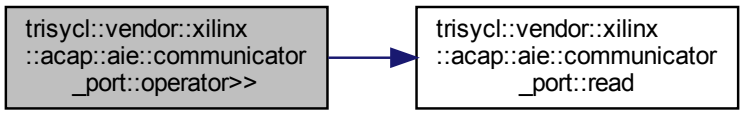
Alias to the waiting read to a communicator output port.

Definition at line 102 of file connection.hpp.

```
00102                                {
00103     v = static_cast<T>(read());
00104     return *this;
00105   }
```

References read().

Here is the call graph for this function:



12.12.3.4 operator>>() [2/2]

```
template<typename T >
auto & trisycl::vendor::xilinx::acap::aie::communicator_port::operator>> (
            T & v )  [inline]
```

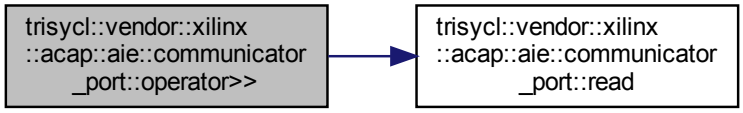
Alias to the waiting read to a communicator output port.

Definition at line 102 of file connection.hpp.

```
00102                           {
00103     v = static_cast<T>(read());
00104     return *this;
00105   }
```

References read().

Here is the call graph for this function:



12.12.3.5 read() [1/2]

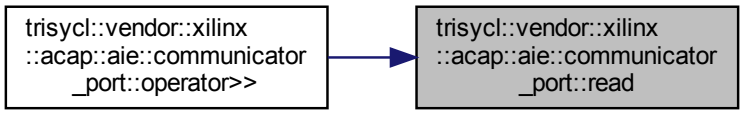
```
virtual axi_packet::value_type trisycl::vendor::xilinx::acap::aie::communicator_port::read ( )
[pure virtual]
```

Waiting read to a core input port.

Implemented in trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, trisycl::vendor::xilinx::acap::aie::fifo_channel, trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStrea trisycl::vendor::xilinx::acap::aie::fifo_channel, trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, and trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >.

Referenced by operator>>().

Here is the caller graph for this function:



12.12.3.6 read() [2/2]

```
virtual axi_packet::value_type trisycl::vendor::xilinx::acap::aie::communicator_port::read ( )
[pure virtual]
```

Waiting read to a core input port.

Implemented in trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, trisycl::vendor::xilinx::acap::aie::fifo_channel, trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStrea trisycl::vendor::xilinx::acap::aie::fifo_channel, trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, and trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >.

12.12.3.7 try_read() [1/2]

```
virtual bool trisycl::vendor::xilinx::acap::aie::communicator_port::try_read (
            axi_packet::value_type &  ) [pure virtual]
```

Non-blocking read from a communicator output port.

Returns

true if the value was correctly read

Implemented in trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, trisycl::vendor::xilinx::acap::aie::receiving_dma trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, trisycl::vendor::xilinx::acap::aie::fifo_ch trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStream trisycl::vendor::xilinx::acap::aie::fifo_channel, and trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >.

12.12.3.8 try_read() [2/2]

```
virtual bool trisycl::vendor::xilinx::acap::aie::communicator_port::try_read (
            axi_packet::value_type &  ) [pure virtual]
```

Non-blocking read from a communicator output port.

Returns

true if the value was correctly read

Implemented in trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, trisycl::vendor::xilinx::acap::aie::receiving_dma trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, trisycl::vendor::xilinx::acap::aie::fifo_ch trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStream trisycl::vendor::xilinx::acap::aie::fifo_channel, and trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >.

12.12.3.9 try_write() [1/2]

```
virtual bool trisycl::vendor::xilinx::acap::aie::communicator_port::try_write (
            const axi_packet &  )  [pure virtual]
```

Try to enqueue a packet to input.

Returns

true if the packet is correctly enqueued

Implemented in trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, trisycl::vendor::xilinx::acap::aie::fifo_channel, trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStrea trisycl::vendor::xilinx::acap::aie::fifo_channel, trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, and trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >.

12.12.3.10 try_write() [2/2]

```
virtual bool trisycl::vendor::xilinx::acap::aie::communicator_port::try_write (
            const axi_packet &  )  [pure virtual]
```

Try to enqueue a packet to input.

Returns

true if the packet is correctly enqueued

Implemented in trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, trisycl::vendor::xilinx::acap::aie::fifo_channel, trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStrea trisycl::vendor::xilinx::acap::aie::fifo_channel, trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, and trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >.

12.12.3.11 write() [1/2]

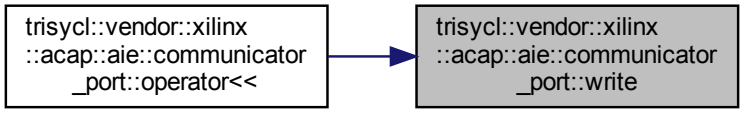
```
virtual void trisycl::vendor::xilinx::acap::aie::communicator_port::write (
            const axi_packet &  )  [pure virtual]
```

Enqueue a packet on the communicator input.

Implemented in trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, trisycl::vendor::xilinx::acap::aie::fifo_channel, trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStrea trisycl::vendor::xilinx::acap::aie::fifo_channel, trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, and trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >.

Referenced by operator<<().

Here is the caller graph for this function:



12.12.3.12 write() [2/2]

```
virtual void trisycl::vendor::xilinx::acap::aie::communicator_port::write (
            const axi_packet &  )  [pure virtual]
```

Enqueue a packet on the communicator input.

Implemented in trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, trisycl::vendor::xilinx::acap::aie::fifo_channel, trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStrea trisycl::vendor::xilinx::acap::aie::fifo_channel, trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, and trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >.

The documentation for this class was generated from the following files:

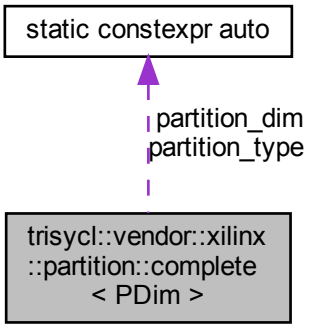
- include/sycl/vendor/Xilinx/acap/aie/connection.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/connection.hpp

12.13 trisycl::vendor::xilinx::partition::complete< PDim > Struct Template Reference

Represent a complete partition.

```
#include <partition_array.hpp>
```

Collaboration diagram for trisycl::vendor::xilinx::partition::complete< PDim >:



Static Public Attributes

- static constexpr auto partition_dim = PDim
- static constexpr auto partition_type = type::complete

12.13.1 Detailed Description

template<std::size_t PDim = 1>
struct trisycl::vendor::xilinx::partition::complete< PDim >

Represent a complete partition.

The single array would be partitioned into individual elements. That is if we have a 4-element array with one dimension, the array is completely partitioned into distributed RAM or 4 independent registers.

Parameters

PDim	is the dimension that user wants to apply complete partition on. If PDim is 0, all dimensions will be completely partitioned.

Todo Deal with multi-dimension array. Now, since we can only deal with 1-dim, PDim is set to 1 by default.

The single array would be partitioned into individual elements. That is if we have a 4-element array with one dimension, the array is completely partitioned into distributed RAM or 4 independent registers.

Parameters

PDim	is the dimension that user wants to apply complete partition on. If PDim is 0, all dimensions will be completely partitioned.

Todo Deal with multi-dimension array. Now, since we can only deal with 1-dim, PDim is set to 1 by default.

Definition at line 119 of file partition_array.hpp.

12.13.2 Member Data Documentation

12.13.2.1 partition_dim

```
template<std::size_t PDim = 1>
static constexpr auto trisycl::vendor::xilinx::partition::complete< PDim >::partition_dim =
PDim  [static], [constexpr]
```

Definition at line 120 of file partition_array.hpp.

12.13.2.2 partition_type

```
template<std::size_t PDim = 1>
static constexpr auto trisycl::vendor::xilinx::partition::complete< PDim >::partition_type =
type::complete  [static], [constexpr]
```

Definition at line 121 of file partition_array.hpp.

The documentation for this struct was generated from the following files:

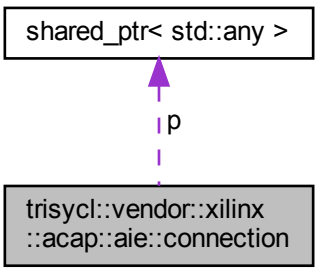
- include/sycl/vendor/Xilinx/fpga/partition_array.hpp
- include/triSYCL/vendor/Xilinx/fpga/partition_array.hpp

12.14 trisycl::vendor::xilinx::acap::aie::connection Struct Reference

A connection between some AXI Stream Switch.

```
#include <connection.hpp>
```

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::connection:



Classes

- struct base

  Base class of connection port.
- struct input

  Abstract an input port.
- struct output

  Abstract an output port.

Public Member Functions

- template<typename Pipe >
  connection (Pipe &&pipe)

  Create a connection.
- auto in ()

  Get the input port of this connection.
- auto out ()

  Get the output port of this connection.
- template<typename Pipe >
  connection (Pipe &&pipe)

  Create a connection.
- auto in ()

  Get the input port of this connection.
- auto out ()

  Get the output port of this connection.

Public Attributes

- std::shared_ptr< std::any > p

  The concrete communication media with some type erasure.

12.14.1 Detailed Description

A connection between some AXI Stream Switch.

Definition at line 260 of file connection.hpp.

12.14.2 Constructor & Destructor Documentation

12.14.2.1 connection() [1/2]

```
template<typename Pipe >
trisycl::vendor::xilinx::acap::aie::connection::connection (
            Pipe && pipe )  [inline]
```

Create a connection.

Parameters

in	*pipe*	is the communication media to be used

Definition at line 269 of file connection.hpp.

```
00270     :  p { new std::any { std::forward<Pipe>(pipe) } } {}
```

12.14.2.2 connection() [2/2]

```
template<typename Pipe >
trisycl::vendor::xilinx::acap::aie::connection::connection (
            Pipe && pipe )  [inline]
```

Create a connection.

Parameters

in	*pipe*	is the communication media to be used

Definition at line 269 of file connection.hpp.

```
00270     :  p { new std::any { std::forward<Pipe>(pipe) } } {}
```

12.14.3 Member Function Documentation

12.14.3.1 in() [1/2]

```
auto trisycl::vendor::xilinx::acap::aie::connection::in ( )  [inline]
```

Get the input port of this connection.

Definition at line 274 of file connection.hpp.

```
00274             {
00275     return input { *this };
00276   }
```

12.14.3.2 in() [2/2]

```
auto trisycl::vendor::xilinx::acap::aie::connection::in ( )  [inline]
```

Get the input port of this connection.

Definition at line 274 of file connection.hpp.

```
00274             {
00275     return input { *this };
00276   }
```

12.14.3.3 out() [1/2]

```
auto trisycl::vendor::xilinx::acap::aie::connection::out ( )  [inline]
```

Get the output port of this connection.

Definition at line 280 of file connection.hpp.

```
00280             {
00281     return output { *this };
00282   }
```

12.14.3.4 out() [2/2]

```
auto trisycl::vendor::xilinx::acap::aie::connection::out ( )  [inline]
```

Get the output port of this connection.

Definition at line 280 of file connection.hpp.

```
00280             {
00281     return output { *this };
00282   }
```

12.14.4 Member Data Documentation

12.14.4.1 p

```
std::shared_ptr< std::any > trisycl::vendor::xilinx::acap::aie::connection::p
```

The concrete communication media with some type erasure.

Definition at line 262 of file connection.hpp.

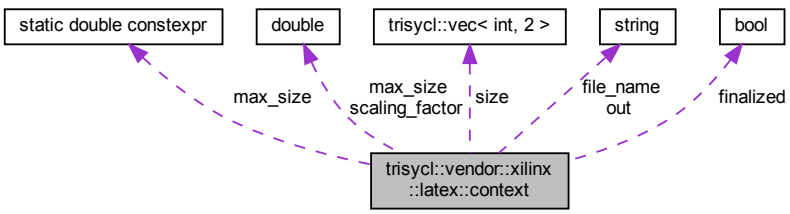
The documentation for this struct was generated from the following files:

- include/sycl/vendor/Xilinx/acap/aie/connection.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/connection.hpp

12.15 trisycl::vendor::xilinx::latex::context Class Reference

```
#include <latex.hpp>
```

Collaboration diagram for trisycl::vendor::xilinx::latex::context:



Public Member Functions

- context (const vec< int, 2 > &s, const std::string &f_name={})

  Create a graphics context.
- void add (const std::string &content)

  Add textual LaTeX content to the context.
- double scale (double v) const
- void display ()

  Display the content.
- context (const vec< int, 2 > &s, const std::string &f_name={})

  Create a graphics context.
- void add (const std::string &content)

  Add textual LaTeX content to the context.
- double scale (double v) const
- void display ()

  Display the content.

Static Public Member Functions

- static std::string clean_node (const std::string_view &node_name, const std::string_view &prefix="")

  Build a TikZ/PGF valid name string by concatenating a prefix and a capitalized node name after having removed any '_'.
- static std::string clean_node (const std::string_view &node_name, const std::string_view &prefix="")

  Build a TikZ/PGF valid name string by concatenating a prefix and a capitalized node name after having removed any '_'.

Public Attributes

- double scaling_factor

  Scale all the coordinates by this factor to be under the TeX max size.
- vec< int, 2 > size

  Size of the drawing in the original units (unscaled)
- std::string file_name

  Name of the file to write into.
- std::string out

  Store the LaTeX output.
- bool finalized = false

  Keep track whether the content is finalized for final user consumption.

Static Public Attributes

- static double constexpr max_size = 574

Private Member Functions

- void finalize ()

  Finalize the content before using it.
- void finalize ()

  Finalize the content before using it.

12.15.1 Detailed Description

Definition at line 32 of file latex.hpp.

12.15.2 Constructor & Destructor Documentation

12.15.2.1 context() [1/2]

```
trisycl::vendor::xilinx::latex::context::context (
            const vec< int, 2 > & s,
            const std::string & f_name = {} )  [inline]
```

Create a graphics context.

Parameters

in	*s*	is the 2D size of the drawing to be generated in mm
in	*f_name*	is the file name of the LaTeX to be generated into or standard output by default

Definition at line 61 of file latex.hpp.

```
00061                                                       {})
00062     :  size { s }, file_name { f_name } {
00063     // Compute the scaling factor to remain under the TeX limit
00064     auto m = std::max(size.x(), size.y());
00065     scaling_factor = m > max_size ?  max_size/m :  1;
00066     // Generate the LaTeX header
00067     out = (boost::format { R"(%% To be compiled with lualatex instead of pdflatex
00068 %% to avoid a bug on _ and to handle huge memory automatically.
```

12.15.2.2 context() [2/2]

```
trisycl::vendor::xilinx::latex::context::context (
            const vec< int, 2 > & s,
            const std::string & f_name = {} )  [inline]
```

Create a graphics context.

Parameters

in	*s*	is the 2D size of the drawing to be generated in mm
in	*f_name*	is the file name of the LaTeX to be generated into or standard output by default

Definition at line 61 of file latex.hpp.

```
00061                                                       {})
00062     :  size { s }, file_name { f_name } {
00063     // Compute the scaling factor to remain under the TeX limit
00064     auto m = std::max(size.x(), size.y());
00065     scaling_factor = m > max_size ?  max_size/m :  1;
00066     // Generate the LaTeX header
```

```
00067     out = (boost::format { R"(%% To be compiled with lualatex instead of pdflatex
00068 %% to avoid a bug on _ and to handle huge memory automatically.
```

12.15.3 Member Function Documentation

12.15.3.1 add() [1/2]

```
void trisycl::vendor::xilinx::latex::context::add (
             const std::string & content )  [inline]
```

Add textual LaTeX content to the context.

Definition at line 72 of file latex.hpp.

```
00072                     {geometry}
00073 %% Use a font allowing arbitrary size
00074 \usepackage{lmodern}
```

12.15.3.2 add() [2/2]

```
void trisycl::vendor::xilinx::latex::context::add (
             const std::string & content )  [inline]
```

Add textual LaTeX content to the context.

Definition at line 72 of file latex.hpp.

```
00072                     {geometry}
00073 %% Use a font allowing arbitrary size
00074 \usepackage{lmodern}
```

12.15.3.3 clean_node() [1/2]

```
static std::string trisycl::vendor::xilinx::latex::context::clean_node (
             const std::string_view & node_name,
             const std::string_view & prefix = "" )  [inline], [static]
```

Build a TikZ/PGF valid name string by concatenating a prefix and a capitalized node name after having removed any '_'.

\input[in] node_name is a string-like TikZ/PGF that may contain some '_'

\input[in] prefix is an optional string-like prefix

Definition at line 118 of file latex.hpp.

```
00120                   {
00121     if (!finalized) {
00122       // Add the LaTeX epilogue
00123       out += R"(
00124 \end{tikzpicture}
00125
00126 \end{document})";
00127 }
00128 }
00129
00130 public:
```

References finalized, and out.

12.15.3.4 clean_node() [2/2]

```
static std::string trisycl::vendor::xilinx::latex::context::clean_node (
            const std::string_view & node_name,
            const std::string_view & prefix = "" )  [inline], [static]
```

Build a TikZ/PGF valid name string by concatenating a prefix and a capitalized node name after having removed any '_'.

\input[in] node_name is a string-like TikZ/PGF that may contain some '_'

\input[in] prefix is an optional string-like prefix

Definition at line 118 of file latex.hpp.

```
00120                 {
00121     if (!finalized) {
00122       // Add the LaTeX epilogue
00123       out += R"(
00124 \end{tikzpicture}
00125
00126 \end{document})";
00127 }
00128 }
00129
00130 public:
```

References finalized, and out.

12.15.3.5 display() [1/2]

```
void trisycl::vendor::xilinx::latex::context::display ( )  [inline]
```

Display the content.

Display the LaTeX textual content on or into a file, to be compiled for example with lualatex

Definition at line 102 of file latex.hpp.

```
00107                                 {
00108     out += content;
```

12.15.3.6 display() [2/2]

```
void trisycl::vendor::xilinx::latex::context::display ( )  [inline]
```

Display the content.

Display the LaTeX textual content on or into a file, to be compiled for example with lualatex

Definition at line 102 of file latex.hpp.

```
00107                                 {
00108     out += content;
```

12.15.3.7 finalize() [1/2]

```
void trisycl::vendor::xilinx::latex::context::finalize ( )  [inline], [private]
```

Finalize the content before using it.

Definition at line 85 of file latex.hpp.

```
00086       {document}
00087 %% No page number, header or footer
00088 \thispagestyle{empty}
00089 %% Skip the usual space at the beginning of a paragraph
00090 \noindent
00091 %% Use a super small font.  Use sans-serif for readability
00092 \fontsize{%3%}{%3%}\selectfont\sffamily
00093 %% Use remembering in every picture so we can use named coordinates
```

12.15.3.8 finalize() [2/2]

```
void trisycl::vendor::xilinx::latex::context::finalize ( )  [inline], [private]
```

Finalize the content before using it.

Definition at line 85 of file latex.hpp.

```
00086       {document}
00087 %% No page number, header or footer
00088 \thispagestyle{empty}
00089 %% Skip the usual space at the beginning of a paragraph
00090 \noindent
00091 %% Use a super small font.  Use sans-serif for readability
00092 \fontsize{%3%}{%3%}\selectfont\sffamily
00093 %% Use remembering in every picture so we can use named coordinates
```

12.15.3.9 scale() [1/2]

```
double trisycl::vendor::xilinx::latex::context::scale (
            double v ) const  [inline]
```

Definition at line 78 of file latex.hpp.

```
00078           {tikzlings}
00079 \definecolor{orangeSYCL}{RGB}{242,104,34}
00080 %% Some cool palettes
```

12.15.3.10 scale() [2/2]

```
double trisycl::vendor::xilinx::latex::context::scale (
            double v ) const  [inline]
```

Definition at line 78 of file latex.hpp.

```
00078           {tikzlings}
00079 \definecolor{orangeSYCL}{RGB}{242,104,34}
00080 %% Some cool palettes
```

12.15.4 Member Data Documentation

12.15.4.1 file_name

```
std::string trisycl::vendor::xilinx::latex::context::file_name
```

Name of the file to write into.

Definition at line 46 of file latex.hpp.

12.15.4.2 finalized

```
bool trisycl::vendor::xilinx::latex::context::finalized = false
```

Keep track whether the content is finalized for final user consumption.

Definition at line 52 of file latex.hpp.

Referenced by clean_node().

12.15.4.3 max_size

```
double constexpr trisycl::vendor::xilinx::latex::context::max_size = 574  [static], [constexpr]
```

Definition at line 37 of file latex.hpp.

12.15.4.4 out

```
std::string trisycl::vendor::xilinx::latex::context::out
```

Store the LaTeX output.

Definition at line 49 of file latex.hpp.

Referenced by clean_node().

12.15.4.5 scaling_factor

```
double trisycl::vendor::xilinx::latex::context::scaling_factor
```

Scale all the coordinates by this factor to be under the TeX max size.

Definition at line 40 of file latex.hpp.

12.15.4.6 size

```
vec<int, 2> trisycl::vendor::xilinx::latex::context::size
```

Size of the drawing in the original units (unscaled)

Definition at line 43 of file latex.hpp.

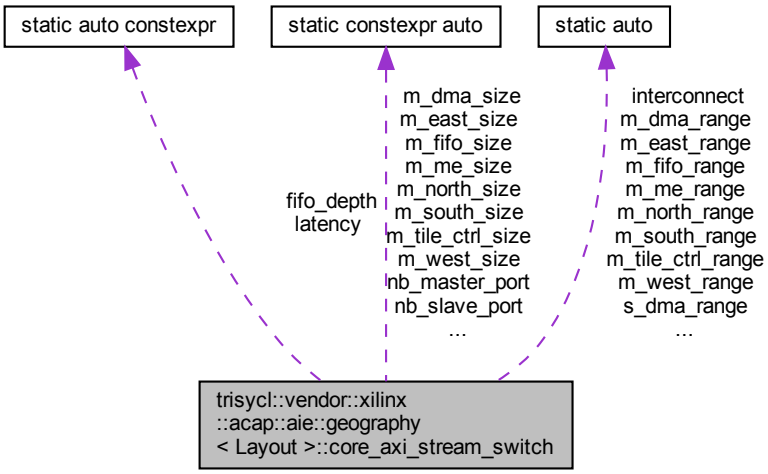
The documentation for this class was generated from the following files:

- include/sycl/vendor/Xilinx/latex.hpp
- include/triSYCL/vendor/Xilinx/latex.hpp

12.16 trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch Struct Reference

```
#include <geography.hpp>
```

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch:



Public Types

- enum class master_port_layout : std::int8_t {
  me_0 , me_1 , me_last = me_1 , dma_0 ,
  dma_1 , dma_last = dma_1 , tile_ctrl , tile_ctrl_last = tile_ctrl ,
  fifo_0 , fifo_1 , fifo_last = fifo_1 , south_0 ,
  south_1 , south_2 , south_3 , south_last = south_3 ,
  west_0 , west_1 , west_2 , west_3 ,
  west_last = west_3 , north_0 , north_1 , north_2 ,
  north_3 , north_4 , north_5 , north_last = north_5 ,
  east_0 , east_1 , east_2 , east_3 ,
  east_last = east_3 , size , me_0 , me_1 ,
  me_last = me_1 , dma_0 , dma_1 , dma_last = dma_1 ,
  tile_ctrl , tile_ctrl_last = tile_ctrl , fifo_0 , fifo_1 ,
  fifo_last = fifo_1 , south_0 , south_1 , south_2 ,
  south_3 , south_last = south_3 , west_0 , west_1 ,
  west_2 , west_3 , west_last = west_3 , north_0 ,
  north_1 , north_2 , north_3 , north_4 ,
  north_5 , north_last = north_5 , east_0 , east_1 ,
  east_2 , east_3 , east_last = east_3 , size }

  Layout of the AXI stream master ports in the switch.
- enum class slave_port_layout : std::int8_t {
  me_0 , me_1 , me_last = me_1 , dma_0 ,
  dma_1 , dma_last = dma_1 , tile_ctrl , tile_ctrl_last = tile_ctrl ,
  fifo_0 , fifo_1 , fifo_last = fifo_1 , south_0 ,
  south_1 , south_2 , south_3 , south_4 ,
  south_5 , south_last = south_5 , west_0 , west_1 ,
  west_2 , west_3 , west_last = west_3 , north_0 ,
  north_1 , north_2 , north_3 , north_last = north_3 ,
  east_0 , east_1 , east_2 , east_3 ,
  east_last = east_3 , core_trace , mem_trace , size ,
  me_0 , me_1 , me_last = me_1 , dma_0 ,
  dma_1 , dma_last = dma_1 , tile_ctrl , tile_ctrl_last = tile_ctrl ,
  fifo_0 , fifo_1 , fifo_last = fifo_1 , south_0 ,
  south_1 , south_2 , south_3 , south_4 ,
  south_5 , south_last = south_5 , west_0 , west_1 ,
  west_2 , west_3 , west_last = west_3 , north_0 ,
  north_1 , north_2 , north_3 , north_last = north_3 ,
  east_0 , east_1 , east_2 , east_3 ,
  east_last = east_3 , core_trace , mem_trace , size }

  Layout of the AXI stream slave ports in the switch.
- enum class master_port_layout : std::int8_t {
  me_0 , me_1 , me_last = me_1 , dma_0 ,
  dma_1 , dma_last = dma_1 , tile_ctrl , tile_ctrl_last = tile_ctrl ,
  fifo_0 , fifo_1 , fifo_last = fifo_1 , south_0 ,
  south_1 , south_2 , south_3 , south_last = south_3 ,
  west_0 , west_1 , west_2 , west_3 ,
  west_last = west_3 , north_0 , north_1 , north_2 ,
  north_3 , north_4 , north_5 , north_last = north_5 ,
  east_0 , east_1 , east_2 , east_3 ,
  east_last = east_3 , size , me_0 , me_1 ,
  me_last = me_1 , dma_0 , dma_1 , dma_last = dma_1 ,
  tile_ctrl , tile_ctrl_last = tile_ctrl , fifo_0 , fifo_1 ,
  fifo_last = fifo_1 , south_0 , south_1 , south_2 ,
  south_3 , south_last = south_3 , west_0 , west_1 ,
  west_2 , west_3 , west_last = west_3 , north_0 ,
  north_1 , north_2 , north_3 , north_4 ,

north_5 , north_last = north_5 , east_0 , east_1 ,
east_2 , east_3 , east_last = east_3 , size }

Layout of the AXI stream master ports in the switch.

- enum class slave_port_layout : std::int8_t {
me_0 , me_1 , me_last = me_1 , dma_0 ,
dma_1 , dma_last = dma_1 , tile_ctrl , tile_ctrl_last = tile_ctrl ,
fifo_0 , fifo_1 , fifo_last = fifo_1 , south_0 ,
south_1 , south_2 , south_3 , south_4 ,
south_5 , south_last = south_5 , west_0 , west_1 ,
west_2 , west_3 , west_last = west_3 , north_0 ,
north_1 , north_2 , north_3 , north_last = north_3 ,
east_0 , east_1 , east_2 , east_3 ,
east_last = east_3 , core_trace , mem_trace , size ,
me_0 , me_1 , me_last = me_1 , dma_0 ,
dma_1 , dma_last = dma_1 , tile_ctrl , tile_ctrl_last = tile_ctrl ,
fifo_0 , fifo_1 , fifo_last = fifo_1 , south_0 ,
south_1 , south_2 , south_3 , south_4 ,
south_5 , south_last = south_5 , west_0 , west_1 ,
west_2 , west_3 , west_last = west_3 , north_0 ,
north_1 , north_2 , north_3 , north_last = north_3 ,
east_0 , east_1 , east_2 , east_3 ,
east_last = east_3 , core_trace , mem_trace , size }

Layout of the AXI stream slave ports in the switch.

Static Public Attributes

- static auto constexpr latency = 4

Default number of registers on default router paths.

- static auto constexpr fifo_depth = 16

Depth of additional FIFO on specific FIFO paths.

- static constexpr auto nb_master_port

Number of AXI stream master ports in the switch.

- static auto m_me_range

A range of the master ports connected to the tile inputs.

- static constexpr auto m_me_size

Number of master ports connected to the tile inputs.

- static auto m_dma_range

A range of the master ports connected to the DMA inputs.

- static constexpr auto m_dma_size

Number of master ports connected to the DMA inputs.

- static auto m_tile_ctrl_range

A range of the master ports connected to the tile control inputs.

- static constexpr auto m_tile_ctrl_size

Number of master ports connected to the tile control inputs.

- static auto m_fifo_range

A range of the master ports connected to the FIFO inputs.

- static constexpr auto m_fifo_size

Number of master ports connected to the FIFO inputs.

- static auto m_south_range

A range of the South master ports.

- static constexpr auto m_south_size

Number of South master ports.

- static auto m_west_range

  A range of the West master ports.
- static constexpr auto m_west_size

  Number of West master ports.
- static auto m_north_range

  A range of the North master ports.
- static constexpr auto m_north_size

  Number of North master ports.
- static auto m_east_range

  A range of the East master ports.
- static constexpr auto m_east_size

  Number of East master ports.
- static constexpr auto nb_slave_port

  Number of AXI stream slave ports in the switch.
- static auto s_me_range

  A range of the slave ports connected to the tile outputs.
- static constexpr auto s_me_size

  Number of slave ports connected to the tile outputs.
- static auto s_dma_range

  A range of the slave ports connected to the DMA outputs.
- static constexpr auto s_dma_size

  Number of slave ports connected to the DMA outputs.
- static auto s_tile_ctrl_range

  A range of the slave ports connected to the tile control outputs.
- static constexpr auto s_tile_ctrl_size

  Number of slave ports connected to the tile control outputs.
- static auto s_fifo_range

  A range of the slave ports connected to the FIFO outputs.
- static constexpr auto s_fifo_size

  Number of slave ports connected to the FIFO outputs.
- static auto s_south_range

  A range of the South slave ports.
- static constexpr auto s_south_size

  Number of South slave ports.
- static auto s_west_range

  A range of the West slave ports.
- static constexpr auto s_west_size

  Number of West slave ports.
- static auto s_north_range

  A range of the North slave ports.
- static constexpr auto s_north_size

  Number of North slave ports.
- static auto s_east_range

  A range of the East slave ports.
- static constexpr auto s_east_size

  Number of East slave ports.
- static auto interconnect

  Describe the AXI stream switch interconnection neighborhood.

Friends

- bool constexpr is_axi_master (master_port_layout)

  Use a hidden friend to have introspection for master_port_layout.
- bool constexpr is_axi_master (slave_port_layout)

  Use a hidden friend to have introspection for slave_port_layout.
- bool constexpr is_axi_master (master_port_layout)

  Use a hidden friend to have introspection for master_port_layout.
- bool constexpr is_axi_master (slave_port_layout)

  Use a hidden friend to have introspection for slave_port_layout.

12.16.1 Detailed Description

template<typename Layout>
struct trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch

Definition at line 365 of file geography.hpp.

12.16.2 Member Enumeration Documentation

12.16.2.1 master_port_layout [1/2]

```
template<typename Layout >
enum class trisycl::vendor::xilinx::acap::aie::geography::core_axi_stream_switch::master_port_layout
:  std::int8_t  [strong]
```

Layout of the AXI stream master ports in the switch.

Revision v2.02, 4.2.1 Types of AXI-Streams, Table 4-2, p. 101

Enumerator

me_0	
me_1	
me_last	Used for user port validation.
dma_0	
dma_1	
dma_last	Used for DMA validation.
tile_ctrl	
tile_ctrl_last	Used for validation.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	

Enumerator

south_last	Used for south validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_4	
north_5	
north_last	Used for north validation.
east_0	
east_1	
east_2	
east_3	
east_last	Used for east validation.
size	
me_0	
me_1	
me_last	Used for user port validation.
dma_0	
dma_1	
dma_last	Used for DMA validation.
tile_ctrl	
tile_ctrl_last	Used for validation.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	
south_last	Used for south validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_4	
north_5	
north_last	Used for north validation.
east_0	
east_1	
east_2	

Enumerator

east_3	
east_last	Used for east validation.
size	

Definition at line 375 of file geography.hpp.

```
00375                                   :  std::int8_t {
00376       me_0,
00377       me_1, me_last = me_1, ///< Used for user port validation
00378       dma_0,
00379       dma_1, dma_last = dma_1, ///< Used for DMA validation
00380       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation
00381       fifo_0,
00382       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00383       south_0,
00384       south_1,
00385       south_2,
00386       south_3, south_last = south_3, ///< Used for south validation
00387       west_0,
00388       west_1,
00389       west_2,
00390       west_3, west_last = west_3, ///< Used for west validation
00391       north_0,
00392       north_1,
00393       north_2,
00394       north_3,
00395       north_4,
00396       north_5, north_last = north_5, ///< Used for north validation
00397       east_0,
00398       east_1,
00399       east_2,
00400       east_3, east_last = east_3, ///< Used for east validation
00401       // To measure the enum
00402       size
00403     };
```

12.16.2.2 master_port_layout [2/2]

```
template<typename Layout >
enum class trisycl::vendor::xilinx::acap::aie::geography::core_axi_stream_switch::master_port_layout
:  std::int8_t  [strong]
```

Layout of the AXI stream master ports in the switch.

Revision v2.02, 4.2.1 Types of AXI-Streams, Table 4-2, p. 101

Enumerator

me_0	
me_1	
me_last	Used for user port validation.
dma_0	
dma_1	
dma_last	Used for DMA validation.
tile_ctrl	
tile_ctrl_last	Used for validation.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	

Enumerator

south_1	
south_2	
south_3	
south_last	Used for south validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_4	
north_5	
north_last	Used for north validation.
east_0	
east_1	
east_2	
east_3	
east_last	Used for east validation.
size	
me_0	
me_1	
me_last	Used for user port validation.
dma_0	
dma_1	
dma_last	Used for DMA validation.
tile_ctrl	
tile_ctrl_last	Used for validation.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	
south_last	Used for south validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_4	
north_5	
north_last	Used for north validation.

Enumerator

east_0	
east_1	
east_2	
east_3	
east_last	Used for east validation.
size	

Definition at line 375 of file geography.hpp.

```
00375                                    :  std::int8_t {
00376       me_0,
00377       me_1, me_last = me_1, ///< Used for user port validation
00378       dma_0,
00379       dma_1, dma_last = dma_1, ///< Used for DMA validation
00380       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation
00381       fifo_0,
00382       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00383       south_0,
00384       south_1,
00385       south_2,
00386       south_3, south_last = south_3, ///< Used for south validation
00387       west_0,
00388       west_1,
00389       west_2,
00390       west_3, west_last = west_3, ///< Used for west validation
00391       north_0,
00392       north_1,
00393       north_2,
00394       north_3,
00395       north_4,
00396       north_5, north_last = north_5, ///< Used for north validation
00397       east_0,
00398       east_1,
00399       east_2,
00400       east_3, east_last = east_3, ///< Used for east validation
00401       // To measure the enum
00402       size
00403     };
```

12.16.2.3 slave_port_layout [1/2]

```
template<typename Layout >
enum class trisycl::vendor::xilinx::acap::aie::geography::core_axi_stream_switch::slave_port_layout
:  std::int8_t  [strong]
```

Layout of the AXI stream slave ports in the switch.

Revision v2.02, 4.2.1 Types of AXI-Streams, Table 4-3, p. 102

Enumerator

me_0	
me_1	
me_last	Used for user port validation.
dma_0	
dma_1	
dma_last	Used for DMA validation.
tile_ctrl	
tile_ctrl_last	Used for validation,.
fifo_0	

Enumerator

fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	
south_4	
south_5	
south_last	Used for south validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_last	Used for north validation.
east_0	
east_1	
east_2	
east_3	
east_last	Used for east validation.
core_trace	
mem_trace	
size	
me_0	
me_1	
me_last	Used for user port validation.
dma_0	
dma_1	
dma_last	Used for DMA validation.
tile_ctrl	
tile_ctrl_last	Used for validation,.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	
south_4	
south_5	
south_last	Used for south validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.

Enumerator

north_0	
north_1	
north_2	
north_3	
north_last	Used for north validation.
east_0	
east_1	
east_2	
east_3	
east_last	Used for east validation.
core_trace	
mem_trace	
size	

Definition at line 501 of file geography.hpp.

```
00501                                    :  std::int8_t {
00502       me_0,
00503       me_1, me_last = me_1, ///< Used for user port validation
00504       dma_0,
00505       dma_1, dma_last = dma_1, ///< Used for DMA validation
00506       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation,
00507       fifo_0,
00508       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00509       south_0,
00510       south_1,
00511       south_2,
00512       south_3,
00513       south_4,
00514       south_5, south_last = south_5, ///< Used for south validation
00515       west_0,
00516       west_1,
00517       west_2,
00518       west_3, west_last = west_3, ///< Used for west validation
00519       north_0,
00520       north_1,
00521       north_2,
00522       north_3, north_last = north_3, ///< Used for north validation
00523       east_0,
00524       east_1,
00525       east_2,
00526       east_3, east_last = east_3, ///< Used for east validation
00527       core_trace,
00528       mem_trace,
00529       // To measure the enum
00530       size
00531     };
```

12.16.2.4 slave_port_layout [2/2]

```
template<typename Layout >
enum class trisycl::vendor::xilinx::acap::aie::geography::core_axi_stream_switch::slave_port_layout
:  std::int8_t  [strong]
```

Layout of the AXI stream slave ports in the switch.

Revision v2.02, 4.2.1 Types of AXI-Streams, Table 4-3, p. 102

Enumerator

me_0	

Enumerator

me_1	
me_last	Used for user port validation.
dma_0	
dma_1	
dma_last	Used for DMA validation.
tile_ctrl	
tile_ctrl_last	Used for validation,.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	
south_4	
south_5	
south_last	Used for south validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_last	Used for north validation.
east_0	
east_1	
east_2	
east_3	
east_last	Used for east validation.
core_trace	
mem_trace	
size	
me_0	
me_1	
me_last	Used for user port validation.
dma_0	
dma_1	
dma_last	Used for DMA validation.
tile_ctrl	
tile_ctrl_last	Used for validation,.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	

Enumerator

south_4	
south_5	
south_last	Used for south validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_last	Used for north validation.
east_0	
east_1	
east_2	
east_3	
east_last	Used for east validation.
core_trace	
mem_trace	
size	

Definition at line 501 of file geography.hpp.

```
00501                                    :  std::int8_t {
00502       me_0,
00503       me_1, me_last = me_1, ///< Used for user port validation
00504       dma_0,
00505       dma_1, dma_last = dma_1, ///< Used for DMA validation
00506       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation,
00507       fifo_0,
00508       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00509       south_0,
00510       south_1,
00511       south_2,
00512       south_3,
00513       south_4,
00514       south_5, south_last = south_5, ///< Used for south validation
00515       west_0,
00516       west_1,
00517       west_2,
00518       west_3, west_last = west_3, ///< Used for west validation
00519       north_0,
00520       north_1,
00521       north_2,
00522       north_3, north_last = north_3, ///< Used for north validation
00523       east_0,
00524       east_1,
00525       east_2,
00526       east_3, east_last = east_3, ///< Used for east validation
00527       core_trace,
00528       mem_trace,
00529       // To measure the enum
00530       size
00531     };
```

12.16.3 Friends And Related Function Documentation

12.16.3.1 is_axi_master [1/4]

```
template<typename Layout >
bool constexpr is_axi_master (
            master_port_layout  )  [friend]
```

Use a hidden friend to have introspection for master_port_layout.

Definition at line 408 of file geography.hpp.

```
00408                                                       {
00409       // Yes, master_port_layout is definitely an AXI master
00410       return true;
00411     }
```

12.16.3.2 is_axi_master [2/4]

```
template<typename Layout >
bool constexpr is_axi_master (
            master_port_layout  )  [friend]
```

Use a hidden friend to have introspection for master_port_layout.

Definition at line 408 of file geography.hpp.

```
00408                                                       {
00409       // Yes, master_port_layout is definitely an AXI master
00410       return true;
00411     }
```

12.16.3.3 is_axi_master [3/4]

```
template<typename Layout >
bool constexpr is_axi_master (
            slave_port_layout  )  [friend]
```

Use a hidden friend to have introspection for slave_port_layout.

Definition at line 536 of file geography.hpp.

```
00536                                                      {
00537       // No, slave_port_layout is definitely not an AXI master
00538       return false;
00539     }
```

12.16.3.4 is_axi_master [4/4]

```
template<typename Layout >
bool constexpr is_axi_master (
            slave_port_layout  )  [friend]
```

Use a hidden friend to have introspection for slave_port_layout.

Definition at line 536 of file geography.hpp.

```
00536                                                      {
00537       // No, slave_port_layout is definitely not an AXI master
00538       return false;
00539     }
```

12.16.4 Member Data Documentation

12.16.4.1 fifo_depth

```
template<typename Layout >
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::fifo_depth = 16  [static], [constexpr]
```

Depth of additional FIFO on specific FIFO paths.

Definition at line 370 of file geography.hpp.

12.16.4.2 interconnect

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::interconnect  [inline], [static]
```

Initial value:

```
= boost::hana::make_tuple(

        std::tuple { 1, 0, m_east_range, s_west_range }

      , std::tuple { -1, 0, m_west_range, s_east_range }

      , std::tuple { 0, 1, m_north_range, s_south_range }

      , std::tuple { 0, -1, m_south_range, s_north_range }
      )
```

Describe the AXI stream switch interconnection neighborhood.

Definition at line 621 of file geography.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_neighborhood().

12.16.4.3 latency

```
template<typename Layout >
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::latency = 4  [static], [constexpr]
```

Default number of registers on default router paths.

Definition at line 367 of file geography.hpp.

12.16.4.4 m_dma_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::m_dma_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::dma_0, master_port_layout::dma_last)
```

A range of the master ports connected to the DMA inputs.

Definition at line 430 of file geography.hpp.

12.16.4.5 m_dma_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::m_dma_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::dma_0,
                         master_port_layout::dma_last)
```

Number of master ports connected to the DMA inputs.

Definition at line 434 of file geography.hpp.

12.16.4.6 m_east_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::m_east_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::east_0,
                       master_port_layout::east_last)
```

A range of the East master ports.

Definition at line 489 of file geography.hpp.

12.16.4.7 m_east_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::m_east_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::east_0,
                         master_port_layout::east_last)
```

Number of East master ports.

Definition at line 494 of file geography.hpp.

12.16.4.8 m_fifo_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::m_fifo_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::fifo_0,
                       master_port_layout::fifo_last)
```

A range of the master ports connected to the FIFO inputs.

Definition at line 449 of file geography.hpp.

12.16.4.9 m_fifo_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::m_fifo_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::fifo_0,
                         master_port_layout::fifo_last)
```

Number of master ports connected to the FIFO inputs.

Definition at line 454 of file geography.hpp.

12.16.4.10 m_me_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::m_me_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::me_0, master_port_layout::me_last)
```

A range of the master ports connected to the tile inputs.

Definition at line 422 of file geography.hpp.

12.16.4.11 m_me_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::m_me_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::me_0, master_port_layout::me_last)
```

Number of master ports connected to the tile inputs.

Definition at line 426 of file geography.hpp.

12.16.4.12 m_north_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::m_north_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::north_0,
                       master_port_layout::north_last)
```

A range of the North master ports.

Definition at line 479 of file geography.hpp.

12.16.4.13 m_north_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::m_north_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::north_0,
                         master_port_layout::north_last)
```

Number of North master ports.

Definition at line 484 of file geography.hpp.

12.16.4.14 m_south_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::m_south_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::south_0,
                       master_port_layout::south_last)
```

A range of the South master ports.

Definition at line 459 of file geography.hpp.

12.16.4.15 m_south_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::m_south_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::south_0,
                         master_port_layout::south_last)
```

Number of South master ports.

Definition at line 464 of file geography.hpp.

12.16.4.16 m_tile_ctrl_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::m_tile_ctrl_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::tile_ctrl,
                       master_port_layout::tile_ctrl_last)
```

A range of the master ports connected to the tile control inputs.

Definition at line 439 of file geography.hpp.

12.16.4.17 m_tile_ctrl_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::m_tile_ctrl_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::tile_ctrl,
                         master_port_layout::tile_ctrl_last)
```

Number of master ports connected to the tile control inputs.

Definition at line 444 of file geography.hpp.

12.16.4.18 m_west_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::m_west_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::west_0,
                       master_port_layout::west_last)
```

A range of the West master ports.

Definition at line 469 of file geography.hpp.

12.16.4.19 m_west_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::m_west_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::west_0,
                         master_port_layout::west_last)
```

Number of West master ports.

Definition at line 474 of file geography.hpp.

12.16.4.20 nb_master_port

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::nb_master_port  [static], [constexpr]
```

Initial value:

```
=
      detail::underlying_value(master_port_layout::size)
```

Number of AXI stream master ports in the switch.

Todo factorize with a mix-in the following between master and slave

Todo factorize with a mix-in the following between master and slave

Definition at line 418 of file geography.hpp.

12.16.4.21 nb_slave_port

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::nb_slave_port  [static], [constexpr]
```

Initial value:

```
=
      detail::underlying_value(slave_port_layout::size)
```

Number of AXI stream slave ports in the switch.

Definition at line 544 of file geography.hpp.

12.16.4.22 s_dma_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::s_dma_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::dma_0, slave_port_layout::dma_last)
```

A range of the slave ports connected to the DMA outputs.

Definition at line 556 of file geography.hpp.

12.16.4.23 s_dma_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::s_dma_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::dma_0, slave_port_layout::dma_last)
```

Number of slave ports connected to the DMA outputs.

Definition at line 560 of file geography.hpp.

12.16.4.24 s_east_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::s_east_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::east_0, slave_port_layout::east_last)
```

A range of the East slave ports.

Definition at line 612 of file geography.hpp.

12.16.4.25 s_east_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::s_east_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::east_0,
                         slave_port_layout::east_last)
```

Number of East slave ports.

Definition at line 616 of file geography.hpp.

12.16.4.26 s_fifo_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::s_fifo_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::fifo_0, slave_port_layout::fifo_last)
```

A range of the slave ports connected to the FIFO outputs.

Definition at line 574 of file geography.hpp.

12.16.4.27 s_fifo_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::s_fifo_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::fifo_0,
                         slave_port_layout::fifo_last)
```

Number of slave ports connected to the FIFO outputs.

Definition at line 578 of file geography.hpp.

12.16.4.28 s_me_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::s_me_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::me_0, slave_port_layout::me_last)
```

A range of the slave ports connected to the tile outputs.

Definition at line 548 of file geography.hpp.

12.16.4.29 s_me_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::s_me_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::me_0, slave_port_layout::me_last)
```

Number of slave ports connected to the tile outputs.

Definition at line 552 of file geography.hpp.

12.16.4.30 s_north_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::s_north_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::north_0,
                       slave_port_layout::north_last)
```

A range of the North slave ports.

Definition at line 602 of file geography.hpp.

12.16.4.31 s_north_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::s_north_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::north_0,
                         slave_port_layout::north_last)
```

Number of North slave ports.

Definition at line 607 of file geography.hpp.

12.16.4.32 s_south_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::s_south_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::south_0,
                       slave_port_layout::south_last)
```

A range of the South slave ports.

Definition at line 583 of file geography.hpp.

12.16.4.33 s_south_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::s_south_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::south_0,
                         slave_port_layout::south_last)
```

Number of South slave ports.

Definition at line 588 of file geography.hpp.

12.16.4.34 s_tile_ctrl_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::s_tile_ctrl_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::tile_ctrl,
                       slave_port_layout::tile_ctrl_last)
```

A range of the slave ports connected to the tile control outputs.

Definition at line 564 of file geography.hpp.

12.16.4.35 s_tile_ctrl_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::s_tile_ctrl_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::tile_ctrl,
                         slave_port_layout::tile_ctrl_last)
```

Number of slave ports connected to the tile control outputs.

Definition at line 569 of file geography.hpp.

12.16.4.36 s_west_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch↩
::s_west_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::west_0, slave_port_layout::west_last)
```

A range of the West slave ports.

Definition at line 593 of file geography.hpp.

12.16.4.37 s_west_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_↩
stream_switch::s_west_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::west_0,
                         slave_port_layout::west_last)
```

Number of West slave ports.

Definition at line 597 of file geography.hpp.

The documentation for this struct was generated from the following files:

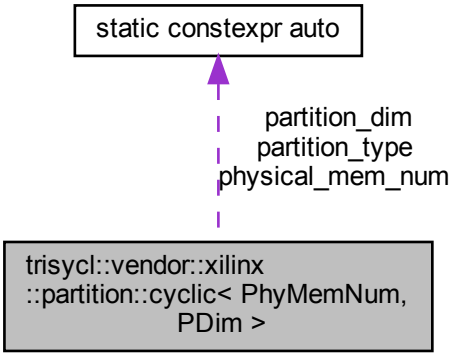
- include/sycl/vendor/Xilinx/acap/aie/geography.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/geography.hpp

12.17 trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim > Struct Template Reference

Represent a cyclic partition.

`#include <partition_array.hpp>`

Collaboration diagram for trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim >:



Static Public Attributes

- static constexpr auto physical_mem_num = PhyMemNum
- static constexpr auto partition_dim = PDim
- static constexpr auto partition_type = type::cyclic

12.17.1 Detailed Description

template<std::size_t PhyMemNum = 1, std::size_t PDim = 1>
struct trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim >

Represent a cyclic partition.

The single array would be partitioned into several small physical memories in this case. These small physical memories can be accessed simultaneously which drive the performance. Each element in the array would be partitioned to each memory in order and cyclically.

That is if we have a 4-element array which contains 4 integers 0, 1, 2, and 3. If we set factor to 2, and partition dimension to 1 for this cyclic partition array. Then, the contents of this array will be distributed to 2 physical memories: one contains 1, 3 and the other contains 2,4.

Parameters

PhyMemNum	is the number of physical memories that user wants to have.
PDim	is the dimension that user wants to apply cyclic partition on. If PDim is 0, all dimensions will be partitioned with cyclic order.

Todo Deal with multi-dimension array. Now, since we can only deal with 1-dim, PDim is set to 1 by default.

The single array would be partitioned into several small physical memories in this case. These small physical memories can be accessed simultaneously which drive the performance. Each element in the array would be partitioned to each memory in order and cyclically.

That is if we have a 4-element array which contains 4 integers 0, 1, 2, and 3. If we set factor to 2, and partition dimension to 1 for this cyclic partition array. Then, the contents of this array will be distributed to 2 physical memories: one contains 1, 3 and the other contains 2,4.

Parameters

PhyMemNum	is the number of physical memories that user wants to have.
PDim	is the dimension that user wants to apply cyclic partition on. If PDim is 0, all dimensions will be partitioned with cyclic order.

Todo Deal with multi-dimension array. Now, since we can only deal with 1-dim, PDim is set to 1 by default.

Definition at line 68 of file partition_array.hpp.

12.17.2 Member Data Documentation

12.17.2.1 partition_dim

```
template<std::size_t PhyMemNum = 1, std::size_t PDim = 1>
static constexpr auto trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim >::partition↩
_dim = PDim  [static], [constexpr]
```

Definition at line 70 of file partition_array.hpp.

12.17.2.2 partition_type

```
template<std::size_t PhyMemNum = 1, std::size_t PDim = 1>
static constexpr auto trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim >::partition↩
_type = type::cyclic  [static], [constexpr]
```

Definition at line 71 of file partition_array.hpp.

12.17.2.3 physical_mem_num

```
template<std::size_t PhyMemNum = 1, std::size_t PDim = 1>
static constexpr auto trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim >::physical↩
_mem_num = PhyMemNum  [static], [constexpr]
```

Definition at line 69 of file partition_array.hpp.

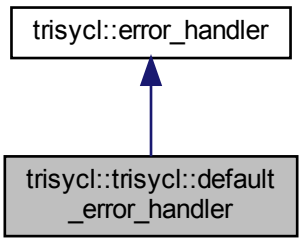
The documentation for this struct was generated from the following files:

- include/sycl/vendor/Xilinx/fpga/partition_array.hpp
- include/triSYCL/vendor/Xilinx/fpga/partition_array.hpp

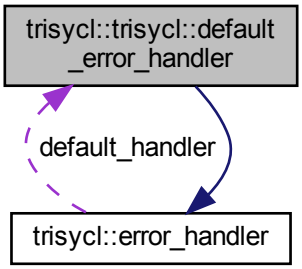
12.18 trisycl::trisycl::default_error_handler Struct Reference

```
#include <error_handler.hpp>
```

Inheritance diagram for trisycl::trisycl::default_error_handler:



Collaboration diagram for trisycl::trisycl::default_error_handler:



Public Member Functions

- void report_error (exception &) override

  The method to define to be called in the case of an error.

Additional Inherited Members

12.18.1 Detailed Description

Definition at line 51 of file error_handler.hpp.

12.18.2 Member Function Documentation

12.18.2.1 report_error()

```
void trisycl::trisycl::default_error_handler::report_error (
            exception & error )  [inline], [override], [virtual]
```

The method to define to be called in the case of an error.

Todo Add "virtual void" to the specification

Implements trisycl::error_handler.

Definition at line 53 of file error_handler.hpp.

```
00053                                                {
00054     }
```

The documentation for this struct was generated from the following file:

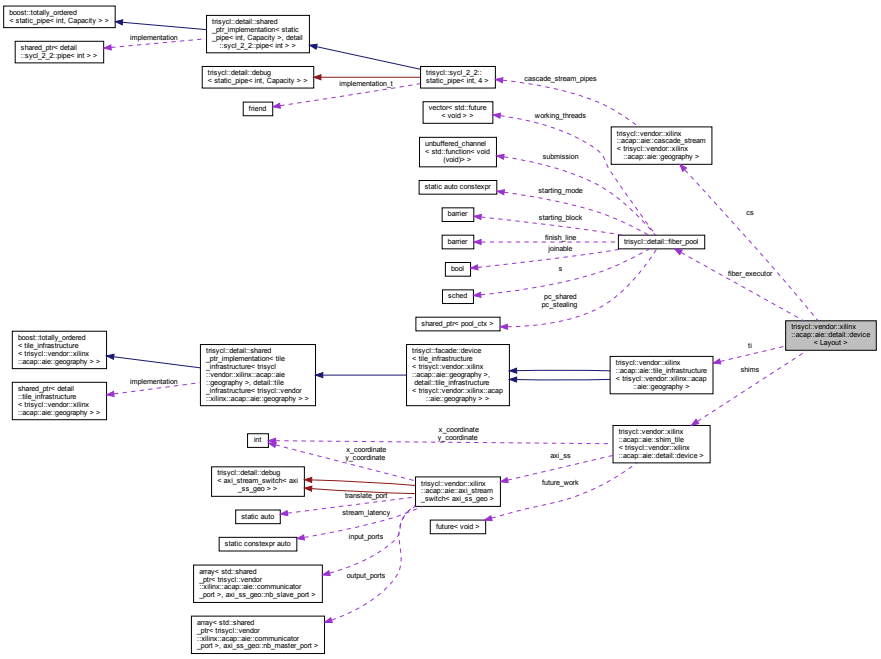
- include/triSYCL/error_handler.hpp

12.19 trisycl::vendor::xilinx::acap::aie::detail::device< Layout > Struct Template Reference

Create a SYCL-like device view of an AI Engine CGRA with some layout.

`#include <device.hpp>`

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::detail::device< Layout >:



Public Types

- using geo = geography< Layout >

  The geography of the CGRA.
- using cass = typename geo::core_axi_stream_switch

  Naming shortcut for the core AXI stream switch.
- using csp = typename cass::slave_port_layout

  Naming shortcut for the slave ports of the core AXI stream switch.
- using cmp = typename cass::master_port_layout

  Naming shortcut for the master ports of the core AXI stream switch.
- using sass = typename geo::shim_axi_stream_switch

  Naming shortcut for the shim AXI stream switch.
- using ssp = typename sass::slave_port_layout

  Naming shortcut for the slave ports of the shim AXI stream switch.
- using smp = typename sass::master_port_layout

  Naming shortcut for the master ports of the shim AXI stream switch.
- using geo = geography< Layout >

  The geography of the CGRA.
- using cass = typename geo::core_axi_stream_switch

  Naming shortcut for the core AXI stream switch.

- using csp = typename cass::slave_port_layout

  Naming shortcut for the slave ports of the core AXI stream switch.
- using cmp = typename cass::master_port_layout

  Naming shortcut for the master ports of the core AXI stream switch.
- using sass = typename geo::shim_axi_stream_switch

  Naming shortcut for the shim AXI stream switch.
- using ssp = typename sass::slave_port_layout

  Naming shortcut for the slave ports of the shim AXI stream switch.
- using smp = typename sass::master_port_layout

  Naming shortcut for the master ports of the shim AXI stream switch.

Public Member Functions

- auto & tile (int x, int y)

  Access to the common infrastructure part of a tile.
- template<typename F >
  void for_each_tile_index (F &&f)

  Apply a function for each tile index of the device.
- template<typename F >
  void for_each_tile (F &&f)

  Apply a function for each tile infrastructure of the device.
- template<typename F >
  void for_each_tile_x_index (F &&f)

  Apply a function for each x tile index of the device.
- template<typename F >
  void for_each_tile_y_index (F &&f)

  Apply a function for each y tile index of the device.
- auto & mem (int x, int y)

  Shortcut to access to the common infrastructure part of a tile memory.
- auto & shim (int x)

  Access to the shim tile.
- auto & cascade ()

  Access the cascade connections.
- template<typename SrcPort , typename DstPort >
  void connect (SrcPort &&src, DstPort &&dst)

  Connect the ports of 2 tiles or shims together with an hyperspace switched circuit, jumping over the underlying routing infrastructure.
- template<typename F >
  void for_each_tile_neighborhood (F &&f)

  Apply a function on all the AXI stream of the neighborhood of each tile.
- device ()

  Build some device level infrastructure.
- auto display (const std::string &file_name={})

  Display the device layout.
- auto & tile (int x, int y)

  Access to the common infrastructure part of a tile.
- template<typename F >
  void for_each_tile_index (F &&f)

  Apply a function for each tile index of the device.
- template<typename F >
  void for_each_tile (F &&f)

Apply a function for each tile infrastructure of the device.

- template<typename F >
  void for_each_tile_x_index (F &&f)

  Apply a function for each x tile index of the device.
- template<typename F >
  void for_each_tile_y_index (F &&f)

  Apply a function for each y tile index of the device.
- auto & mem (int x, int y)

  Shortcut to access to the common infrastructure part of a tile memory.
- auto & shim (int x)

  Access to the shim tile.
- auto & cascade ()

  Access the cascade connections.
- template<typename SrcPort , typename DstPort >
  void connect (SrcPort &&src, DstPort &&dst)

  Connect the ports of 2 tiles or shims together with an hyperspace switched circuit, jumping over the underlying routing infrastructure.
- template<typename F >
  void for_each_tile_neighborhood (F &&f)

  Apply a function on all the AXI stream of the neighborhood of each tile.
- device ()

  Build some device level infrastructure.
- auto display (const std::string &file_name={})

  Display the device layout.

Public Attributes

- cascade_stream< geo > cs

  The cascade stream infrastructure of the CGRA.
- ::trisycl::detail::fiber_pool fiber_executor

  A fiber pool executor to run the infrastructure powered by TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS std::thread.
- aie::tile_infrastructure< geo > ti [geo::y_size][geo::x_size]

  Keep track of all the (non detail) infrastructure tiles of this device.
- shim_tile< device > shims [geo::x_size]

  The shim tiles on the lower row of the tile array.

12.19.1 Detailed Description

template<typename Layout>
struct trisycl::vendor::xilinx::acap::aie::detail::device< Layout >

Create a SYCL-like device view of an AI Engine CGRA with some layout.

Parameters

Layout	is the layout description of the machine to instantiate with the physical size

Definition at line 41 of file device.hpp.

12.19.2 Member Typedef Documentation

12.19.2.1 cass [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::cass = typename geo::core_axi_stream_swit
```

Naming shortcut for the core AXI stream switch.

Definition at line 47 of file device.hpp.

12.19.2.2 cass [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::cass = typename geo::core_axi_stream_swit
```

Naming shortcut for the core AXI stream switch.

Definition at line 47 of file device.hpp.

12.19.2.3 cmp [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::cmp = typename cass↩
::master_port_layout
```

Naming shortcut for the master ports of the core AXI stream switch.

Definition at line 53 of file device.hpp.

12.19.2.4 cmp [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::cmp = typename cass↩
::master_port_layout
```

Naming shortcut for the master ports of the core AXI stream switch.

Definition at line 53 of file device.hpp.

12.19.2.5 csp [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::csp = typename cass↩
::slave_port_layout
```

Naming shortcut for the slave ports of the core AXI stream switch.

Definition at line 50 of file device.hpp.

12.19.2.6 csp [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::csp = typename cass↩
::slave_port_layout
```

Naming shortcut for the slave ports of the core AXI stream switch.

Definition at line 50 of file device.hpp.

12.19.2.7 geo [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::geo = geography<Layout>
```

The geography of the CGRA.

Definition at line 44 of file device.hpp.

12.19.2.8 geo [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::geo = geography<Layout>
```

The geography of the CGRA.

Definition at line 44 of file device.hpp.

12.19.2.9 sass [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::sass = typename geo::shim_axi_stream_swit
```

Naming shortcut for the shim AXI stream switch.

Definition at line 56 of file device.hpp.

12.19.2.10 sass [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::sass = typename geo::shim_axi_stream_swit
```

Naming shortcut for the shim AXI stream switch.

Definition at line 56 of file device.hpp.

12.19.2.11 smp [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::smp = typename sass↩
::master_port_layout
```

Naming shortcut for the master ports of the shim AXI stream switch.

Definition at line 62 of file device.hpp.

12.19.2.12 smp [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::smp = typename sass↩
::master_port_layout
```

Naming shortcut for the master ports of the shim AXI stream switch.

Definition at line 62 of file device.hpp.

12.19.2.13 ssp [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::ssp = typename sass↩
::slave_port_layout
```

Naming shortcut for the slave ports of the shim AXI stream switch.

Definition at line 59 of file device.hpp.

12.19.2.14 ssp [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::ssp = typename sass↩
::slave_port_layout
```

Naming shortcut for the slave ports of the shim AXI stream switch.

Definition at line 59 of file device.hpp.

12.19.3 Constructor & Destructor Documentation

12.19.3.1 device() [1/2]

```
template<typename Layout >
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::device ( )  [inline]
```

Build some device level infrastructure.

Make the connections around the shim tiles

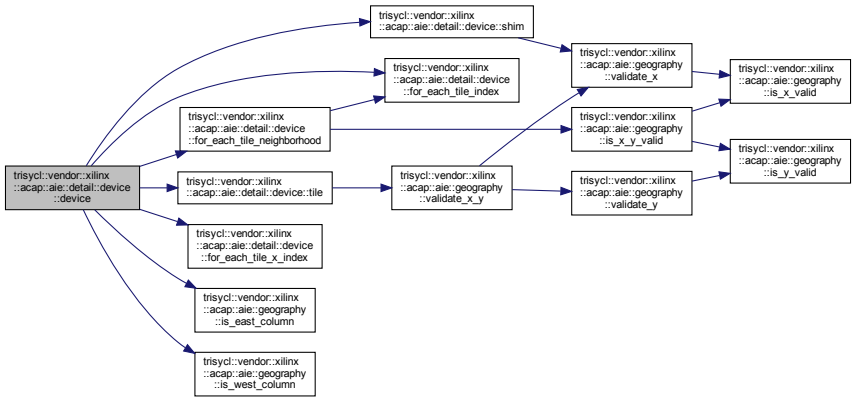
Definition at line 226 of file device.hpp.

```
00226       {
00227     // Initialize all the tiles with their network connections first
00228     for_each_tile_index([&](auto x, auto y) {
00229       // Create & start the tile infrastructure
00230       tile(x, y) = { x, y, *this, fiber_executor };
00231     });
00232     for_each_tile_x_index([&](auto x) {
00233       // Start the shim infrastructure
00234       shim(x).start(x, fiber_executor);
00235     });
00236
00237     // Only then we can connect the inter-core tile AXI stream NoC
00238     for_each_tile_neighborhood(
00239         [&](auto x, auto y, auto nx, auto ny, auto m, auto s) {
00240           tile(x, y).output(m) = tile(nx, ny).input(s);
00241         });
00242
00243 /// Make the connections around the shim tiles
00244     for_each_tile_x_index([&](auto x) {
00245       // The connection from shim tile to core tile
00246       for (auto [o, i] :
00247            ranges::views::zip(sass::m_north_range, cass::s_south_range))
00248         shim(x).output(o) = tile(x, 0).input(i);
00249       // The connection from core tile to shim tile
00250       for (auto [o, i] :
```

```
00251           ranges::views::zip(cass::m_south_range, sass::s_north_range))
00252         tile(x, 0).output(o) = shim(x).input(i);
00253       if (!geo::is_east_column(x))
00254         for (auto [o, i] :
00255              ranges::views::zip(sass::m_east_range, sass::s_west_range))
00256           shim(x).output(o) = shim(x + 1).input(i);
00257       if (!geo::is_west_column(x))
00258         for (auto [o, i] :
00259              ranges::views::zip(sass::m_west_range, sass::s_east_range))
00260           shim(x).output(o) = shim(x - 1).input(i);
00261     });
00262   }
```

References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::fiber_executor, trisycl::vendor::xilinx::acap::aie::detail::device< trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_neighborhood(), trisycl::vendor::xilinx::acap::aie::detail::devic i, trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_east_column(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::shim(), trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::tile(), x, and trisycl::y.

Here is the call graph for this function:



12.19.3.2 device() [2/2]

```
template<typename Layout >
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::device ( )  [inline]
```

Build some device level infrastructure.

Make the connections around the shim tiles

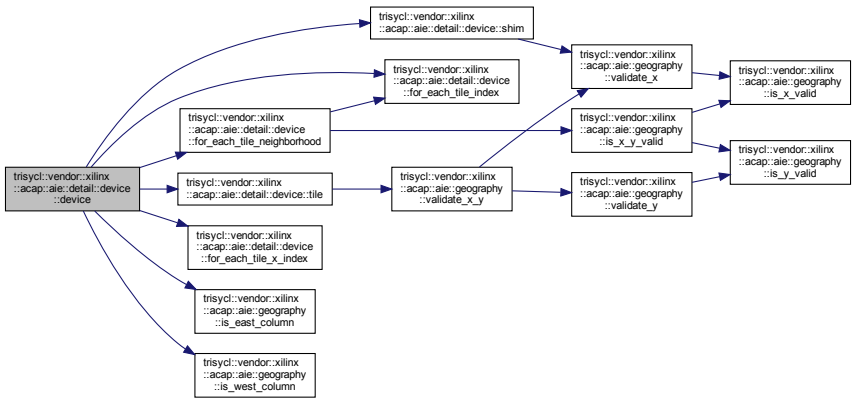
Definition at line 226 of file device.hpp.

```
00226           {
00227     // Initialize all the tiles with their network connections first
00228     for_each_tile_index([&](auto x, auto y) {
00229       // Create & start the tile infrastructure
00230       tile(x, y) = { x, y, *this, fiber_executor };
00231     });
00232     for_each_tile_x_index([&](auto x) {
00233       // Start the shim infrastructure
00234       shim(x).start(x, fiber_executor);
00235     });
00236
00237     // Only then we can connect the inter-core tile AXI stream NoC
00238     for_each_tile_neighborhood(
00239         [&](auto x, auto y, auto nx, auto ny, auto m, auto s) {
```

```
00240           tile(x, y).output(m) = tile(nx, ny).input(s);
00241         });
00242
00243 /// Make the connections around the shim tiles
00244     for_each_tile_x_index([&](auto x) {
00245       // The connection from shim tile to core tile
00246       for (auto [o, i] :
00247            ranges::views::zip(sass::m_north_range, cass::s_south_range))
00248         shim(x).output(o) = tile(x, 0).input(i);
00249       // The connection from core tile to shim tile
00250       for (auto [o, i] :
00251            ranges::views::zip(cass::m_south_range, sass::s_north_range))
00252         tile(x, 0).output(o) = shim(x).input(i);
00253       if (!geo::is_east_column(x))
00254         for (auto [o, i] :
00255              ranges::views::zip(sass::m_east_range, sass::s_west_range))
00256           shim(x).output(o) = shim(x + 1).input(i);
00257       if (!geo::is_west_column(x))
00258         for (auto [o, i] :
00259              ranges::views::zip(sass::m_west_range, sass::s_east_range))
00260           shim(x).output(o) = shim(x - 1).input(i);
00261     });
00262   }
```

References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::fiber_executor, trisycl::vendor::xilinx::acap::aie::detail::device< trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_neighborhood(), trisycl::vendor::xilinx::acap::aie::detail::devic i, trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_east_column(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::shim(), trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::tile(), x, and trisycl::y.

Here is the call graph for this function:



12.19.4 Member Function Documentation

12.19.4.1 cascade() [1/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::cascade ( )  [inline]
```

Access the cascade connections.

Todo To remove?

Definition at line 165 of file device.hpp.

```
00165 { return cs; }
```

References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::cs.

12.19.4.2 cascade() [2/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::cascade ( )  [inline]
```

Access the cascade connections.

Todo To remove?

Definition at line 165 of file device.hpp.

```
00165 { return cs; }
```

References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::cs.

12.19.4.3 connect() [1/2]

```
template<typename Layout >
template<typename SrcPort , typename DstPort >
void trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::connect (
            SrcPort && src,
            DstPort && dst )  [inline]
```

Connect the ports of 2 tiles or shims together with an hyperspace switched circuit, jumping over the underlying routing infrastructure.

This is used to speed up communication in emulation and to skip the need of a routing planner.

Todo To deprecate and implement in another way?

Parameters

in	*T*	is the type of the data to be transferred
in	*SrcPort*	is the type of the source port, such as port::tile or port::shim
in	*DstPort*	is the type of the destination port, such as port::tile or port::shim

Exceptions

trisycl::runtime_error	if some coordinates or port numbers are invalid

Todo Refactor, make the difference between user & physical ports

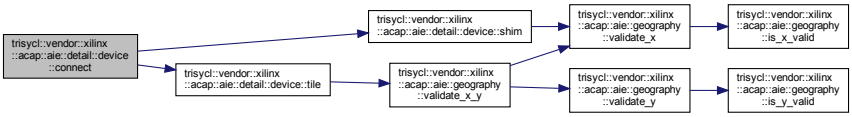
Definition at line 190 of file device.hpp.

```
00190                                          {
00191     static_assert(std::is_same_v<SrcPort, port::tile> ||
00192                   std::is_same_v<SrcPort, port::shim>,
00193                   "SrcPort type should be port::tile or port::shim");
00194     static_assert(std::is_same_v<DstPort, port::tile> ||
```

```
00195                     std::is_same_v<DstPort, port::shim>,
00196                   "DstPort type should be port::tile or port::shim");
00197     // Get the destination in the final switch
00198     auto channel = [&] {
00199       if constexpr (std::is_same_v<DstPort, port::tile>)
00200         return tile(dst.x, dst.y).in_connection(dst.port);
00201       else if constexpr (std::is_same_v<DstPort, port::shim>)
00202         return shim(dst.x).bli_in_connection(dst.port);
00203     }();
00204     // And connect the source  port to it
00205     if constexpr (std::is_same_v<SrcPort, port::tile>)
00206       tile(src.x, src.y).out_connection(src.port) = channel;
00207     else if constexpr (std::is_same_v<SrcPort, port::shim>)
00208       shim(src.x).bli_out_connection(src.port) = channel;
00209   }
```

References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::shim(), src, and trisycl::vendor::xilinx::acap::aie::detail::device<

Here is the call graph for this function:



12.19.4.4 connect() [2/2]

```
template<typename Layout >
template<typename SrcPort , typename DstPort >
void trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::connect (
            SrcPort && src,
            DstPort && dst )  [inline]
```

Connect the ports of 2 tiles or shims together with an hyperspace switched circuit, jumping over the underlying routing infrastructure.

This is used to speed up communication in emulation and to skip the need of a routing planner.

Todo To deprecate and implement in another way?

Parameters

in	*T*	is the type of the data to be transferred
in	*SrcPort*	is the type of the source port, such as port::tile or port::shim
in	*DstPort*	is the type of the destination port, such as port::tile or port::shim

Exceptions

trisycl::runtime_error	if some coordinates or port numbers are invalid

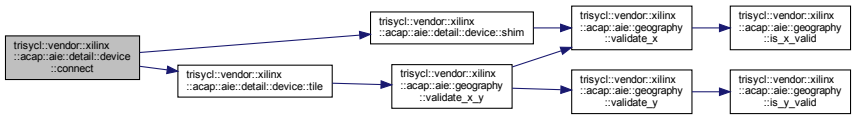
Todo Refactor, make the difference between user & physical ports

Definition at line 190 of file device.hpp.

```
00190                                                {
00191     static_assert(std::is_same_v<SrcPort, port::tile> ||
00192                       std::is_same_v<SrcPort, port::shim>,
00193                   "SrcPort type should be port::tile or port::shim");
00194     static_assert(std::is_same_v<DstPort, port::tile> ||
00195                       std::is_same_v<DstPort, port::shim>,
00196                   "DstPort type should be port::tile or port::shim");
00197     // Get the destination in the final switch
00198     auto channel = [&] {
00199       if constexpr (std::is_same_v<DstPort, port::tile>)
00200         return tile(dst.x, dst.y).in_connection(dst.port);
00201       else if constexpr (std::is_same_v<DstPort, port::shim>)
00202         return shim(dst.x).bli_in_connection(dst.port);
00203     }();
00204     // And connect the source  port to it
00205     if constexpr (std::is_same_v<SrcPort, port::tile>)
00206       tile(src.x, src.y).out_connection(src.port) = channel;
00207     else if constexpr (std::is_same_v<SrcPort, port::shim>)
00208       shim(src.x).bli_out_connection(src.port) = channel;
00209   }
```

References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::shim(), src, and trisycl::vendor::xilinx::acap::aie::detail::device<

Here is the call graph for this function:



12.19.4.5 display() [1/2]

```
template<typename Layout >
auto trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::display (
              const std::string & file_name = {} )  [inline]
```

Display the device layout.

Parameters

in	*file_name*	is the name of the file to write the LaTeX drawing into or use std::cerr by default

The LaTeX generation is handled by a LaTeX context

Definition at line 269 of file device.hpp.

```
00269                                            {}) {
00270     // Compute the drawing size starting with the individual tile size
00271     auto tile_size = tile(0, 0).display_size();
00272     // And expanding according to the device size.  Add 1 in each
00273     // dimension to fit in the page because of the borders
00274     auto x_size = tile_size.x() * geo::x_size + 1;
00275     auto y_size = tile_size.y() * geo::y_size + 1;
00276 /// The LaTeX generation is handled by a LaTeX context
00277     latex::context c { { x_size, y_size }, file_name };
00278
```

```
00279     for_each_tile_index([&](auto x, auto y) { tile(x, y).display(c); });
00280
00281     // Connect each tile to its neighbors
00282     for_each_tile_neighborhood(
00283         [&](auto x, auto y, auto nx, auto ny, auto m, auto s) {
00284           c.add((boost::format { R"(
00285 \draw (node cs:name=TileX%1%Y%2%M%3%)
00286 -- (node cs:name=TileX%4%Y%5%S%6%);)" } %
00287 x % y % c.clean_node(magic_enum::enum_name(m)) % nx % ny %
00288 c.clean_node(magic_enum::enum_name(s)))
00289 .str());
00290 });
00291
```

12.19.4.6 display() [2/2]

```
template<typename Layout >
auto trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::display (
             const std::string & file_name = {} )  [inline]
```

Display the device layout.

Parameters

in	*file_name*	is the name of the file to write the LaTeX drawing into or use std::cerr by default

The LaTeX generation is handled by a LaTeX context

Definition at line 269 of file device.hpp.

```
00269                                       {}) {
00270     // Compute the drawing size starting with the individual tile size
00271     auto tile_size = tile(0, 0).display_size();
00272     // And expanding according to the device size.  Add 1 in each
00273     // dimension to fit in the page because of the borders
00274     auto x_size = tile_size.x() * geo::x_size + 1;
00275     auto y_size = tile_size.y() * geo::y_size + 1;
00276 /// The LaTeX generation is handled by a LaTeX context
00277     latex::context c { { x_size, y_size }, file_name };
00278
00279     for_each_tile_index([&](auto x, auto y) { tile(x, y).display(c); });
00280
00281     // Connect each tile to its neighbors
00282     for_each_tile_neighborhood(
00283         [&](auto x, auto y, auto nx, auto ny, auto m, auto s) {
00284           c.add((boost::format { R"(
00285 \draw (node cs:name=TileX%1%Y%2%M%3%)
00286 -- (node cs:name=TileX%4%Y%5%S%6%);)" } %
00287 x % y % c.clean_node(magic_enum::enum_name(m)) % nx % ny %
00288 c.clean_node(magic_enum::enum_name(s)))
00289 .str());
00290 });
00291
```

12.19.4.7 for_each_tile() [1/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile (
             F && f )  [inline]
```

Apply a function for each tile infrastructure of the device.

Parameters

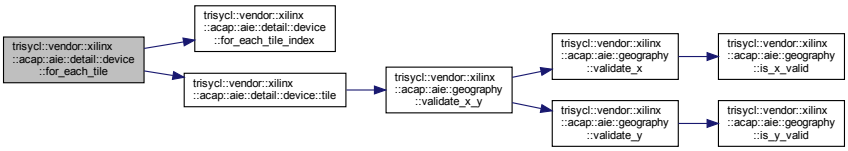
f	is a callable that is to be called like `f(x,y)` for each tile

Definition at line 110 of file device.hpp.

```
00110                                                {
00111     for_each_tile_index([&](auto x, auto y) { f(tile(x, y)); });
00112   };
```

References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_index(), trisycl::vendor::xilinx::acap::aie::detail::d
x, and trisycl::y.

Here is the call graph for this function:



12.19.4.8 for_each_tile() [2/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile (
            F && f )  [inline]
```

Apply a function for each tile infrastructure of the device.

Parameters

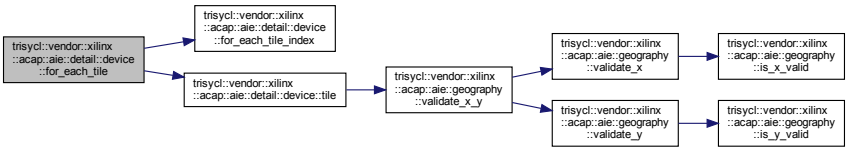
f	is a callable that is to be called like `f(x,y)` for each tile

Definition at line 110 of file device.hpp.

```
00110                                                {
00111     for_each_tile_index([&](auto x, auto y) { f(tile(x, y)); });
00112   };
```

References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_index(), trisycl::vendor::xilinx::acap::aie::detail::d
x, and trisycl::y.

Here is the call graph for this function:



12.19.4.9 for_each_tile_index() [1/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_index (
            F && f )  [inline]
```

Apply a function for each tile index of the device.

Parameters

f	is a callable that is to be called like `f(x,y)` for each tile

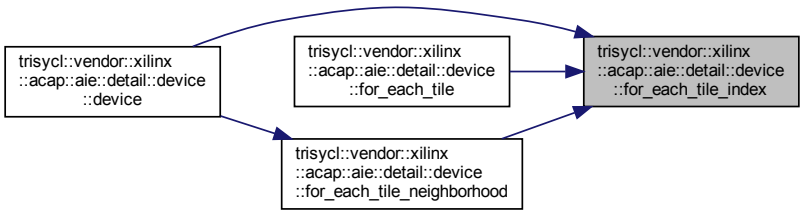
Definition at line 98 of file device.hpp.

```
00098                                                    {
00099     for (auto [x, y] :
00100          ranges::views::cartesian_product(ranges::views::iota(0, geo::x_size),
00101                                           ranges::views::iota(0, geo::y_size)))
00102       f(x, y);
00103   };
```

References x, trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_size, trisycl::y, and trisycl::vendor::xilinx::acap::aie::geograph

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::device(), trisycl::vendor::xilinx::acap::aie::detail::device< La and trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_neighborhood().

Here is the caller graph for this function:



12.19.4.10 for_each_tile_index() [2/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_index (
            F && f )  [inline]
```

Apply a function for each tile index of the device.

Parameters

f	is a callable that is to be called like `f(x,y)` for each tile

Definition at line 98 of file device.hpp.

```
00098                                                    {
00099     for (auto [x, y] :
00100          ranges::views::cartesian_product(ranges::views::iota(0, geo::x_size),
00101                                           ranges::views::iota(0, geo::y_size)))
00102       f(x, y);
00103   };
```

References x, trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_size, trisycl::y, and trisycl::vendor::xilinx::acap::aie::geograph

12.19.4.11 for_each_tile_neighborhood() [1/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_neighborhood
(
             F && f )  [inline]
```

Apply a function on all the AXI stream of the neighborhood of each tile.

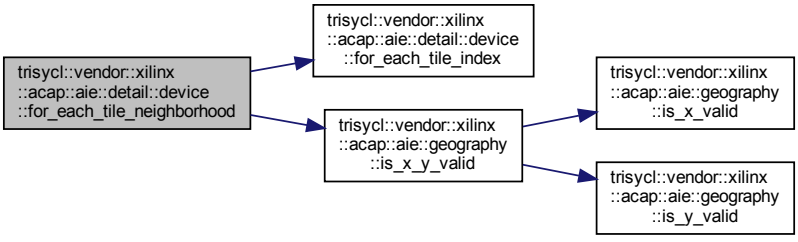
Definition at line 212 of file device.hpp.

```
00212                                                                {
00213     for_each_tile_index([&](auto x, auto y) {
00214       // No CTAD yet with Boost::Hana and Clang++-10 (but works with g++-9)
00215       boost::hana::for_each(
00216           geo::core_axi_stream_switch::interconnect, [&](auto connections) {
00217             auto [dx, dy, master_range, slave_range] = connections;
00218             if (geo::is_x_y_valid(x + dx, y + dy))
00219               for (auto [o, i] :  ranges::views::zip(master_range, slave_range))
00220                 f(x, y, x + dx, y + dy, o, i);
00221           });
00222     });
00223   }
```

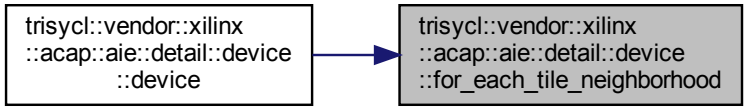
References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_index(), i, trisycl::vendor::xilinx::acap::aie::geogra trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_y_valid(), x, and trisycl::y.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::device().

Here is the call graph for this function:



Here is the caller graph for this function:



12.19.4.12 for_each_tile_neighborhood() [2/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_neighborhood
(
              F && f )  [inline]
```

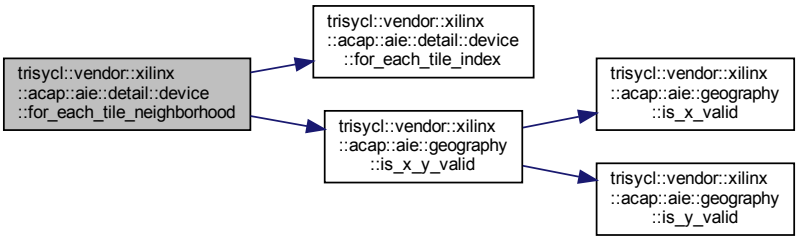
Apply a function on all the AXI stream of the neighborhood of each tile.

Definition at line 212 of file device.hpp.

```
00212                                                                    {
00213     for_each_tile_index([&](auto x, auto y) {
00214       // No CTAD yet with Boost::Hana and Clang++-10 (but works with g++-9)
00215       boost::hana::for_each(
00216           geo::core_axi_stream_switch::interconnect, [&](auto connections) {
00217             auto [dx, dy, master_range, slave_range] = connections;
00218             if (geo::is_x_y_valid(x + dx, y + dy))
00219               for (auto [o, i] :  ranges::views::zip(master_range, slave_range))
00220                 f(x, y, x + dx, y + dy, o, i);
00221           });
00222     });
00223   }
```

References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_index(), i, trisycl::vendor::xilinx::acap::aie::geogra
trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_y_valid(), x, and trisycl::y.

Here is the call graph for this function:



12.19.4.13 for_each_tile_x_index() [1/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_x_index (
            F && f )  [inline]
```

Apply a function for each x tile index of the device.

Parameters

f	is a callable that is to be called like `f(x)` for each horizontal index value

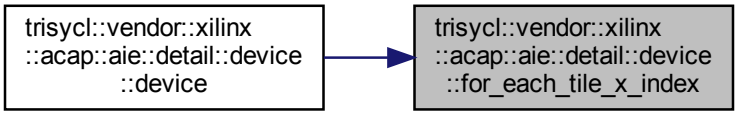
Definition at line 119 of file device.hpp.

```
00119                                                                    {
00120     for (auto x :  ranges::views::iota(0, geo::x_size))
00121       f(x);
00122   };
```

References x, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_size.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::device().

Here is the caller graph for this function:



12.19.4.14 for_each_tile_x_index() [2/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_x_index (
            F && f )  [inline]
```

Apply a function for each x tile index of the device.

Parameters

f	is a callable that is to be called like `f(x)` for each horizontal index value

Definition at line 119 of file device.hpp.

```
00119                                                                    {
```

```
00120     for (auto x :  ranges::views::iota(0, geo::x_size))
00121       f(x);
00122   };
```

References x, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::x_size.

12.19.4.15 for_each_tile_y_index() [1/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_y_index (
            F && f )  [inline]
```

Apply a function for each y tile index of the device.

Parameters

f	is a callable that is to be called like `f(x)` for each vertical index value

Definition at line 129 of file device.hpp.

```
00129                                                    {
00130     for (auto y :  ranges::views::iota(0, geo::y_size))
00131       f(y);
00132   };
```

References trisycl::y, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::y_size.

12.19.4.16 for_each_tile_y_index() [2/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_y_index (
            F && f )  [inline]
```

Apply a function for each y tile index of the device.

Parameters

f	is a callable that is to be called like `f(x)` for each vertical index value

Definition at line 129 of file device.hpp.

```
00129                                                    {
00130     for (auto y :  ranges::views::iota(0, geo::y_size))
00131       f(y);
00132   };
```

References trisycl::y, and trisycl::vendor::xilinx::acap::aie::geography< Layout >::y_size.

12.19.4.17 mem() [1/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::mem (
            int x,
            int y )  [inline]
```

Shortcut to access to the common infrastructure part of a tile memory.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Exceptions

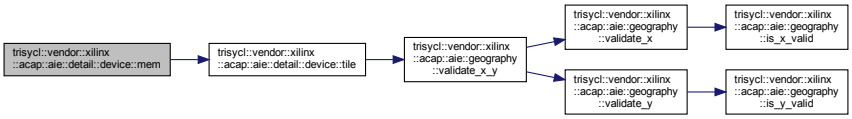
trisycl::runtime_error	if the coordinate is invalid

Definition at line 142 of file device.hpp.

```
00142                                {
00143     return tile(x, y).mem();
00144   }
```

References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::tile(), x, and trisycl::y.

Here is the call graph for this function:



12.19.4.18 mem() [2/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::mem (
            int x,
            int y )  [inline]
```

Shortcut to access to the common infrastructure part of a tile memory.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Exceptions

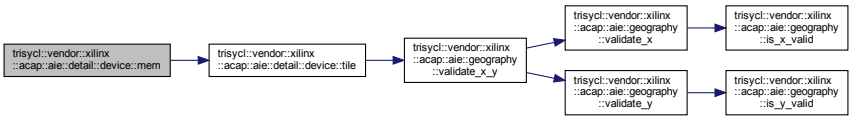
trisycl::runtime_error	if the coordinate is invalid

Definition at line 142 of file device.hpp.

```
00142                                {
00143     return tile(x, y).mem();
00144   }
```

References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::tile(), x, and trisycl::y.

Here is the call graph for this function:



12.19.4.19 shim() [1/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::shim (
            int x )  [inline]
```

Access to the shim tile.

Parameters

in	*x*	is the horizontal coordinate of the shim tile

Exceptions

trisycl::runtime_error	if the coordinate is invalid

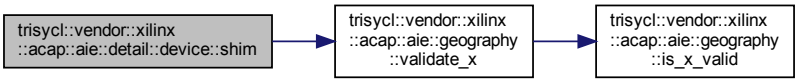
Definition at line 158 of file device.hpp.

```
00158                       {
00159     geo::validate_x(x);
00160     return shims[x];
00161   }
```

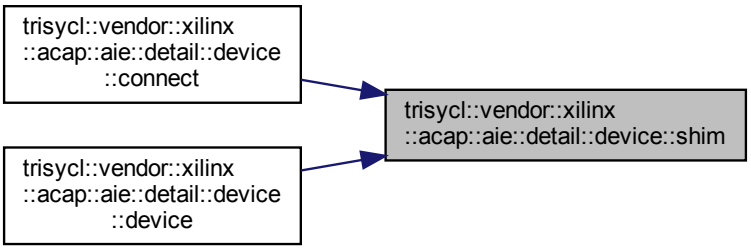
References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::shims, trisycl::vendor::xilinx::acap::aie::geography< Layout >:: and x.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::connect(), and trisycl::vendor::xilinx::acap::aie::detail::device

Here is the call graph for this function:



Here is the caller graph for this function:



12.19.4.20 shim() [2/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::shim (
            int x )  [inline]
```

Access to the shim tile.

Parameters

in	*x*	is the horizontal coordinate of the shim tile

Exceptions

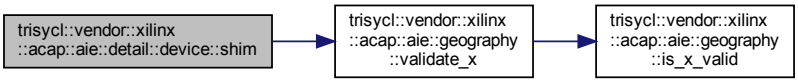
trisycl::runtime_error	if the coordinate is invalid

Definition at line 158 of file device.hpp.

```
00158                       {
00159     geo::validate_x(x);
00160     return shims[x];
00161   }
```

References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::shims, trisycl::vendor::xilinx::acap::aie::geography< Layout >:: and x.

Here is the call graph for this function:



12.19.4.21 tile() [1/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::tile (
            int x,
            int y )  [inline]
```

Access to the common infrastructure part of a tile.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Exceptions

trisycl::runtime_error	if the coordinate is invalid

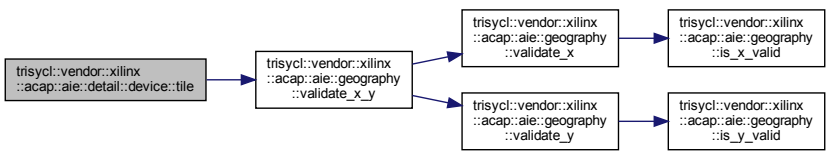
Definition at line 88 of file device.hpp.

```
00088                                {
00089     geo::validate_x_y(x, y);
00090     return ti[y][x];
00091   }
```

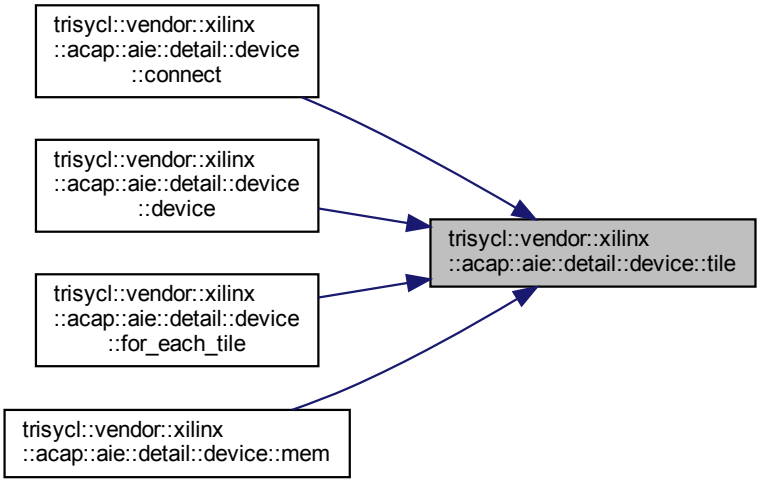
References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::ti, trisycl::vendor::xilinx::acap::aie::geography< Layout >::valid
x, and trisycl::y.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::connect(), trisycl::vendor::xilinx::acap::aie::detail::device< L
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile(), and trisycl::vendor::xilinx::acap::aie::detail::device< Layout

Here is the call graph for this function:



Here is the caller graph for this function:



12.19.4.22 tile() [2/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::tile (
            int x,
            int y )  [inline]
```

Access to the common infrastructure part of a tile.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Exceptions

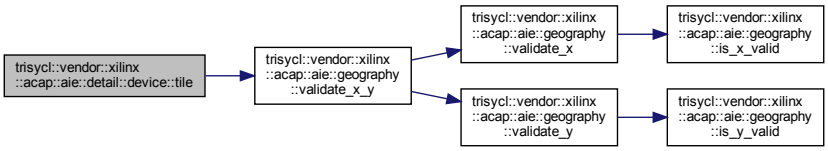
trisycl::runtime_error	if the coordinate is invalid

Definition at line 88 of file device.hpp.

```
00088                                {
00089     geo::validate_x_y(x, y);
00090     return ti[y][x];
00091   }
```

References trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::ti, trisycl::vendor::xilinx::acap::aie::geography< Layout >::valid
x, and trisycl::y.

Here is the call graph for this function:



12.19.5 Member Data Documentation

12.19.5.1 cs

```
template<typename Layout >
cascade_stream<geo> trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::cs
```

The cascade stream infrastructure of the CGRA.

Todo remove and distribute it instead

Todo remove and distribute it instead

Definition at line 66 of file device.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::cascade().

12.19.5.2 fiber_executor

```
template<typename Layout >
::trisycl::detail::fiber_pool trisycl::vendor::xilinx::acap::aie::detail::device< Layout >↩
::fiber_executor
```

Initial value:

```
{
    1, ::trisycl::detail::fiber_pool::sched::round_robin, false
  }
```

A fiber pool executor to run the infrastructure powered by TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS std::thread.

Definition at line 70 of file device.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::device().

12.19.5.3 shims

```
template<typename Layout >
shim_tile<device> trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::shims[geo::x_size]
```

The shim tiles on the lower row of the tile array.

For now we consider only homogeneous shim tiles.

Definition at line 150 of file device.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::shim().

12.19.5.4 ti

```
template<typename Layout >
aie::tile_infrastructure<geo> trisycl::vendor::xilinx::acap::aie::detail::device< Layout >↩
::ti[geo::y_size][geo::x_size]
```

Keep track of all the (non detail) infrastructure tiles of this device.

Definition at line 78 of file device.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::tile().

The documentation for this struct was generated from the following files:

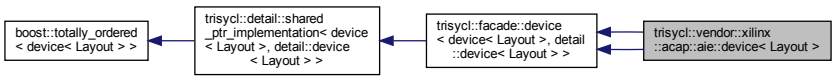
- include/sycl/vendor/Xilinx/acap/aie/device/detail/device.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/device/detail/device.hpp

12.20 trisycl::vendor::xilinx::acap::aie::device< Layout > Class Template Reference

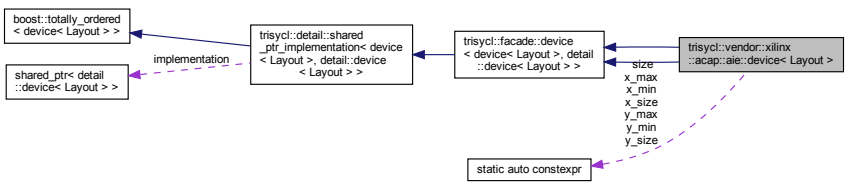
Create a SYCL-like device view of an AI Engine CGRA with some layout.

```
#include <device.hpp>
```

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::device< Layout >:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::device< Layout >:



Public Types

- using geo = typename dd::geo

  Expose some useful internal implementation. Look at them for their documentation.
- using cass = typename dd::cass
- using csp = typename dd::csp
- using cmp = typename dd::cmp
- using sass = typename dd::sass
- using ssp = typename dd::ssp
- using smp = typename dd::smp
- using geo = typename dd::geo

  Expose some useful internal implementation. Look at them for their documentation.
- using cass = typename dd::cass
- using csp = typename dd::csp
- using cmp = typename dd::cmp
- using sass = typename dd::sass
- using ssp = typename dd::ssp
- using smp = typename dd::smp

Public Member Functions

- device ()

  The default constructor makes a new device.
- template<typename F >
  void for_each_tile_index (F &&f)

  Apply a function for each tile index of the device.
- template<typename F >
  void for_each_tile (F &&f)

  Apply a function for each tile infrastructure of the device.
- template<typename F >
  void for_each_tile_x_index (F &&f)

  Apply a function for each x tile index of the device.
- template<typename F >
  void for_each_tile_y_index (F &&f)

  Apply a function for each y tile index of the device.
- auto & tile (int x, int y)

  Access to the common infrastructure part of a tile.
- auto & mem (int x, int y)

  Access to the common infrastructure part of a tile memory.
- auto & shim (int x)

  Access to the shim tile.
- auto & cascade ()

  Access the cascade connections.
- auto queue ()

  Create a queue on this device.
- template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template< typename Device, int X, int Y > typename Memory = acap::aie::memory>
  auto program ()

  Create a program execution for this device.
- template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template< typename Device, int X, int Y > typename Memory = acap::aie::memory>
  void run ()

Shortcut to run synchronously a program on this queue.

- template<typename Invocable >
  void run (Invocable &&f)

  Shortcut to run synchronously an heterogeneous invocable on this queue.
- template<typename Invocable >
  void uniform_run (Invocable &&f)

  Shortcut to run synchronously a uniform invocable on this queue.
- template<typename SrcPort , typename DstPort >
  void connect (SrcPort &&src, DstPort &&dst)

  Connect the ports of 2 tiles or shims together with an hyperspace switched circuit, jumping over the underlying routing infrastructure.
- template<typename F >
  void for_each_tile_neighborhood (F &&f)

  Apply a invocable on all the AXI stream of the neighborhood of each tile.
- auto display (const std::string &file_name={})

  Display the device layout.
- device ()

  The default constructor makes a new device.
- template<typename F >
  void for_each_tile_index (F &&f)

  Apply a function for each tile index of the device.
- template<typename F >
  void for_each_tile (F &&f)

  Apply a function for each tile infrastructure of the device.
- template<typename F >
  void for_each_tile_x_index (F &&f)

  Apply a function for each x tile index of the device.
- template<typename F >
  void for_each_tile_y_index (F &&f)

  Apply a function for each y tile index of the device.
- auto & tile (int x, int y)

  Access to the common infrastructure part of a tile.
- auto & mem (int x, int y)

  Access to the common infrastructure part of a tile memory.
- auto & shim (int x)

  Access to the shim tile.
- auto & cascade ()

  Access the cascade connections.
- auto queue ()

  Create a queue on this device.
- template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template< typename Device, int X, int Y > typename Memory = acap::aie::memory>
  auto program ()

  Create a program execution for this device.
- template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template< typename Device, int X, int Y > typename Memory = acap::aie::memory>
  void run ()

  Shortcut to run synchronously a program on this queue.
- template<typename Invocable >
  void run (Invocable &&f)

  Shortcut to run synchronously an heterogeneous invocable on this queue.
- template<typename Invocable >
  void uniform_run (Invocable &&f)

Shortcut to run synchronously a uniform invocable on this queue.

- template<typename SrcPort , typename DstPort >
  void connect (SrcPort &&src, DstPort &&dst)

  Connect the ports of 2 tiles or shims together with an hyperspace switched circuit, jumping over the underlying routing infrastructure.
- template<typename F >
  void for_each_tile_neighborhood (F &&f)

  Apply a invocable on all the AXI stream of the neighborhood of each tile.
- auto display (const std::string &file_name={})

  Display the device layout.

Static Public Attributes

- static auto constexpr x_min = geo::x_min

  Minimum horizontal coordinate.
- static auto constexpr x_max = geo::x_max

  Maximum horizontal coordinate.
- static auto constexpr x_size = geo::x_size

  Number of tile horizontally in the CGRA.
- static auto constexpr y_min = geo::y_min

  Minimum vertical coordinate.
- static auto constexpr y_max = geo::y_max

  Maximum vertical coordinate.
- static auto constexpr y_size = geo::y_size

  Number of tile vertically in the CGRA.
- static auto constexpr size = geo::size

  Total number of tiles in the CGRA.

Private Types

- using dd = detail::device< Layout >

  The type encapsulating the implementation.
- using facade_t = facade::device< device< Layout >, dd >

  The façade used to implement part of the use facing type.
- using dd = detail::device< Layout >

  The type encapsulating the implementation.
- using facade_t = facade::device< device< Layout >, dd >

  The façade used to implement part of the use facing type.

Additional Inherited Members

12.20.1 Detailed Description

template<typename Layout>
class trisycl::vendor::xilinx::acap::aie::device< Layout >

Create a SYCL-like device view of an AI Engine CGRA with some layout.

Parameters

Layout	is the layout description of the machine to instantiate with the physical size

Definition at line 30 of file device.hpp.

12.20.2 Member Typedef Documentation

12.20.2.1 cass [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::cass = typename dd::cass
```

Definition at line 44 of file device.hpp.

12.20.2.2 cass [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::cass = typename dd::cass
```

Definition at line 44 of file device.hpp.

12.20.2.3 cmp [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::cmp = typename dd::cmp
```

Definition at line 46 of file device.hpp.

12.20.2.4 cmp [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::cmp = typename dd::cmp
```

Definition at line 46 of file device.hpp.

12.20.2.5 csp [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::csp = typename dd::csp
```

Definition at line 45 of file device.hpp.

12.20.2.6 csp [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::csp = typename dd::csp
```

Definition at line 45 of file device.hpp.

12.20.2.7 dd [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::dd = detail::device<Layout>  [private]
```

The type encapsulating the implementation.

Definition at line 32 of file device.hpp.

12.20.2.8 dd [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::dd = detail::device<Layout>  [private]
```

The type encapsulating the implementation.

Definition at line 32 of file device.hpp.

12.20.2.9 facade_t [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::facade_t = facade::device<device<Layout>,
dd>  [private]
```

The façade used to implement part of the use facing type.

Definition at line 35 of file device.hpp.

12.20.2.10 facade_t [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::facade_t = facade::device<device<Layout>,
dd>  [private]
```

The façade used to implement part of the use facing type.

Definition at line 35 of file device.hpp.

12.20.2.11 geo [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::geo = typename dd::geo
```

Expose some useful internal implementation. Look at them for their documentation.

Definition at line 43 of file device.hpp.

12.20.2.12 geo [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::geo = typename dd::geo
```

Expose some useful internal implementation. Look at them for their documentation.

Definition at line 43 of file device.hpp.

12.20.2.13 sass [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::sass = typename dd::sass
```

Definition at line 47 of file device.hpp.

12.20.2.14 sass [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::sass = typename dd::sass
```

Definition at line 47 of file device.hpp.

12.20.2.15 smp [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::smp = typename dd::smp
```

Definition at line 49 of file device.hpp.

12.20.2.16 smp [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::smp = typename dd::smp
```

Definition at line 49 of file device.hpp.

12.20.2.17 ssp [1/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::ssp = typename dd::ssp
```

Definition at line 48 of file device.hpp.

12.20.2.18 ssp [2/2]

```
template<typename Layout >
using trisycl::vendor::xilinx::acap::aie::device< Layout >::ssp = typename dd::ssp
```

Definition at line 48 of file device.hpp.

12.20.3 Constructor & Destructor Documentation

12.20.3.1 device() [1/2]

```
template<typename Layout >
trisycl::vendor::xilinx::acap::aie::device< Layout >::device ( )  [inline]
```

The default constructor makes a new device.

Definition at line 75 of file device.hpp.

```
00076       :  facade_t { std::make_shared<dd>() } {}
```

12.20.3.2 device() [2/2]

```
template<typename Layout >
trisycl::vendor::xilinx::acap::aie::device< Layout >::device ( )  [inline]
```

The default constructor makes a new device.

Definition at line 75 of file device.hpp.
```
00076       :  facade_t { std::make_shared<dd>() } {}
```

12.20.4 Member Function Documentation

12.20.4.1 cascade() [1/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::device< Layout >::cascade ( )  [inline]
```

Access the cascade connections.

Todo To remove?

Definition at line 144 of file device.hpp.
```
00144 { return implementation->cascade(); }
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation.

12.20.4.2 cascade() [2/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::device< Layout >::cascade ( )  [inline]
```

Access the cascade connections.

Todo To remove?

Definition at line 144 of file device.hpp.
```
00144 { return implementation->cascade(); }
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation.

12.20.4.3 connect() [1/2]

```
template<typename Layout >
template<typename SrcPort , typename DstPort >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::connect (
              SrcPort && src,
              DstPort && dst )  [inline]
```

Connect the ports of 2 tiles or shims together with an hyperspace switched circuit, jumping over the underlying routing infrastructure.

This is used to speed up communication in emulation and to skip the need of a routing planner.

Todo To deprecate and implement in another way?

Parameters

in	*T*	is the type of the data to be transferred
in	*SrcPort*	is the type of the source port, such as port::tile or port::shim
in	*DstPort*	is the type of the destination port, such as port::tile or port::shim

Exceptions

trisycl::runtime_error	if some coordinates or port numbers are invalid

Todo Refactor, make the difference between user & physical ports

Definition at line 223 of file device.hpp.

```
00223                                        {
00224     implementation->connect(std::forward<SrcPort>(src),
00225                             std::forward<DstPort>(dst));
00226   }
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation, and src.

12.20.4.4 connect() [2/2]

```
template<typename Layout >
template<typename SrcPort , typename DstPort >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::connect (
            SrcPort && src,
            DstPort && dst )  [inline]
```

Connect the ports of 2 tiles or shims together with an hyperspace switched circuit, jumping over the underlying routing infrastructure.

This is used to speed up communication in emulation and to skip the need of a routing planner.

Todo To deprecate and implement in another way?

Parameters

in	*T*	is the type of the data to be transferred
in	*SrcPort*	is the type of the source port, such as port::tile or port::shim
in	*DstPort*	is the type of the destination port, such as port::tile or port::shim

Exceptions

trisycl::runtime_error	if some coordinates or port numbers are invalid

Todo Refactor, make the difference between user & physical ports

Definition at line 223 of file device.hpp.

```
00223                                       {
00224     implementation->connect(std::forward<SrcPort>(src),
00225                             std::forward<DstPort>(dst));
00226   }
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation, and src.

12.20.4.5 display() [1/2]

```
template<typename Layout >
auto trisycl::vendor::xilinx::acap::aie::device< Layout >::display (
             const std::string & file_name = {} )  [inline]
```

Display the device layout.

Parameters

in	*file_name*	is the name of the file to write the LaTeX drawing into or use std::cerr by default

Definition at line 239 of file device.hpp.

```
00239                                       {}) {
00240     return implementation->display(file_name);
00241   }
```

12.20.4.6 display() [2/2]

```
template<typename Layout >
auto trisycl::vendor::xilinx::acap::aie::device< Layout >::display (
             const std::string & file_name = {} )  [inline]
```

Display the device layout.

Parameters

in	*file_name*	is the name of the file to write the LaTeX drawing into or use std::cerr by default

Definition at line 239 of file device.hpp.

```
00239                                       {}) {
00240     return implementation->display(file_name);
00241   }
```

12.20.4.7 for_each_tile() [1/2]

```
template<typename Layout >
template<typename F >
```

```
void trisycl::vendor::xilinx::acap::aie::device< Layout >::for_each_tile (
            F && f )  [inline]
```

Apply a function for each tile infrastructure of the device.

Parameters

f	is a callable that is to be called like `f(x,y)` for each tile

Definition at line 92 of file device.hpp.

```
00092                                                {
00093     implementation->for_each_tile(std::forward<F>(f));
00094   };
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation.

12.20.4.8 for_each_tile() [2/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::for_each_tile (
            F && f )  [inline]
```

Apply a function for each tile infrastructure of the device.

Parameters

f	is a callable that is to be called like `f(x,y)` for each tile

Definition at line 92 of file device.hpp.

```
00092                                                {
00093     implementation->for_each_tile(std::forward<F>(f));
00094   };
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation.

12.20.4.9 for_each_tile_index() [1/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::for_each_tile_index (
            F && f )  [inline]
```

Apply a function for each tile index of the device.

Parameters

f	is a callable that is to be called like `f(x,y)` for each tile

Definition at line 83 of file device.hpp.

```
00083                                                 {
00084     implementation->for_each_tile_index(std::forward<F>(f));
00085   };
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation.

12.20.4.10 for_each_tile_index() [2/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::for_each_tile_index (
             F && f )  [inline]
```

Apply a function for each tile index of the device.

Parameters

f	is a callable that is to be called like `f(x,y)` for each tile

Definition at line 83 of file device.hpp.

```
00083                                                 {
00084     implementation->for_each_tile_index(std::forward<F>(f));
00085   };
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation.

12.20.4.11 for_each_tile_neighborhood() [1/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::for_each_tile_neighborhood (
             F && f )  [inline]
```

Apply a invocable on all the AXI stream of the neighborhood of each tile.

Definition at line 230 of file device.hpp.

```
00230                                                           {
00231     implementation->for_each_tile_neighborhood(std::forward<F>(f));
00232   }
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation.

12.20.4.12 for_each_tile_neighborhood() [2/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::for_each_tile_neighborhood (
            F && f )  [inline]
```

Apply a invocable on all the AXI stream of the neighborhood of each tile.

Definition at line 230 of file device.hpp.

```
00230                                                          {
00231     implementation->for_each_tile_neighborhood(std::forward<F>(f));
00232   }
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation.

12.20.4.13 for_each_tile_x_index() [1/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::for_each_tile_x_index (
            F && f )  [inline]
```

Apply a function for each x tile index of the device.

Parameters

f	is a callable that is to be called like `f(x)` for each horizontal index value

Definition at line 101 of file device.hpp.

```
00101                                                  {
00102     implementation->for_each_tile_x_index(std::forward<F>(f));
00103   };
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation.

12.20.4.14 for_each_tile_x_index() [2/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::for_each_tile_x_index (
            F && f )  [inline]
```

Apply a function for each x tile index of the device.

Parameters

f	is a callable that is to be called like `f(x)` for each horizontal index value

Definition at line 101 of file device.hpp.

```
00101                                                    {
00102     implementation->for_each_tile_x_index(std::forward<F>(f));
00103   };
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation.

12.20.4.15 for_each_tile_y_index() [1/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::for_each_tile_y_index (
            F && f )  [inline]
```

Apply a function for each y tile index of the device.

Parameters

f	is a callable that is to be called like `f(x)` for each vertical index value

Definition at line 110 of file device.hpp.

```
00110                                                    {
00111     implementation->for_each_tile_y_index(std::forward<F>(f));
00112   };
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation.

12.20.4.16 for_each_tile_y_index() [2/2]

```
template<typename Layout >
template<typename F >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::for_each_tile_y_index (
            F && f )  [inline]
```

Apply a function for each y tile index of the device.

Parameters

f	is a callable that is to be called like `f(x)` for each vertical index value

Definition at line 110 of file device.hpp.

```
00110                                                    {
00111     implementation->for_each_tile_y_index(std::forward<F>(f));
00112   };
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation.

12.20.4.17 mem() [1/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::device< Layout >::mem (
            int x,
            int y )  [inline]
```

Access to the common infrastructure part of a tile memory.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Exceptions

trisycl::runtime_error	if the coordinate is invalid

Definition at line 132 of file device.hpp.

```
00132 { return implementation->mem(x, y); }
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation, x, and trisycl::y.

12.20.4.18 mem() [2/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::device< Layout >::mem (
            int x,
            int y )  [inline]
```

Access to the common infrastructure part of a tile memory.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Exceptions

trisycl::runtime_error	if the coordinate is invalid

Definition at line 132 of file device.hpp.

```
00132 { return implementation->mem(x, y); }
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation, x, and trisycl::y.

12.20.4.19 program() [1/2]

```
template<typename Layout >
template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template<
typename Device, int X, int Y > typename Memory = acap::aie::memory>
auto trisycl::vendor::xilinx::acap::aie::device< Layout >::program ( )  [inline]
```

Create a program execution for this device.

Parameters

Tile	is the description of the program tiles to instantiate. By default each tile will run an empty program.
Memory	is the description of the machine memory modules. By default the machine has empty memory modules.

Definition at line 161 of file device.hpp.

```
00161                 {
00162     return aie::program<device, Tile, Memory> { *this };
00163   }
```

12.20.4.20 program() [2/2]

```
template<typename Layout >
template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template<
typename Device, int X, int Y > typename Memory = acap::aie::memory>
auto trisycl::vendor::xilinx::acap::aie::device< Layout >::program ( )  [inline]
```

Create a program execution for this device.

Parameters

Tile	is the description of the program tiles to instantiate. By default each tile will run an empty program.
Memory	is the description of the machine memory modules. By default the machine has empty memory modules.

Definition at line 161 of file device.hpp.

```
00161                 {
00162     return aie::program<device, Tile, Memory> { *this };
00163   }
```

12.20.4.21 queue() [1/2]

```
template<typename Layout >
auto trisycl::vendor::xilinx::acap::aie::device< Layout >::queue ( )  [inline]
```

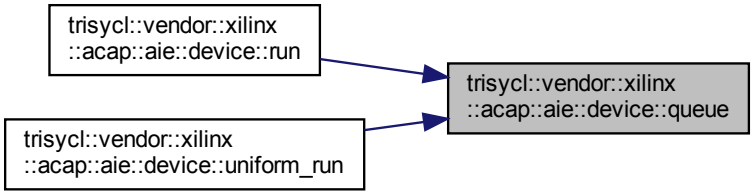
Create a queue on this device.

Definition at line 147 of file device.hpp.

```
00147 { return vendor::xilinx::acap::aie::queue { *this }; }
```

Referenced by trisycl::vendor::xilinx::acap::aie::device< Layout >::run(), and trisycl::vendor::xilinx::acap::aie::device< Layout >::unifo

Here is the caller graph for this function:



12.20.4.22 queue() [2/2]

```
template<typename Layout >
auto trisycl::vendor::xilinx::acap::aie::device< Layout >::queue ( )  [inline]
```

Create a queue on this device.

Definition at line 147 of file device.hpp.

```
00147 { return vendor::xilinx::acap::aie::queue { *this }; }
```

12.20.4.23 run() [1/4]

```
template<typename Layout >
template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template<
typename Device, int X, int Y > typename Memory = acap::aie::memory>
void trisycl::vendor::xilinx::acap::aie::device< Layout >::run ( )  [inline]
```

Shortcut to run synchronously a program on this queue.

Parameters

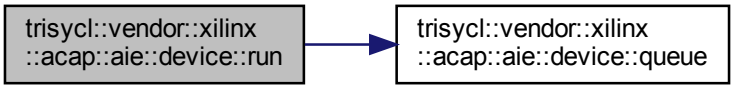
Tile	is the description of the program tiles to instantiate. By default each tile will run an empty program.
Memory	is the description of the machine memory modules. By default the machine has empty memory modules.

Definition at line 177 of file device.hpp.

```
00177             {
00178     queue().template run<Tile, Memory>();
00179   }
```

References trisycl::vendor::xilinx::acap::aie::device< Layout >::queue().

Here is the call graph for this function:



12.20.4.24 run() [2/4]

```
template<typename Layout >
template<template< typename Device, int X, int Y > typename Tile = acap::aie::tile, template<
typename Device, int X, int Y > typename Memory = acap::aie::memory>
void trisycl::vendor::xilinx::acap::aie::device< Layout >::run ( )  [inline]
```

Shortcut to run synchronously a program on this queue.

Parameters

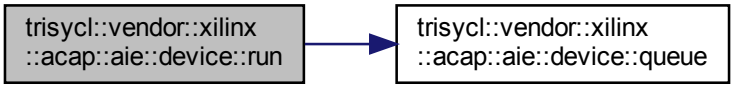
Tile	is the description of the program tiles to instantiate. By default each tile will run an empty program.
Memory	is the description of the machine memory modules. By default the machine has empty memory modules.

Definition at line 177 of file device.hpp.

```
00177             {
00178     queue().template run<Tile, Memory>();
00179   }
```

References trisycl::vendor::xilinx::acap::aie::device< Layout >::queue().

Here is the call graph for this function:



12.20.4.25 run() [3/4]

```
template<typename Layout >
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::run (
             Invocable && f )  [inline]
```

Shortcut to run synchronously an heterogeneous invocable on this queue.

Parameters

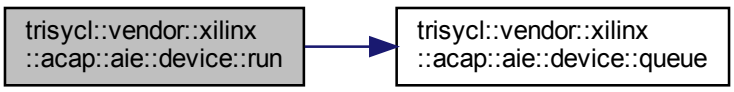
f	is an invocable taking an heterogeneous tile handler

Definition at line 186 of file device.hpp.

```
00186                                                    {
00187     queue().run(std::forward<Invocable>(f));
00188   }
```

References trisycl::vendor::xilinx::acap::aie::device< Layout >::queue().

Here is the call graph for this function:



12.20.4.26 run() [4/4]

```
template<typename Layout >
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::run (
             Invocable && f )  [inline]
```

Shortcut to run synchronously an heterogeneous invocable on this queue.

Parameters

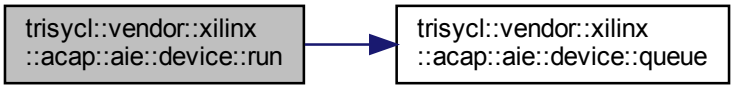
f	is an invocable taking an heterogeneous tile handler

Definition at line 186 of file device.hpp.

```
00186                                                    {
00187     queue().run(std::forward<Invocable>(f));
00188   }
```

References trisycl::vendor::xilinx::acap::aie::device< Layout >::queue().

Here is the call graph for this function:



12.20.4.27 shim() [1/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::device< Layout >::shim (
            int x )  [inline]
```

Access to the shim tile.

Parameters

in	*x*	is the horizontal coordinate of the shim tile

Exceptions

trisycl::runtime_error	if the coordinate is invalid

Definition at line 140 of file device.hpp.

```
00140 { return implementation->shim(x); }
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation, and x.

12.20.4.28 shim() [2/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::device< Layout >::shim (
            int x )  [inline]
```

Access to the shim tile.

Parameters

in	*x*	is the horizontal coordinate of the shim tile

Exceptions

trisycl::runtime_error	if the coordinate is invalid

Definition at line 140 of file device.hpp.

```
00140 { return implementation->shim(x); }
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation, and x.

12.20.4.29 tile() [1/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::device< Layout >::tile (
            int x,
            int y )  [inline]
```

Access to the common infrastructure part of a tile.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Exceptions

trisycl::runtime_error	if the coordinate is invalid

Definition at line 122 of file device.hpp.

```
00122 { return implementation->tile(x, y); }
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation, x, and trisycl::y.

12.20.4.30 tile() [2/2]

```
template<typename Layout >
auto & trisycl::vendor::xilinx::acap::aie::device< Layout >::tile (
            int x,
            int y )  [inline]
```

Access to the common infrastructure part of a tile.

Parameters

in	*x*	is the horizontal tile coordinate
in	*y*	is the vertical tile coordinate

Exceptions

trisycl::runtime_error	if the coordinate is invalid

Definition at line 122 of file device.hpp.

```
00122 { return implementation->tile(x, y); }
```

References trisycl::detail::shared_ptr_implementation< device< Layout >, detail::device< Layout > >::implementation, x, and trisycl::y.

12.20.4.31 uniform_run() [1/2]

```
template<typename Layout >
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::uniform_run (
            Invocable && f )  [inline]
```

Shortcut to run synchronously a uniform invocable on this queue.

Parameters

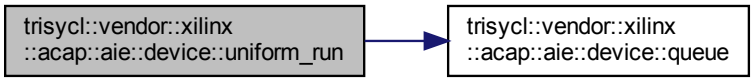
f	is an invocable taking a uniform tile handler

Definition at line 195 of file device.hpp.

```
00195                                                                    {
00196     queue().uniform_run(std::forward<Invocable>(f));
00197     queue().wait();
00198   }
```

References trisycl::vendor::xilinx::acap::aie::device< Layout >::queue().

Here is the call graph for this function:



12.20.4.32 uniform_run() [2/2]

```
template<typename Layout >
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::device< Layout >::uniform_run (
            Invocable && f )  [inline]
```

Shortcut to run synchronously a uniform invocable on this queue.

Parameters

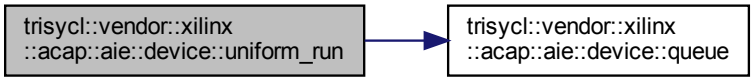
f	is an invocable taking a uniform tile handler

Definition at line 195 of file device.hpp.

```
00195                                                                    {
00196     queue().uniform_run(std::forward<Invocable>(f));
00197     queue().wait();
00198   }
```

References trisycl::vendor::xilinx::acap::aie::device< Layout >::queue().

Here is the call graph for this function:



12.20.5 Member Data Documentation

12.20.5.1 size

```
template<typename Layout >
auto constexpr trisycl::vendor::xilinx::acap::aie::device< Layout >::size = geo::size  [static],
[constexpr]
```

Total number of tiles in the CGRA.

Definition at line 72 of file device.hpp.

12.20.5.2 x_max

```
template<typename Layout >
auto constexpr trisycl::vendor::xilinx::acap::aie::device< Layout >::x_max = geo::x_max  [static],
[constexpr]
```

Maximum horizontal coordinate.

Definition at line 57 of file device.hpp.

12.20.5.3 x_min

```
template<typename Layout >
auto constexpr trisycl::vendor::xilinx::acap::aie::device< Layout >::x_min = geo::x_min  [static],
[constexpr]
```

Minimum horizontal coordinate.

Definition at line 54 of file device.hpp.

12.20.5.4 x_size

```
template<typename Layout >
auto constexpr trisycl::vendor::xilinx::acap::aie::device< Layout >::x_size = geo::x_size
[static], [constexpr]
```

Number of tile horizontally in the CGRA.

Definition at line 60 of file device.hpp.

12.20.5.5 y_max

```
template<typename Layout >
auto constexpr trisycl::vendor::xilinx::acap::aie::device< Layout >::y_max = geo::y_max  [static],
[constexpr]
```

Maximum vertical coordinate.

Definition at line 66 of file device.hpp.

12.20.5.6 y_min

```
template<typename Layout >
auto constexpr trisycl::vendor::xilinx::acap::aie::device< Layout >::y_min = geo::y_min  [static],
[constexpr]
```

Minimum vertical coordinate.

Definition at line 63 of file device.hpp.

12.20.5.7 y_size

```
template<typename Layout >
auto constexpr trisycl::vendor::xilinx::acap::aie::device< Layout >::y_size = geo::y_size
[static], [constexpr]
```

Number of tile vertically in the CGRA.

Definition at line 69 of file device.hpp.

The documentation for this class was generated from the following files:

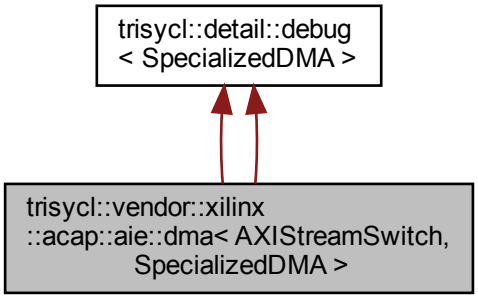
- include/sycl/vendor/Xilinx/acap/aie/device.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/device.hpp

12.21 trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA > Class Template Reference

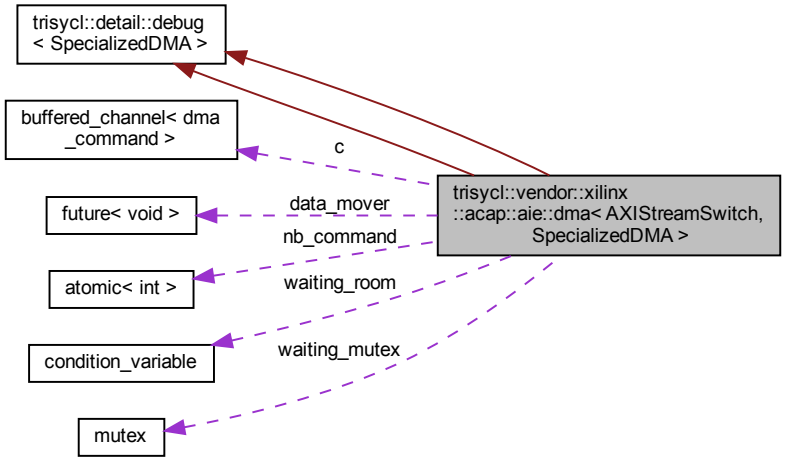
Generic DMA.

```
#include <dma.hpp>
```

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >:



Public Member Functions

- dma (::trisycl::detail::fiber_pool &fe)

  Start the DMA engine using an executor.
- ~dma ()

  Destructor handling the correct infrastructure shutdown.
- SpecializedDMA & wait ()

  Wait for the transfers to complete.
- dma (::trisycl::detail::fiber_pool &fe)

  Start the DMA engine using an executor.
- ~dma ()

  Destructor handling the correct infrastructure shutdown.
- SpecializedDMA & wait ()

  Wait for the transfers to complete.

Protected Member Functions

- template<typename Cmd >
  void push_command (Cmd &&command)

  Enqueue a DMA command.
- template<typename Cmd >
  void push_command (Cmd &&command)

  Enqueue a DMA command.

Private Types

- using dma_command = std::function< void(void)>

  Use std::span for 1D contiguous transfers only for now.
- using dma_command = std::function< void(void)>

  Use std::span for 1D contiguous transfers only for now.

Private Member Functions

- void join ()

  Wait for the fiber to complete.
- template<typename Invocable >
  void launch (::trisycl::detail::fiber_pool &fiber_executor, Invocable &&f)

  Launch the data mover on the execution pool.
- auto pop_command (auto &command)

  Dequeue a DMA command.
- void commit_command ()

  Notify a command has been processed.
- void join ()

  Wait for the fiber to complete.
- template<typename Invocable >
  void launch (::trisycl::detail::fiber_pool &fiber_executor, Invocable &&f)

  Launch the data mover on the execution pool.
- auto pop_command (auto &command)

  Dequeue a DMA command.
- void commit_command ()

  Notify a command has been processed.

Private Attributes

- boost::fibers::buffered_channel< dma_command > c { 8 }

  Queue for generic DMA commands.
- ::trisycl::detail::fiber_pool::future< void > data_mover

  To shepherd the fiber playing the data mover behind the DMA.
- std::atomic< int > nb_command { 0 }

  Count the in-flight commands.
- boost::fibers::condition_variable waiting_room

  To wait for the DMA command queue to drain.
- boost::fibers::mutex waiting_mutex

  To protect the waiting mechanism.

Additional Inherited Members

12.21.1 Detailed Description

template<typename AXIStreamSwitch, typename SpecializedDMA>
class trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >

Generic DMA.

Definition at line 103 of file dma.hpp.

12.21.2 Member Typedef Documentation

12.21.2.1 dma_command [1/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
using trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::dma_command
= std::function<void(void)>  [private]
```

Use std::span for 1D contiguous transfers only for now.

The type of commands executable by the DMA. Very generic for now

Definition at line 110 of file dma.hpp.

12.21.2.2 dma_command [2/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
using trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::dma_command
= std::function<void(void)>  [private]
```

Use std::span for 1D contiguous transfers only for now.

The type of commands executable by the DMA. Very generic for now

Definition at line 110 of file dma.hpp.

12.21.3 Constructor & Destructor Documentation

12.21.3.1 dma() [1/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::dma (
            ::trisycl::detail::fiber_pool & fe )  [inline]
```

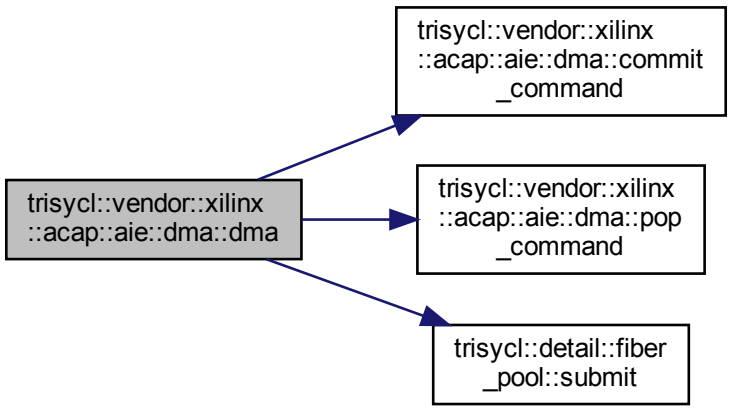
Start the DMA engine using an executor.

Definition at line 156 of file dma.hpp.

```
00156                                       {
00157     data_mover = fe.submit([&] {
00158       dma_command sp;
00159       for (;;) {
00160         // Read a DMA command
00161         if (this->pop_command(sp) != boost::fibers::channel_op_status::success)
00162           // The channel is closed, stop working
00163           break;
00164         // Execute the command
00165         sp();
00166         commit_command();
00167       }
00168     });
00169   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::commit_command(), trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::data_mover, trisycl::vendor::xilinx::acap::aie::dma< AX and trisycl::detail::fiber_pool::submit().

Here is the call graph for this function:



12.21.3.2 ~dma() [1/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::~dma ( )  [inline]
```

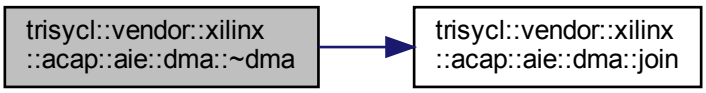
Destructor handling the correct infrastructure shutdown.

Definition at line 172 of file dma.hpp.

```
00172         {
00173     // Close the command queue so the DMA does not accept any work
00174     c.close();
00175     /* Wait for the commands to drain to avoid calling
00176 std::terminate on destruction...  */
00177     join();
00178   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::c, and trisycl::vendor::xilinx::acap::aie::dma

Here is the call graph for this function:



12.21.3.3 dma() [2/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::dma (
            ::trisycl::detail::fiber_pool & fe )  [inline]
```

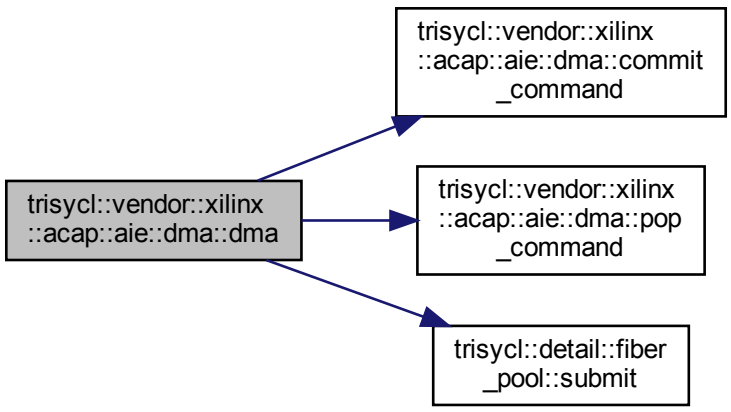
Start the DMA engine using an executor.

Definition at line 156 of file dma.hpp.

```
00156                                       {
00157     data_mover = fe.submit([&] {
00158       dma_command sp;
00159       for (;;) {
00160         // Read a DMA command
00161         if (this->pop_command(sp) != boost::fibers::channel_op_status::success)
00162           // The channel is closed, stop working
00163           break;
00164         // Execute the command
00165         sp();
00166         commit_command();
00167       }
00168     });
00169   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::commit_command(), trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::data_mover, trisycl::vendor::xilinx::acap::aie::dma< AX and trisycl::detail::fiber_pool::submit().

Here is the call graph for this function:



12.21.3.4 ~dma() [2/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::~dma ( )  [inline]
```

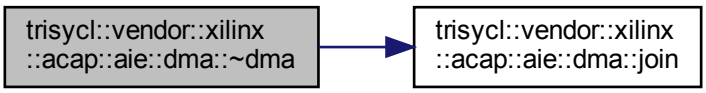
Destructor handling the correct infrastructure shutdown.

Definition at line 172 of file dma.hpp.

```
00172           {
00173     // Close the command queue so the DMA does not accept any work
00174     c.close();
00175     /* Wait for the commands to drain to avoid calling
00176 std::terminate on destruction...  */
00177     join();
00178   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::c, and trisycl::vendor::xilinx::acap::aie::dma

Here is the call graph for this function:



12.21.4 Member Function Documentation

12.21.4.1 commit_command() [1/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
void trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::commit_↩
command ( )  [inline], [private]
```

Notify a command has been processed.

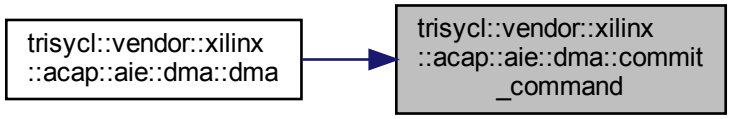
Definition at line 141 of file dma.hpp.

```
00141                           {
00142     if (--nb_command == 0)
00143       // Warn anyone waiting for the DMA queue to drain it might be empty now
00144       waiting_room.notify_all();
00145   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::nb_command, and trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::waiting_room.

Referenced by trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::dma().

Here is the caller graph for this function:



12.21.4.2 commit_command() [2/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
void trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::commit_↩
command ( )  [inline], [private]
```

Notify a command has been processed.

Definition at line 141 of file dma.hpp.

```
00141                         {
00142     if (--nb_command == 0)
00143       // Warn anyone waiting for the DMA queue to drain it might be empty now
00144       waiting_room.notify_all();
00145   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::nb_command, and trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::waiting_room.

12.21.4.3 join() [1/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
void trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::join ( )
[inline], [private]
```

Wait for the fiber to complete.

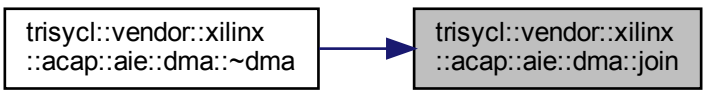
Definition at line 128 of file dma.hpp.

```
00128               {
00129     // Get the value of the future, to get an exception if any
00130     data_mover.get();
00131   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::data_mover.

Referenced by trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::~dma().

Here is the caller graph for this function:



12.21.4.4 join() [2/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
void trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::join ( )
[inline], [private]
```

Wait for the fiber to complete.

Definition at line 128 of file dma.hpp.

```
00128               {
00129     // Get the value of the future, to get an exception if any
00130     data_mover.get();
00131   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::data_mover.

12.21.4.5 launch() [1/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::launch (
             ::trisycl::detail::fiber_pool & fiber_executor,
             Invocable && f )  [inline], [private]
```

Launch the data mover on the execution pool.

Definition at line 135 of file dma.hpp.

```
00135 {}
```

12.21.4.6 launch() [2/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
template<typename Invocable >
void trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::launch (
             ::trisycl::detail::fiber_pool & fiber_executor,
             Invocable && f )  [inline], [private]
```

Launch the data mover on the execution pool.

Definition at line 135 of file dma.hpp.

```
00135 {}
```

12.21.4.7 pop_command() [1/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
auto trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::pop_command (
            auto & command )  [inline], [private]
```

Dequeue a DMA command.

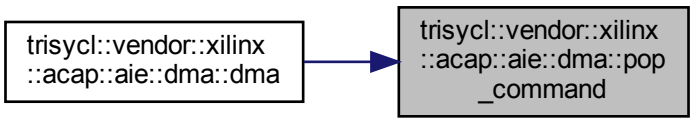
Definition at line 138 of file dma.hpp.

```
00138 { return c.pop(command); }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::c.

Referenced by trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::dma().

Here is the caller graph for this function:



12.21.4.8 pop_command() [2/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
auto trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::pop_command (
            auto & command )  [inline], [private]
```

Dequeue a DMA command.

Definition at line 138 of file dma.hpp.

```
00138 { return c.pop(command); }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::c.

12.21.4.9 push_command() [1/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
template<typename Cmd >
void trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::push_command
(
            Cmd && command )  [inline], [protected]
```

Enqueue a DMA command.

Definition at line 149 of file dma.hpp.

```
00149                                                 {
00150     ++nb_command;
00151     c.push(std::forward<Cmd>(command));
00152   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::c, and trisycl::vendor::xilinx::acap::aie::dma

12.21.4.10 push_command() [2/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
template<typename Cmd >
void trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::push_command
(
            Cmd && command )  [inline], [protected]
```

Enqueue a DMA command.

Definition at line 149 of file dma.hpp.

```
00149                                                    {
00150     ++nb_command;
00151     c.push(std::forward<Cmd>(command));
00152   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::c, and trisycl::vendor::xilinx::acap::aie::dma

12.21.4.11 wait() [1/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
SpecializedDMA & trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >↩
::wait ( )  [inline]
```

Wait for the transfers to complete.

The API assumes that the program is written in such a way that at some point the queue will be empty so there is no continuous DMA command enqueuing for example.

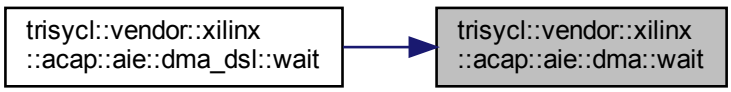
Definition at line 185 of file dma.hpp.

```
00185                          {
00186     if (nb_command != 0) {
00187       // We need to wait because there are still operations executing
00188       std::unique_lock lk { waiting_mutex };
00189       waiting_room.wait(lk, [&] { return nb_command == 0; });
00190     }
00191     /* To be able to chain another DMA operation on it, return the DMA
00192 object itself */
00193     return static_cast<SpecializedDMA&>(*this);
00194   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::nb_command, trisycl::vendor::xilinx::acap:: and trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::waiting_room.

Referenced by trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >::wait().

Here is the caller graph for this function:



12.21.4.12 wait() [2/2]

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
SpecializedDMA & trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >↩
::wait ( )  [inline]
```

Wait for the transfers to complete.

The API assumes that the program is written in such a way that at some point the queue will be empty so there is no continuous DMA command enqueuing for example.

Definition at line 185 of file dma.hpp.

```
00185                           {
00186     if (nb_command != 0) {
00187       // We need to wait because there are still operations executing
00188       std::unique_lock lk { waiting_mutex };
00189       waiting_room.wait(lk, [&] { return nb_command == 0; });
00190     }
00191     /* To be able to chain another DMA operation on it, return the DMA
00192 object itself */
00193     return static_cast<SpecializedDMA&>(*this);
00194   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::nb_command, trisycl::vendor::xilinx::acap:: and trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::waiting_room.

12.21.5 Member Data Documentation

12.21.5.1 c

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
boost::fibers::buffered_channel< dma_command > trisycl::vendor::xilinx::acap::aie::dma< AXIStream↩
Switch, SpecializedDMA >::c { 8 }  [private]
```

Queue for generic DMA commands.

Definition at line 113 of file dma.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::pop_command(), trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::push_command(), and trisycl::vendor::xilinx::acap::aie:

12.21.5.2 data_mover

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
trisycl::detail::fiber_pool::future< void > trisycl::vendor::xilinx::acap::aie::dma< AXIStream↩
Switch, SpecializedDMA >::data_mover  [private]
```

To shepherd the fiber playing the data mover behind the DMA.

Definition at line 116 of file dma.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::dma(), and trisycl::vendor::xilinx::acap::a

12.21.5.3 nb_command

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
std::atomic< int > trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA
>::nb_command { 0 }  [private]
```

Count the in-flight commands.

Definition at line 119 of file dma.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::commit_command(), trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::push_command(), and trisycl::vendor::xilinx::acap::aie:

12.21.5.4 waiting_mutex

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
boost::fibers::mutex trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA
>::waiting_mutex  [private]
```

To protect the waiting mechanism.

Definition at line 125 of file dma.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::wait().

12.21.5.5 waiting_room

```
template<typename AXIStreamSwitch , typename SpecializedDMA >
boost::fibers::condition_variable trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch,
SpecializedDMA >::waiting_room  [private]
```

To wait for the DMA command queue to drain.

Definition at line 122 of file dma.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::commit_command(), and trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::wait().

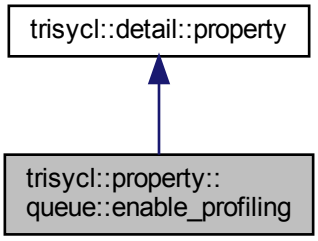
The documentation for this class was generated from the following files:

- include/sycl/vendor/Xilinx/acap/aie/dma.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/dma.hpp

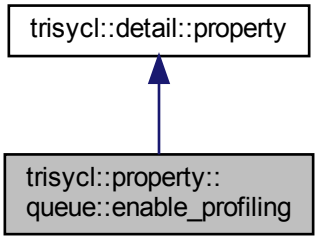
12.22 trisycl::property::queue::enable_profiling Class Reference

`#include <queue.hpp>`

Inheritance diagram for trisycl::property::queue::enable_profiling:



Collaboration diagram for trisycl::property::queue::enable_profiling:



Public Member Functions

- enable_profiling ()

12.22.1 Detailed Description

Definition at line 13 of file queue.hpp.

12.22.2 Constructor & Destructor Documentation

12.22.2.1 enable_profiling()

```
trisycl::property::queue::enable_profiling::enable_profiling ( )  [inline]
```

Definition at line 15 of file queue.hpp.

```
00015 {}
```

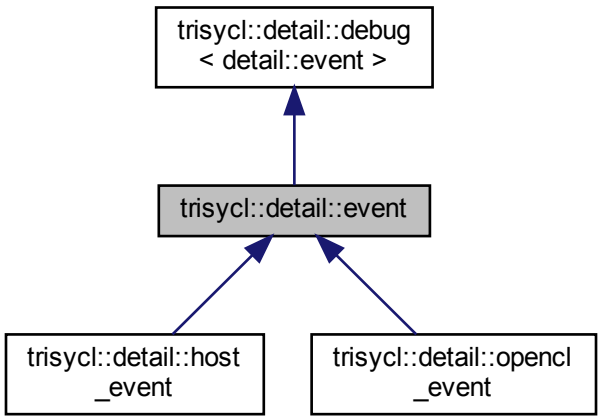
The documentation for this class was generated from the following file:

- include/triSYCL/property/queue.hpp

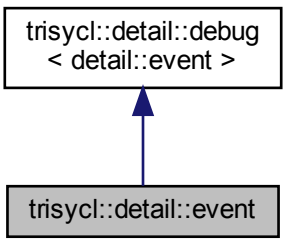
12.23 trisycl::detail::event Struct Reference

```
#include <event.hpp>
```

Inheritance diagram for trisycl::detail::event:



Collaboration diagram for trisycl::detail::event:



Public Member Functions

- virtual cl_event get () const =0

  Return the cl_event of the underlying OpenCL event.
- virtual const boost::compute::event & get_boost_compute () const =0

  Return the underlying Boost.Compute event, if any.
- virtual bool is_host () const =0
- virtual info::event_command_status get_command_execution_status () const =0
- virtual cl_ulong get_profiling_info (info::event_profiling param) const =0
- virtual cl_uint get_reference_count () const =0
- virtual void wait () const =0
- virtual ~event ()

Additional Inherited Members

12.23.1 Detailed Description

Definition at line 14 of file event.hpp.

12.23.2 Constructor & Destructor Documentation

12.23.2.1 ~event()

```
virtual trisycl::detail::event::~event ( )  [inline], [virtual]
```

Definition at line 35 of file event.hpp.

```
00035 {}
```

12.23.3 Member Function Documentation

12.23.3.1 get()

```
virtual cl_event trisycl::detail::event::get ( ) const  [pure virtual]
```

Return the cl_event of the underlying OpenCL event.

Implemented in trisycl::detail::host_event, and trisycl::detail::opencl_event.

12.23.3.2 get_boost_compute()

```
virtual const boost::compute::event & trisycl::detail::event::get_boost_compute ( ) const
[pure virtual]
```

Return the underlying Boost.Compute event, if any.

Implemented in trisycl::detail::host_event, and trisycl::detail::opencl_event.

12.23.3.3 get_command_execution_status()

```
virtual info::event_command_status trisycl::detail::event::get_command_execution_status ( )
const  [pure virtual]
```

Implemented in trisycl::detail::host_event, and trisycl::detail::opencl_event.

12.23.3.4 get_profiling_info()

```
virtual cl_ulong trisycl::detail::event::get_profiling_info (
            info::event_profiling param ) const  [pure virtual]
```

Implemented in trisycl::detail::host_event, and trisycl::detail::opencl_event.

12.23.3.5 get_reference_count()

```
virtual cl_uint trisycl::detail::event::get_reference_count ( ) const  [pure virtual]
```

Implemented in trisycl::detail::host_event, and trisycl::detail::opencl_event.

12.23.3.6 is_host()

```
virtual bool trisycl::detail::event::is_host ( ) const  [pure virtual]
```

Implemented in trisycl::detail::host_event, and trisycl::detail::opencl_event.

12.23.3.7 wait()

```
virtual void trisycl::detail::event::wait ( ) const  [pure virtual]
```

Implemented in trisycl::detail::host_event, and trisycl::detail::opencl_event.

The documentation for this struct was generated from the following file:

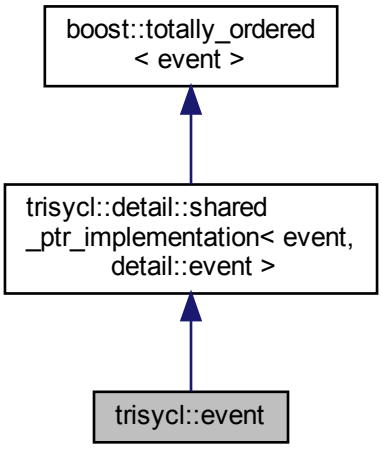
- include/triSYCL/event/detail/event.hpp

12.24 trisycl::event Class Reference

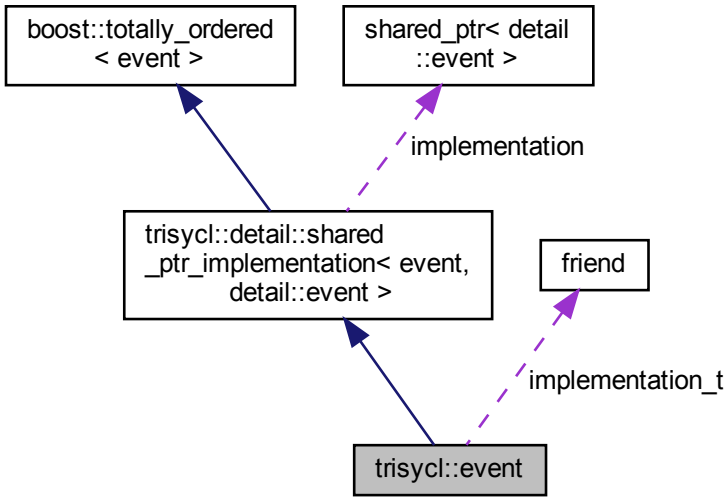
SYCL event.

```
#include <event.hpp>
```

Inheritance diagram for trisycl::event:



Collaboration diagram for trisycl::event:



Public Member Functions

- event ()
- event (cl_event clEvent, const context &syclContext)

  Construct an event class using the clEvent from OpenCL.
- event (const boost::compute::event &e)

  Construct an event class using the boost::compute::event.
- bool is_host () const

  Return true if the event is a SYCL host event.
- cl_event get ()

  Return the cl_event from the underlying OpenCL event.
- boost::compute::event get_boost_compute () const

  Return the underlying boost compute event.
- vector_class< event > get_wait_list ()
- void wait ()

  Wait for the event and the command associated with it to complete.
- void wait_and_throw ()
- template<info::event param>
  auto get_info () const

  Query the event for information.
- template<info::event_profiling param>
  auto get_profiling_info () const

  Query the event for event profiling.

Static Public Member Functions

- static void wait (const vector_class< event > &eventList)
- static void wait_and_throw (const vector_class< event > &eventList)

Private Types

- using implementation_t = typename event::shared_ptr_implementation

Private Attributes

- friend implementation_t

Additional Inherited Members

12.24.1 Detailed Description

SYCL event.

Definition at line 21 of file event.hpp.

12.24.2 Member Typedef Documentation

12.24.2.1 implementation_t

```
using trisycl::event::implementation_t = typename event::shared_ptr_implementation  [private]
```

Definition at line 24 of file event.hpp.

12.24.3 Constructor & Destructor Documentation

12.24.3.1 event() [1/3]

```
trisycl::event::event ( )  [inline]
```

Definition at line 29 of file event.hpp.

```
00029 :  implementation_t { detail::host_event::instance() } {}
```

12.24.3.2 event() [2/3]

```
trisycl::event::event (
              cl_event clEvent,
              const context & syclContext )  [inline]
```

Construct an event class using the clEvent from OpenCL.

This is a triSYCL extension for boost::compute interoperation.

Definition at line 36 of file event.hpp.

```
00037     :  event { boost::compute::event { clEvent } } {}
```

12.24.3.3 event() [3/3]

```
trisycl::event::event (
              const boost::compute::event & e )  [inline]
```

Construct an event class using the boost::compute::event.

This is a triSYCL extension for boost::compute interoperation.

Definition at line 43 of file event.hpp.

```
00044     :  implementation_t { detail::opencl_event::instance(e) } {}
```

12.24.4 Member Function Documentation

12.24.4.1 get()

```
cl_event trisycl::event::get ( )  [inline]
```

Return the cl_event from the underlying OpenCL event.

Definition at line 52 of file event.hpp.

```
00052                  {
00053     return implementation->get();
00054   }
```

References trisycl::detail::shared_ptr_implementation< event, detail::event >::implementation.

12.24.4.2 get_boost_compute()

```
boost::compute::event trisycl::event::get_boost_compute ( ) const  [inline]
```

Return the underlying boost compute event.

This is a triSYCL extension

Definition at line 60 of file event.hpp.

```
00060                                                {
00061     return implementation->get_boost_compute();
00062   }
```

References trisycl::detail::shared_ptr_implementation< event, detail::event >::implementation.

12.24.4.3 get_info()

```
template<info::event param>
auto trisycl::event::get_info ( ) const  [inline]
```

Query the event for information.

12.24.4.4 get_profiling_info()

```
template<info::event_profiling param>
auto trisycl::event::get_profiling_info ( ) const  [inline]
```

Query the event for event profiling.

12.24.4.5 get_wait_list()

```
vector_class< event > trisycl::event::get_wait_list ( )  [inline]
```

Definition at line 65 of file event.hpp.

```
00065                                        {
00066     TRISYCL_UNIMPL;
00067     return {};
00068   }
```

References TRISYCL_UNIMPL.

12.24.4.6 is_host()

```
bool trisycl::event::is_host ( ) const  [inline]
```

Return true if the event is a SYCL host event.

Definition at line 48 of file event.hpp.

```
00048 { return implementation->is_host(); }
```

References trisycl::detail::shared_ptr_implementation< event, detail::event >::implementation.

12.24.4.7 wait() [1/2]

```
void trisycl::event::wait ( )  [inline]
```

Wait for the event and the command associated with it to complete.

Definition at line 73 of file event.hpp.

```
00073               {
00074     implementation->wait();
00075   }
```

References trisycl::detail::shared_ptr_implementation< event, detail::event >::implementation.

12.24.4.8 wait() [2/2]

```
static void trisycl::event::wait (
              const vector_class< event > & eventList )  [inline], [static]
```

Definition at line 77 of file event.hpp.

```
00077                                                  {
00078     TRISYCL_UNIMPL;
00079   }
```

References TRISYCL_UNIMPL.

12.24.4.9 wait_and_throw() [1/2]

```
void trisycl::event::wait_and_throw ( )  [inline]
```

Definition at line 81 of file event.hpp.

```
00081                         {
00082     TRISYCL_UNIMPL;
00083   }
```

References TRISYCL_UNIMPL.

12.24.4.10 wait_and_throw() [2/2]

```
static void trisycl::event::wait_and_throw (
            const vector_class< event > & eventList )  [inline], [static]
```

Definition at line 85 of file event.hpp.

```
00085                                                          {
00086     TRISYCL_UNIMPL;
00087   }
```

References TRISYCL_UNIMPL.

12.24.5 Member Data Documentation

12.24.5.1 implementation_t

```
friend trisycl::event::implementation_t  [private]
```

Definition at line 26 of file event.hpp.

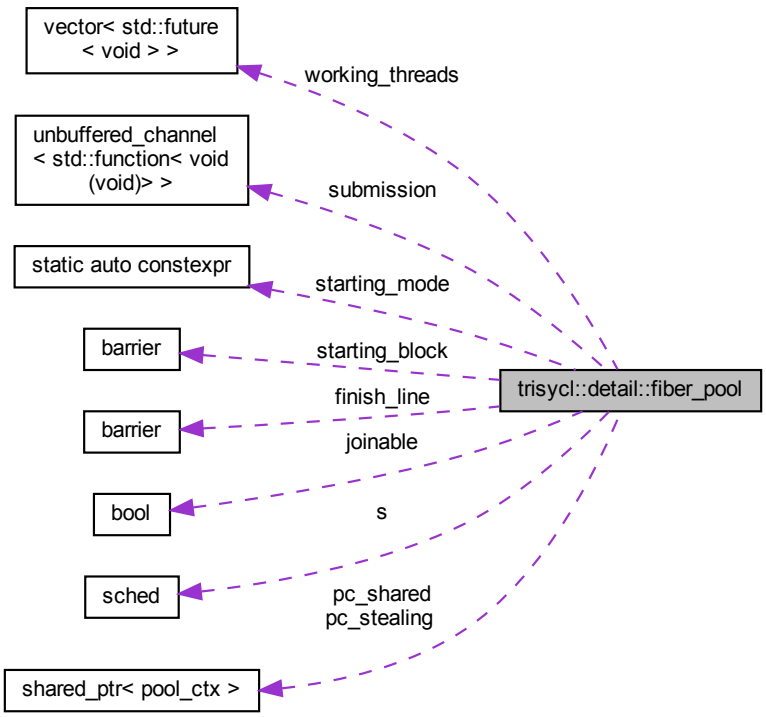
The documentation for this class was generated from the following file:

- include/triSYCL/event.hpp

12.25 trisycl::detail::fiber_pool Class Reference

```
#include <fiber_pool.hpp>
```

Collaboration diagram for trisycl::detail::fiber_pool:



Public Types

- enum class sched { round_robin , shared_work , work_stealing }

  To select some various Boost.Fibers schedulers.
- template<typename T >
  using future = boost::fibers::future< T >

  The type of the future used by the implementation underneath.

Public Member Functions

- fiber_pool (int thread_number, sched scheduler, bool suspend)

  Create a fiber_pool.
- template<typename Callable >
  auto submit (Callable &&work)

  Submit some work on a new fiber.
- void close ()

  Close the submission.
- void join ()

  Wait for all the threads are done.
- ~fiber_pool ()

  Wait for some remaining work to be done.

Private Member Functions

- void run (int i)

  The thread worker job.

Private Attributes

- std::vector< std::future< void > > working_threads

  The thread running the Boost.Fiber schedulers to do the work.
- boost::fibers::unbuffered_channel< std::function< void(void)> > submission

  The queue to submit work.
- boost::barrier starting_block

  To synchronize all the threads before they can run some fibers.
- boost::fibers::barrier finish_line

  To synchronize all the threads after their last fiber.
- bool joinable = true

  To avoid joining several times.
- sched s

  The model of scheduler.
- pooled_work_stealing::ctx pc_stealing
- pooled_shared_work::ctx pc_shared

Static Private Attributes

- static auto constexpr starting_mode = boost::fibers::launch::dispatch

12.25.1 Detailed Description

Definition at line 38 of file fiber_pool.hpp.

12.25.2 Member Typedef Documentation

12.25.2.1 future

```
template<typename T >
using trisycl::detail::fiber_pool::future = boost::fibers::future<T>
```

The type of the future used by the implementation underneath.

Definition at line 83 of file fiber_pool.hpp.

12.25.3 Member Enumeration Documentation

12.25.3.1 sched

```
enum class trisycl::detail::fiber_pool::sched  [strong]
```

To select some various Boost.Fibers schedulers.

Enumerator

round_robin	
shared_work	
work_stealing	

Definition at line 43 of file fiber_pool.hpp.

```
00043                   {
00044     round_robin,
00045     shared_work,
00046     work_stealing
00047     // \todo Add numa
00048   };
```

12.25.4 Constructor & Destructor Documentation

12.25.4.1 fiber_pool()

```
trisycl::detail::fiber_pool::fiber_pool (
              int thread_number,
              sched scheduler,
              bool suspend )  [inline]
```

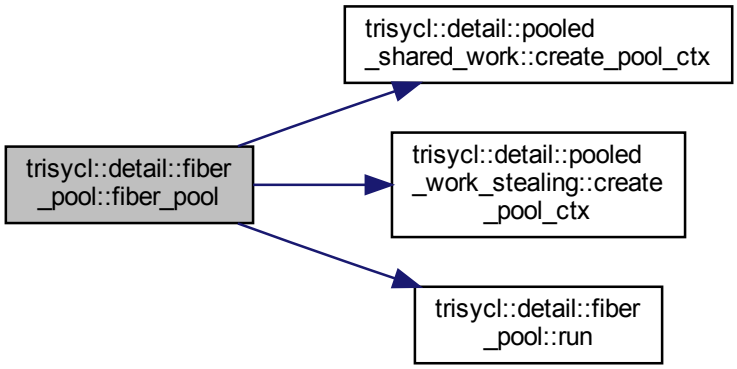
Create a fiber_pool.

Definition at line 86 of file fiber_pool.hpp.

```
00089     :  starting_block { static_cast<unsigned int>(thread_number) + 1 }
00090     , finish_line { static_cast<unsigned int>(thread_number) }
00091     , s { scheduler }
00092   {
00093     if (scheduler == sched::shared_work)
00094       // This scheduler needs a shared context
00095       pc_shared = pooled_shared_work::create_pool_ctx(suspend);
00096     else if (scheduler == sched::work_stealing)
00097       // This scheduler needs a shared context
00098       pc_stealing = pooled_work_stealing::create_pool_ctx
00099         (thread_number, suspend);
00100     // Start the working threads
00101     working_threads = ranges::iota_view { 0, thread_number }
00102                     | ranges::views::transform([&] (int i) {
00103                         return std::async(std::launch::async,
00104                                           [&, i] { run(i); }); })
00105                     | ranges::to<std::vector>;
00106     // Wait for all thread workers to be ready
00107     starting_block.count_down_and_wait();
00108   }
```

References trisycl::detail::pooled_shared_work::create_pool_ctx(), trisycl::detail::pooled_work_stealing::create_pool_ctx(), i, pc_shared, pc_stealing, run(), shared_work, starting_block, work_stealing, and working_threads.

Here is the call graph for this function:



12.25.4.2 ~fiber_pool()

```
trisycl::detail::fiber_pool::~fiber_pool ( )  [inline]
```

Wait for some remaining work to be done.

Definition at line 156 of file fiber_pool.hpp.

```
00156                 {
00157     // Join first if not done already
00158     join();
00159     // There is a circular dependecy of shared_ptr/intrusive_ptr between
00160     // pc_stealing and its schedulers_.  after we join we know none of the
00161     // schedulers_ are still running so we break the cycle and everything
00162     // cleansup itself.
00163     if (pc_stealing)
00164       pc_stealing->schedulers_.clear();
00165   }
```

References join(), and pc_stealing.

Here is the call graph for this function:



12.25.5 Member Function Documentation

12.25.5.1 close()

```
void trisycl::detail::fiber_pool::close ( )  [inline]
```

Close the submission.

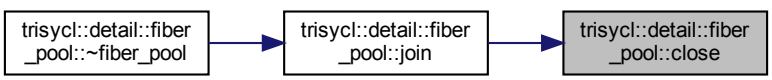
Definition at line 134 of file fiber_pool.hpp.

```
00134               {
00135     // Can be done many times, so no protection required here
00136     submission.close();
00137   }
```

References submission.

Referenced by join().

Here is the caller graph for this function:



12.25.5.2 join()

```
void trisycl::detail::fiber_pool::join ( )  [inline]
```

Wait for all the threads are done.

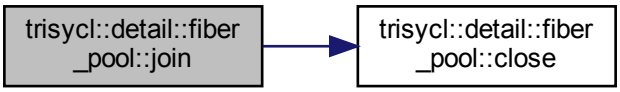
Definition at line 141 of file fiber_pool.hpp.

```
00141             {
00142     // Can be done only once
00143     if (joinable) {
00144       // Close the submission if not done already
00145       close();
00146       for (auto &t :  working_threads)
00147         // A Boost.Fiber scheduler will block its thread if they still
00148         // have some work to do
00149         t.get();
00150       joinable = false;
00151     }
00152   }
```

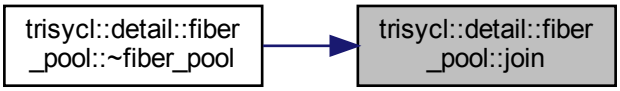
References close(), joinable, and working_threads.

Referenced by ~fiber_pool().

Here is the call graph for this function:



Here is the caller graph for this function:



12.25.5.3 run()

```
void trisycl::detail::fiber_pool::run (
            int i )  [inline], [private]
```

The thread worker job.

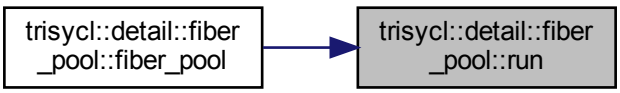
Definition at line 170 of file fiber_pool.hpp.

```
00170                 {
00171     if (s == sched::shared_work)
00172       boost::fibers::use_scheduling_algorithm<pooled_shared_work>(pc_shared);
00173     else if (s == sched::work_stealing)
00174       boost::fibers::use_scheduling_algorithm
00175         <pooled_work_stealing>(pc_stealing);
00176     // Otherwise a round-robin scheduler is used and the fibers will
00177     // use only 1 thread since there is no thread migration in that
00178     // case
00179
00180     // Wait for all thread workers to be ready
00181     starting_block.count_down_and_wait();
00182
00183     // Only the first thread receives and starts the work
00184     if (i == 0) {
00185       // Keep track of each fiber execution to avoid quitting before completion
00186       std::vector<boost::fibers::fiber> fibers;
00187       for (;;) {
00188         decltype(submission)::value_type work;
00189         if (submission.pop(work)
00190             == boost::fibers::channel_op_status::closed)
00191           // Someone asked to stop accepting work
00192           break;
00193         // Launch the work on a new fiber
00194         fibers.emplace_back(starting_mode, std::move(work));
00195       }
00196       // Now wait for the completion of each fiber
00197       for (auto &f :  fibers)
00198         f.join();
00199     }
00200     // Wait for all the threads to finish their fiber execution
00201     finish_line.wait();
00202   }
```

References finish_line, i, pc_shared, pc_stealing, s, shared_work, starting_block, starting_mode, submission, and work_stealing.

Referenced by fiber_pool().

Here is the caller graph for this function:



12.25.5.4 submit()

```
template<typename Callable >
auto trisycl::detail::fiber_pool::submit (
            Callable && work )  [inline]
```

Submit some work on a new fiber.

Parameters

in	*work*	is the callable to execute, taking no arguments and returning a result of some type R

Returns

a future<R>

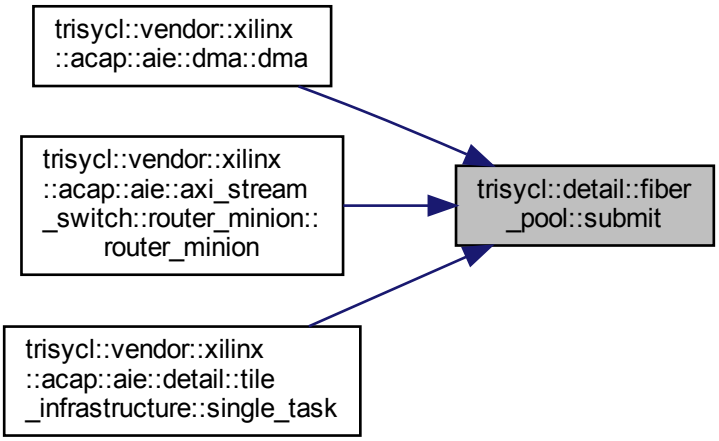
Definition at line 119 of file fiber_pool.hpp.

```
00119                                 {
00120     // Put the packaged_task into a shared_ptr to fit it later in a
00121     // lambda in a std::function since the type is move-only
00122     auto pt = std::make_shared<boost::fibers::packaged_task<void(void)»
00123       ([w = std::forward<Callable>(work)] () mutable { return w(); });
00124     auto f = pt->get_future();
00125     // Submit a lambda to do the type erasure so the submission queue
00126     // type is independent of the type of the packaged_task
00127     submission.push([p = std::move(pt)] () mutable { (*p)(); });
00128     // Return the future to the client to get the result or the exception
00129     return f;
00130   }
```

References p, and submission.

Referenced by trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >::dma(), trisycl::vendor::xilinx::acap::aie::a
and trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::single_task().

Here is the caller graph for this function:



12.25.6 Member Data Documentation

12.25.6.1 finish_line

```
boost::fibers::barrier trisycl::detail::fiber_pool::finish_line  [private]
```

To synchronize all the threads after their last fiber.

Definition at line 65 of file fiber_pool.hpp.

Referenced by run().

12.25.6.2 joinable

```
bool trisycl::detail::fiber_pool::joinable = true  [private]
```

To avoid joining several times.

Definition at line 68 of file fiber_pool.hpp.

Referenced by join().

12.25.6.3 pc_shared

```
pooled_shared_work::ctx trisycl::detail::fiber_pool::pc_shared  [private]
```

Definition at line 77 of file fiber_pool.hpp.

Referenced by fiber_pool(), and run().

12.25.6.4 pc_stealing

```
pooled_work_stealing::ctx trisycl::detail::fiber_pool::pc_stealing  [private]
```

Definition at line 74 of file fiber_pool.hpp.

Referenced by fiber_pool(), run(), and ~fiber_pool().

12.25.6.5 s

```
sched trisycl::detail::fiber_pool::s  [private]
```

The model of scheduler.

Definition at line 71 of file fiber_pool.hpp.

Referenced by run().

12.25.6.6 starting_block

```
boost::barrier trisycl::detail::fiber_pool::starting_block  [private]
```

To synchronize all the threads before they can run some fibers.

Definition at line 62 of file fiber_pool.hpp.

Referenced by fiber_pool(), and run().

12.25.6.7 starting_mode

```
auto constexpr trisycl::detail::fiber_pool::starting_mode = boost::fibers::launch::dispatch
[static], [constexpr], [private]
```

Definition at line 59 of file fiber_pool.hpp.

Referenced by run().

12.25.6.8 submission

```
boost::fibers::unbuffered_channel<std::function<void(void)> > trisycl::detail::fiber_pool↩
::submission  [private]
```

The queue to submit work.

Definition at line 56 of file fiber_pool.hpp.

Referenced by close(), run(), and submit().

12.25.6.9 working_threads

```
std::vector<std::future<void> > trisycl::detail::fiber_pool::working_threads  [private]
```

The thread running the Boost.Fiber schedulers to do the work.

Definition at line 53 of file fiber_pool.hpp.

Referenced by fiber_pool(), and join().

The documentation for this class was generated from the following file:

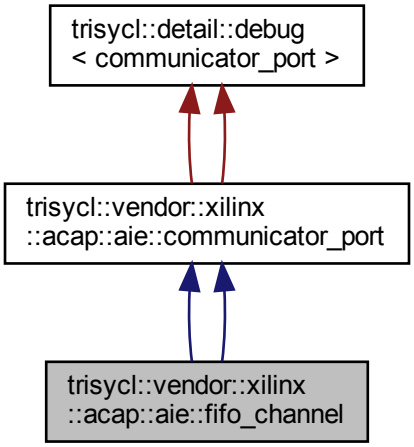
- include/triSYCL/detail/fiber_pool.hpp

12.26 trisycl::vendor::xilinx::acap::aie::fifo_channel Class Reference

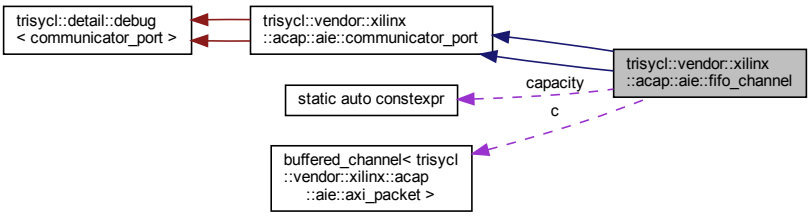
A FIFO channel connection.

`#include <connection.hpp>`

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::fifo_channel:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::fifo_channel:



Public Member Functions

- void write (const axi_packet &v) override

  Enqueue a packet to the FIFO channel.
- bool try_write (const axi_packet &v) override

  Try to enqueue a packet to the FIFO channel.
- value_type read () override

  Waiting read from the FIFO channel.
- bool try_read (value_type &v) override

Non-blocking read from the FIFO channel.

- void write (const axi_packet &v) override

  Enqueue a packet to the FIFO channel.
- bool try_write (const axi_packet &v) override

  Try to enqueue a packet to the FIFO channel.
- value_type read () override

  Waiting read from the FIFO channel.
- bool try_read (value_type &v) override

  Non-blocking read from the FIFO channel.

Private Types

- using value_type = axi_packet::value_type

  Payload data type.
- using value_type = axi_packet::value_type

  Payload data type.

Private Attributes

- boost::fibers::buffered_channel< axi_packet > c { capacity }

  The underlying FIFO.

Static Private Attributes

- static auto constexpr capacity = 16

  FIFO capacity queue.

12.26.1 Detailed Description

A FIFO channel connection.

Todo factorize out router minion

Todo factorize out router minion

Definition at line 115 of file connection.hpp.

12.26.2 Member Typedef Documentation

12.26.2.1 value_type [1/2]

```
using trisycl::vendor::xilinx::acap::aie::fifo_channel::value_type = axi_packet::value_type
[private]
```

Payload data type.

Definition at line 121 of file connection.hpp.

12.26.2.2 value_type [2/2]

```
using trisycl::vendor::xilinx::acap::aie::fifo_channel::value_type = axi_packet::value_type
[private]
```

Payload data type.

Definition at line 121 of file connection.hpp.

12.26.3 Member Function Documentation

12.26.3.1 read() [1/2]

```
value_type trisycl::vendor::xilinx::acap::aie::fifo_channel::read ( )  [inline], [override],
[virtual]
```

Waiting read from the FIFO channel.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 144 of file connection.hpp.

```
00144                                {
00145     return c.value_pop().data;
00146   }
```

References c.

12.26.3.2 read() [2/2]

```
value_type trisycl::vendor::xilinx::acap::aie::fifo_channel::read ( )  [inline], [override],
[virtual]
```

Waiting read from the FIFO channel.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 144 of file connection.hpp.

```
00144                                {
00145     return c.value_pop().data;
00146   }
```

References c.

12.26.3.3 try_read() [1/2]

```
bool trisycl::vendor::xilinx::acap::aie::fifo_channel::try_read (
            value_type & v )  [inline], [override], [virtual]
```

Non-blocking read from the FIFO channel.

Returns

true if the value was correctly read

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 153 of file connection.hpp.

```
00153                                       {
00154     axi_packet p;
00155     if (c.try_pop(p) == boost::fibers::channel_op_status::success) {
00156       v = p.data;
00157       return true;
00158     }
00159     return false;
00160   }
```

References c, and p.

12.26.3.4 try_read() [2/2]

```
bool trisycl::vendor::xilinx::acap::aie::fifo_channel::try_read (
            value_type & v )  [inline], [override], [virtual]
```

Non-blocking read from the FIFO channel.

Returns

true if the value was correctly read

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 153 of file connection.hpp.

```
00153                                       {
00154     axi_packet p;
00155     if (c.try_pop(p) == boost::fibers::channel_op_status::success) {
00156       v = p.data;
00157       return true;
00158     }
00159     return false;
00160   }
```

References c, and p.

12.26.3.5 try_write() [1/2]

```
bool trisycl::vendor::xilinx::acap::aie::fifo_channel::try_write (
            const axi_packet & v )  [inline], [override], [virtual]
```

Try to enqueue a packet to the FIFO channel.

Returns

true if the packet is correctly enqueued

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 138 of file connection.hpp.

```
00138                                                {
00139     return c.try_push(v) == boost::fibers::channel_op_status::success;
00140   }
```

References c.

12.26.3.6 try_write() [2/2]

```
bool trisycl::vendor::xilinx::acap::aie::fifo_channel::try_write (
            const axi_packet & v )  [inline], [override], [virtual]
```

Try to enqueue a packet to the FIFO channel.

Returns

true if the packet is correctly enqueued

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 138 of file connection.hpp.

```
00138                                                {
00139     return c.try_push(v) == boost::fibers::channel_op_status::success;
00140   }
```

References c.

12.26.3.7 write() [1/2]

```
void trisycl::vendor::xilinx::acap::aie::fifo_channel::write (
            const axi_packet & v )  [inline], [override], [virtual]
```

Enqueue a packet to the FIFO channel.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 129 of file connection.hpp.

```
00129                                               {
00130     c.push(v);
00131   }
```

References c.

12.26.3.8 write() [2/2]

```
void trisycl::vendor::xilinx::acap::aie::fifo_channel::write (
            const axi_packet & v )  [inline], [override], [virtual]
```

Enqueue a packet to the FIFO channel.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 129 of file connection.hpp.

```
00129                                       {
00130     c.push(v);
00131   }
```

References c.

12.26.4 Member Data Documentation

12.26.4.1 c

```
boost::fibers::buffered_channel< axi_packet > trisycl::vendor::xilinx::acap::aie::fifo_↩
channel::c { capacity }  [private]
```

The underlying FIFO.

Definition at line 124 of file connection.hpp.

Referenced by read(), try_read(), try_write(), and write().

12.26.4.2 capacity

```
static auto constexpr trisycl::vendor::xilinx::acap::aie::fifo_channel::capacity = 16  [static],
[constexpr], [private]
```

FIFO capacity queue.

Todo check with hardware team for the value

Todo check with hardware team for the value

Definition at line 118 of file connection.hpp.

The documentation for this class was generated from the following files:

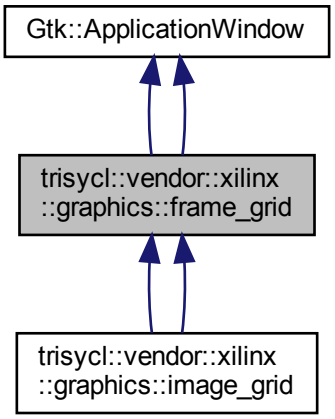
- include/sycl/vendor/Xilinx/acap/aie/connection.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/connection.hpp

12.27 trisycl::vendor::xilinx::graphics::frame_grid Struct Reference

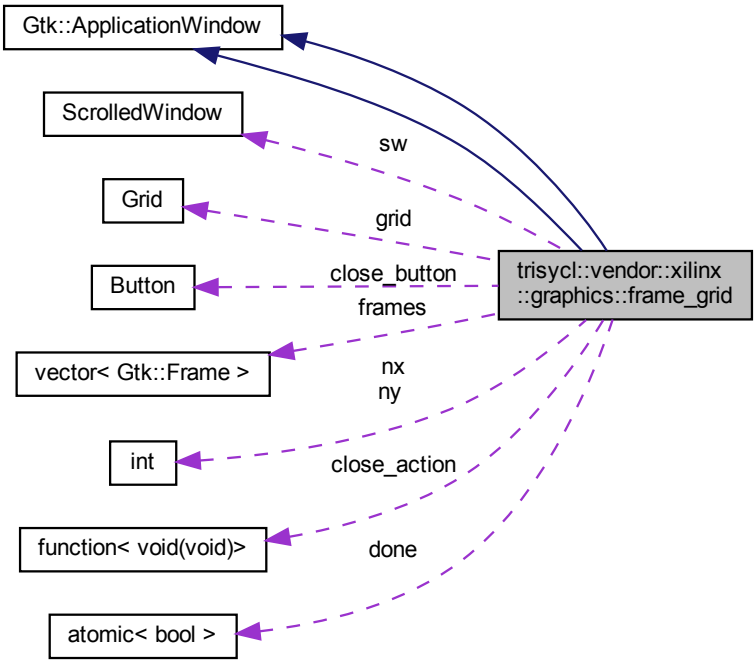
An application window displaying a grid of tiles.

`#include <graphics.hpp>`

Inheritance diagram for trisycl::vendor::xilinx::graphics::frame_grid:



Collaboration diagram for trisycl::vendor::xilinx::graphics::frame_grid:



Public Member Functions

- frame_grid (int nx, int ny)
  Create a grid of tiles.
- void set_close_action (std::function< void(void)> f)
  Set a function to be called on close.
- auto & get_frame (int x, int y)
  Get the frame at a given grid position.
- frame_grid (int nx, int ny)
  Create a grid of tiles.
- void set_close_action (std::function< void(void)> f)
  Set a function to be called on close.
- auto & get_frame (int x, int y)
  Get the frame at a given grid position.

Public Attributes

- Gtk::ScrolledWindow sw
  Nice to have some scroll bars around when the main window is too small.
- Gtk::Grid grid
  The container to represent the grid of tile images.
- Gtk::Button close_button { "Close" }
  A close button in case the closing function is not provided by the window manager.
- std::vector< Gtk::Frame > frames
  The linearized 2D vector of frames used to decorate the tile images with some frames with the tile names.
- int nx
  Number of frame columns.
- int ny
  Number of frame lines.
- std::function< void(void)> close_action
  An action to do when the window is closed.
- std::atomic< bool > done = false
  Set to true by the closing handler.

12.27.1 Detailed Description

An application window displaying a grid of tiles.

Each tile is framed with the tile identifiers

Definition at line 56 of file graphics.hpp.

12.27.2 Constructor & Destructor Documentation

12.27.2.1 frame_grid() [1/2]

```
trisycl::vendor::xilinx::graphics::frame_grid::frame_grid (
            int nx,
            int ny )  [inline]
```

Create a grid of tiles.

Parameters

in	*nx*	is the numer of tiles horizontally
in	*ny*	is the numer of tiles vertically

Definition at line 90 of file graphics.hpp.

```
00090                                 :  nx { nx }, ny { ny } {
00091     set_default_size(1900, 1020);
00092
00093     add(sw);
00094     sw.add(grid);
00095     for (int y = 0; y < ny; ++y)
00096       for (int x = 0; x < nx; ++x) {
00097         std::ostringstream s;
00098         s « "Tile(" « x « ',' « y « ')';
00099         frames.emplace_back(s.str());
00100         frames.back().set_shadow_type(Gtk::SHADOW_ETCHED_OUT);
00101         // A minimal border to save space on main window
00102         frames.back().set_border_width(1);
00103         // Display the frame with the lower y South in a mathematical sense
00104         grid.attach(frames.back(), x, ny - y - 1, 1, 1);
00105       }
00106
00107     grid.add(close_button);
00108
00109     // Make the button the default widget
00110     close_button.set_can_default();
00111     close_button.grab_default();
00112
00113     // Connect the clicked signal of the close button to
00114     close_button.signal_clicked().connect([this] {
00115         // Unmap the window from the screen
00116         hide();
00117         // Call the handler if it does exist
00118         if (close_action)
00119           close_action();
00120       });
00121
00122     // Show all children of the window
00123     show_all_children();
00124   }
```

References close_action, close_button, frames, grid, nx, ny, sw, x, and trisycl::y.

12.27.2.2 frame_grid() [2/2]

```
trisycl::vendor::xilinx::graphics::frame_grid::frame_grid (
            int nx,
            int ny )  [inline]
```

Create a grid of tiles.

Parameters

in	*nx*	is the numer of tiles horizontally
in	*ny*	is the numer of tiles vertically

Definition at line 90 of file graphics.hpp.

```
00090                                 :  nx { nx }, ny { ny } {
00091     set_default_size(1900, 1020);
00092
00093     add(sw);
00094     sw.add(grid);
00095     for (int y = 0; y < ny; ++y)
00096       for (int x = 0; x < nx; ++x) {
```

```
00097         std::ostringstream s;
00098         s « "Tile(" « x « ',' « y « ')';
00099         frames.emplace_back(s.str());
00100         frames.back().set_shadow_type(Gtk::SHADOW_ETCHED_OUT);
00101         // A minimal border to save space on main window
00102         frames.back().set_border_width(1);
00103         // Display the frame with the lower y South in a mathematical sense
00104         grid.attach(frames.back(), x, ny - y - 1, 1, 1);
00105       }
00106
00107     grid.add(close_button);
00108
00109     // Make the button the default widget
00110     close_button.set_can_default();
00111     close_button.grab_default();
00112
00113     // Connect the clicked signal of the close button to
00114     close_button.signal_clicked().connect([this] {
00115         // Unmap the window from the screen
00116         hide();
00117         // Call the handler if it does exist
00118         if (close_action)
00119           close_action();
00120       });
00121
00122     // Show all children of the window
00123     show_all_children();
00124   }
```

References close_action, close_button, frames, grid, nx, ny, sw, x, and trisycl::y.

12.27.3 Member Function Documentation

12.27.3.1 get_frame() [1/2]

```
auto & trisycl::vendor::xilinx::graphics::frame_grid::get_frame (
            int x,
            int y )  [inline]
```

Get the frame at a given grid position.

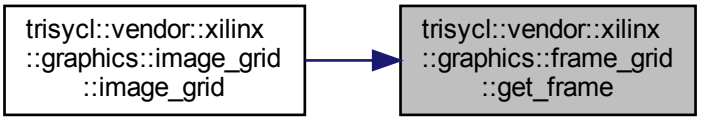
Definition at line 134 of file graphics.hpp.

```
00134                                        {
00135     return frames.at(x + nx*y);
00136   }
```

References frames, nx, x, and trisycl::y.

Referenced by trisycl::vendor::xilinx::graphics::image_grid::image_grid().

Here is the caller graph for this function:



12.27.3.2 get_frame() [2/2]

```
auto & trisycl::vendor::xilinx::graphics::frame_grid::get_frame (
            int x,
            int y )  [inline]
```

Get the frame at a given grid position.

Definition at line 134 of file graphics.hpp.

```
00134                                       {
00135     return frames.at(x + nx*y);
00136   }
```

References frames, nx, x, and trisycl::y.

12.27.3.3 set_close_action() [1/2]

```
void trisycl::vendor::xilinx::graphics::frame_grid::set_close_action (
            std::function< void(void)> f )  [inline]
```

Set a function to be called on close.

Definition at line 128 of file graphics.hpp.

```
00128                                                 {
00129     close_action = f;
00130   }
```

References close_action.

12.27.3.4 set_close_action() [2/2]

```
void trisycl::vendor::xilinx::graphics::frame_grid::set_close_action (
            std::function< void(void)> f )  [inline]
```

Set a function to be called on close.

Definition at line 128 of file graphics.hpp.

```
00128                                                 {
00129     close_action = f;
00130   }
```

References close_action.

12.27.4 Member Data Documentation

12.27.4.1 close_action

```
std::function< void(void)> trisycl::vendor::xilinx::graphics::frame_grid::close_action
```

An action to do when the window is closed.

Definition at line 78 of file graphics.hpp.

Referenced by frame_grid(), and set_close_action().

12.27.4.2 close_button

```
Gtk::Button trisycl::vendor::xilinx::graphics::frame_grid::close_button { "Close" }
```

A close button in case the closing function is not provided by the window manager.

Definition at line 65 of file graphics.hpp.

Referenced by frame_grid().

12.27.4.3 done

```
std::atomic< bool > trisycl::vendor::xilinx::graphics::frame_grid::done = false
```

Set to true by the closing handler.

Definition at line 81 of file graphics.hpp.

Referenced by trisycl::vendor::xilinx::graphics::image_grid::image_grid(), trisycl::vendor::xilinx::graphics::image_grid::is_done_barrier(
and trisycl::vendor::xilinx::graphics::image_grid::submit().

12.27.4.4 frames

```
std::vector< Gtk::Frame > trisycl::vendor::xilinx::graphics::frame_grid::frames
```

The linearized 2D vector of frames used to decorate the tile images with some frames with the tile names.

Definition at line 69 of file graphics.hpp.

Referenced by frame_grid(), and get_frame().

12.27.4.5 grid

```
Gtk::Grid trisycl::vendor::xilinx::graphics::frame_grid::grid
```

The container to represent the grid of tile images.

Definition at line 61 of file graphics.hpp.

Referenced by frame_grid().

12.27.4.6 nx

```
int trisycl::vendor::xilinx::graphics::frame_grid::nx
```

Number of frame columns.

Definition at line 72 of file graphics.hpp.

Referenced by frame_grid(), get_frame(), trisycl::vendor::xilinx::graphics::image_grid::image_grid(), and trisycl::vendor::xilinx::graphics::image_grid::update_tile_data_image().

12.27.4.7 ny

```
int trisycl::vendor::xilinx::graphics::frame_grid::ny
```

Number of frame lines.

Definition at line 75 of file graphics.hpp.

Referenced by frame_grid(), and trisycl::vendor::xilinx::graphics::image_grid::image_grid().

12.27.4.8 sw

```
Gtk::ScrolledWindow trisycl::vendor::xilinx::graphics::frame_grid::sw
```

Nice to have some scroll bars around when the main window is too small.

Definition at line 58 of file graphics.hpp.

Referenced by frame_grid().

The documentation for this struct was generated from the following files:

- include/sycl/vendor/Xilinx/graphics.hpp
- include/triSYCL/vendor/Xilinx/graphics.hpp

12.28 trisycl::half Class Reference

`#include <half.hpp>`

Public Member Functions

- half (int)
- bool operator> (const half &h1)

12.28.1 Detailed Description

Definition at line 12 of file half.hpp.

12.28.2 Constructor & Destructor Documentation

12.28.2.1 half()

```
trisycl::half::half (
            int  )  [inline]
```

Definition at line 14 of file half.hpp.

```
00014 {};
```

12.28.3 Member Function Documentation

12.28.3.1 operator>()

```
bool trisycl::half::operator> (
            const half & h1 )  [inline]
```

Definition at line 16 of file half.hpp.

```
00016 { return false; };
```

The documentation for this class was generated from the following file:

- include/triSYCL/half.hpp

12.29 std::hash< trisycl::buffer< T, Dimensions, Allocator > > Struct Template Reference

`#include <buffer.hpp>`

Public Member Functions

- auto operator() (const trisycl::buffer< T, Dimensions, Allocator > &b) const

12.29.1 Detailed Description

template<typename T, int Dimensions, typename Allocator>
struct std::hash< trisycl::buffer< T, Dimensions, Allocator > >

Definition at line 579 of file buffer.hpp.

12.29.2 Member Function Documentation

12.29.2.1 operator()()

```
template<typename T , int Dimensions, typename Allocator >
auto std::hash< trisycl::buffer< T, Dimensions, Allocator > >::operator() (
             const trisycl::buffer< T, Dimensions, Allocator > & b ) const  [inline]
```

Definition at line 581 of file buffer.hpp.

```
00581                                                                    {
00582     // Forward the hashing to the implementation
00583     return b.hash();
00584   }
```

References b.

The documentation for this struct was generated from the following file:

- include/triSYCL/buffer.hpp

12.30 std::hash< trisycl::context > Struct Reference

```
#include <context.hpp>
```

Public Member Functions

- auto operator() (const trisycl::context &c) const

12.30.1 Detailed Description

Definition at line 234 of file context.hpp.

12.30.2 Member Function Documentation

12.30.2.1 operator()()

```
auto std::hash< trisycl::context >::operator() (
            const trisycl::context & c ) const  [inline]
```

Definition at line 235 of file context.hpp.

```
00235                                                {
00236     return c.hash();
00237     }
```

References c.

The documentation for this struct was generated from the following file:

- include/triSYCL/context.hpp

12.31 std::hash< trisycl::device > Struct Reference

```
#include <device.hpp>
```

Public Member Functions

- auto operator() (const trisycl::device &d) const

12.31.1 Detailed Description

Definition at line 266 of file device.hpp.

12.31.2 Member Function Documentation

12.31.2.1 operator()()

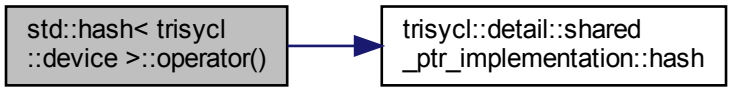
```
auto std::hash< trisycl::device >::operator() (
            const trisycl::device & d ) const  [inline]
```

Definition at line 268 of file device.hpp.

```
00268                                                {
00269     // Forward the hashing to the implementation
00270     return d.hash();
00271   }
```

References trisycl::detail::shared_ptr_implementation< Parent, Implementation >::hash().

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

- include/triSYCL/device.hpp

12.32 std::hash< trisycl::event > Struct Reference

`#include <event.hpp>`

Public Member Functions

- auto operator() (const trisycl::event &e) const

12.32.1 Detailed Description

Definition at line 129 of file event.hpp.

12.32.2 Member Function Documentation

12.32.2.1 operator()()

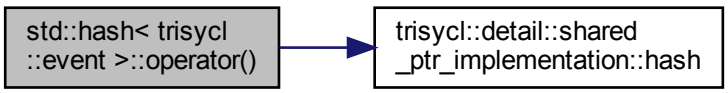
```
auto std::hash< trisycl::event >::operator() (
            const trisycl::event & e ) const  [inline]
```

Definition at line 131 of file event.hpp.

```
00131                                                {
00132     // Forward the hashing to the implementation
00133     return e.hash();
00134   }
```

References trisycl::detail::shared_ptr_implementation< Parent, Implementation >::hash().

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

- include/triSYCL/event.hpp

12.33 std::hash< trisycl::kernel > Struct Reference

`#include <kernel.hpp>`

Public Member Functions

- auto operator() (const trisycl::kernel &k) const

12.33.1 Detailed Description

Definition at line 123 of file kernel.hpp.

12.33.2 Member Function Documentation

12.33.2.1 operator()()

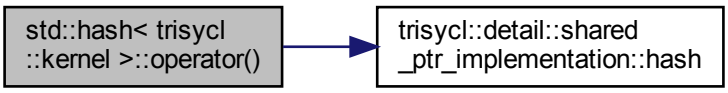
```
auto std::hash< trisycl::kernel >::operator() (
            const trisycl::kernel & k ) const  [inline]
```

Definition at line 125 of file kernel.hpp.

```
00125                                                {
00126     // Forward the hashing to the implementation
00127     return k.hash();
00128   }
```

References trisycl::detail::shared_ptr_implementation< Parent, Implementation >::hash().

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

- include/triSYCL/kernel.hpp

12.34 std::hash< trisycl::platform > Struct Reference

```
#include <platform.hpp>
```

Public Member Functions

- auto operator() (const trisycl::platform &p) const

12.34.1 Detailed Description

Definition at line 189 of file platform.hpp.

12.34.2 Member Function Documentation

12.34.2.1 operator()()

```
auto std::hash< trisycl::platform >::operator() (
            const trisycl::platform & p ) const  [inline]
```

Definition at line 191 of file platform.hpp.

```
00191                                                {
00192     // Forward the hashing to the implementation
00193     return p.hash();
00194   }
```

References p.

The documentation for this struct was generated from the following file:

- include/triSYCL/platform.hpp

12.35 std::hash< trisycl::queue > Struct Reference

```
#include <queue.hpp>
```

Public Member Functions

- auto operator() (const trisycl::queue &q) const

12.35.1 Detailed Description

Definition at line 385 of file queue.hpp.

12.35.2 Member Function Documentation

12.35.2.1 operator()()

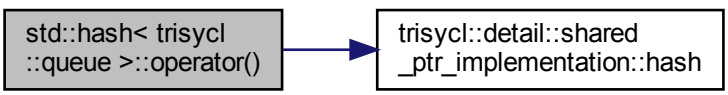
```
auto std::hash< trisycl::queue >::operator() (
            const trisycl::queue & q ) const  [inline]
```

Definition at line 387 of file queue.hpp.

```
00387                                               {
00388     // Forward the hashing to the implementation
00389     return q.hash();
00390   }
```

References trisycl::detail::shared_ptr_implementation< Parent, Implementation >::hash().

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

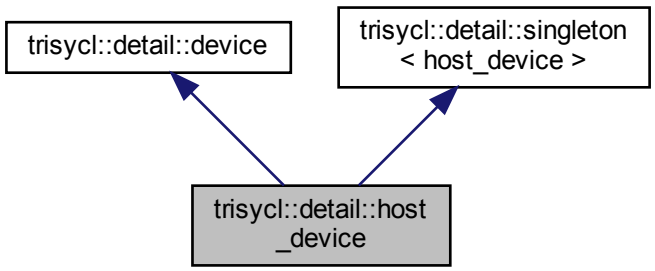
- include/triSYCL/queue.hpp

12.36 trisycl::detail::host_device Class Reference

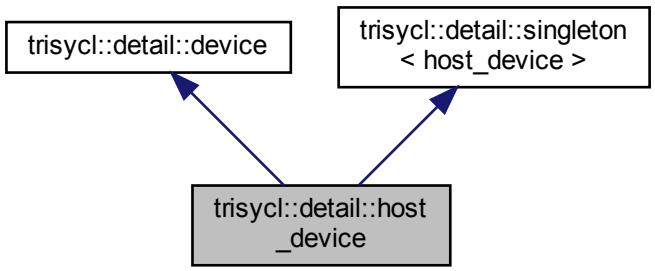
SYCL host device.

`#include <host_device.hpp>`

Inheritance diagram for trisycl::detail::host_device:



Collaboration diagram for trisycl::detail::host_device:



Public Member Functions

- cl_device_id get () const override

  Return the cl_device_id of the underlying OpenCL platform.
- boost::compute::device & get_boost_compute () override

  Return the underlying Boost.Compute device.
- bool is_host () const override

  Return true since the device is a SYCL host device.
- bool is_cpu () const override

  Return false since the host device is not an OpenCL CPU device.

- bool is_gpu () const override

  Return false since the host device is not an OpenCL GPU device.
- bool is_accelerator () const override

  Return false since the host device is not an OpenCL accelerator device.
- trisycl::platform get_platform () const override

  Return the platform of device.
- std::any get_info (info::device param) const override

  Query the device for OpenCL info::device info.
- bool has_extension (const string_class &extension) const override

  Specify whether a specific extension is supported on the device.

Additional Inherited Members

12.36.1 Detailed Description

SYCL host device.

Todo The implementation is quite minimal for now. :-)

Definition at line 29 of file host_device.hpp.

12.36.2 Member Function Documentation

12.36.2.1 get()

```
cl_device_id trisycl::detail::host_device::get ( ) const  [inline], [override], [virtual]
```

Return the cl_device_id of the underlying OpenCL platform.

This throws an error since there is no OpenCL device associated to the host device.

Implements trisycl::detail::device.

Definition at line 40 of file host_device.hpp.

```
00040                                 {
00041     throw non_cl_error("The host device has no OpenCL device");
00042   }
```

12.36.2.2 get_boost_compute()

```
boost::compute::device & trisycl::detail::host_device::get_boost_compute ( )  [inline], [override],
[virtual]
```

Return the underlying Boost.Compute device.

This throws an error since there is no OpenCL device associated to the host device.

Implements trisycl::detail::device.

Definition at line 50 of file host_device.hpp.

```
00050                                                {
00051     throw non_cl_error("The host device has no underlying OpenCL device");
00052   }
```

12.36.2.3 get_info()

```
std::any trisycl::detail::host_device::get_info (
            info::device param ) const  [inline], [override], [virtual]
```

Query the device for OpenCL info::device info.

Implements trisycl::detail::device.

Definition at line 100 of file host_device.hpp.

```
00100                                            {
00101     switch (param) {
00102     TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(max_work_group_size, static_cast<std::size_t>(8))
00103     TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(max_work_item_sizes, (trisycl::id<3>{ 128, 128, 128 }))
00104     TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(max_compute_units, static_cast<trisycl::cl_uint>(8))
00105     TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(device_type, info::device_type::host)
00106     TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(local_mem_type, info::local_mem_type::global)
00107     TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(local_mem_size, static_cast<trisycl::cl_ulong>(32768))
00108     default:
00109       return 0;
00110     }
00111   }
```

References trisycl::info::global, trisycl::info::host, and TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE.

12.36.2.4 get_platform()

```
trisycl::platform trisycl::detail::host_device::get_platform ( ) const  [inline], [override],
[virtual]
```

Return the platform of device.

Return synchronous errors via the SYCL exception class.

Todo To be implemented

Implements trisycl::detail::device.

Definition at line 86 of file host_device.hpp.

```
00086                                            {
00087     TRISYCL_UNIMPL;
00088     return {};
00089   }
```

References TRISYCL_UNIMPL.

12.36.2.5 has_extension()

```
bool trisycl::detail::host_device::has_extension (
            const string_class & extension ) const  [inline], [override], [virtual]
```

Specify whether a specific extension is supported on the device.

Todo To be implemented

Implements trisycl::detail::device.

Definition at line 117 of file host_device.hpp.

```
00117                                                                    {
00118     TRISYCL_UNIMPL;
00119     return {};
00120   }
```

References TRISYCL_UNIMPL.

12.36.2.6 is_accelerator()

```
bool trisycl::detail::host_device::is_accelerator ( ) const  [inline], [override], [virtual]
```

Return false since the host device is not an OpenCL accelerator device.

Implements trisycl::detail::device.

Definition at line 75 of file host_device.hpp.

```
00075                                    {
00076     return false;
00077   }
```

12.36.2.7 is_cpu()

```
bool trisycl::detail::host_device::is_cpu ( ) const  [inline], [override], [virtual]
```

Return false since the host device is not an OpenCL CPU device.

Implements trisycl::detail::device.

Definition at line 63 of file host_device.hpp.

```
00063                                {
00064     return false;
00065   }
```

12.36.2.8 is_gpu()

```
bool trisycl::detail::host_device::is_gpu ( ) const  [inline], [override], [virtual]
```

Return false since the host device is not an OpenCL GPU device.

Implements trisycl::detail::device.

Definition at line 69 of file host_device.hpp.

```
00069                                {
00070     return false;
00071   }
```

12.36.2.9 is_host()

```
bool trisycl::detail::host_device::is_host ( ) const  [inline], [override], [virtual]
```

Return true since the device is a SYCL host device.

Implements trisycl::detail::device.

Definition at line 57 of file host_device.hpp.

```
00057                                {
00058     return true;
00059   }
```

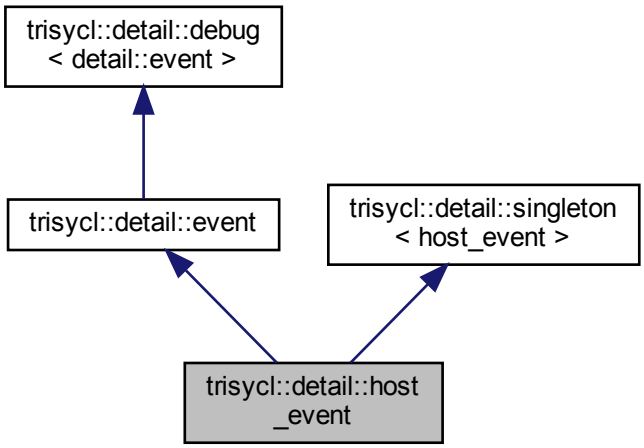
The documentation for this class was generated from the following file:

- include/triSYCL/device/detail/host_device.hpp

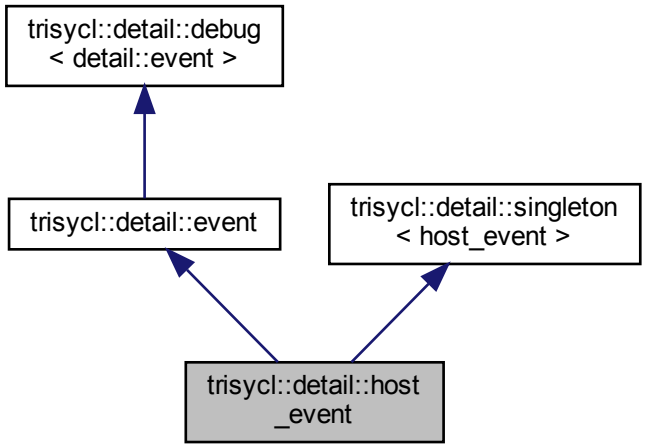
12.37 trisycl::detail::host_event Class Reference

```
#include <host_event.hpp>
```

Inheritance diagram for trisycl::detail::host_event:



Collaboration diagram for trisycl::detail::host_event:



Public Member Functions

- cl_event get () const override

  Return the cl_event of the underlying OpenCL event.
- boost::compute::event & get_boost_compute () const override

  Return the underlying Boost.Compute event, if any.
- bool is_host () const override
- cl_uint get_reference_count () const override
- info::event_command_status get_command_execution_status () const override
- cl_ulong get_profiling_info (info::event_profiling param) const override
- void wait () const override

Additional Inherited Members

12.37.1 Detailed Description

Definition at line 16 of file host_event.hpp.

12.37.2 Member Function Documentation

12.37.2.1 get()

```
cl_event trisycl::detail::host_event::get ( ) const  [inline], [override], [virtual]
```

Return the cl_event of the underlying OpenCL event.

Implements trisycl::detail::event.

Definition at line 21 of file host_event.hpp.

```
00021                                {
00022     throw non_cl_error("The host event has no OpenCL event");
00023   }
```

12.37.2.2 get_boost_compute()

```
boost::compute::event & trisycl::detail::host_event::get_boost_compute ( ) const  [inline],
[override], [virtual]
```

Return the underlying Boost.Compute event, if any.

Implements trisycl::detail::event.

Definition at line 25 of file host_event.hpp.

```
00025                                                  {
00026     throw
00027       non_cl_error("The host device has no underlying Boost Compute event");
00028   }
```

12.37.2.3 get_command_execution_status()

```
info::event_command_status trisycl::detail::host_event::get_command_execution_status ( ) const
[inline], [override], [virtual]
```

Implements trisycl::detail::event.

Definition at line 39 of file host_event.hpp.

```
00039                                                                {
00040     return info::event_command_status::submitted;
00041   }
```

12.37.2.4 get_profiling_info()

```
cl_ulong trisycl::detail::host_event::get_profiling_info (
             info::event_profiling param ) const  [inline], [override], [virtual]
```

Implements trisycl::detail::event.

Definition at line 43 of file host_event.hpp.

```
00043                                                                    {
00044     return 0;
00045   }
```

12.37.2.5 get_reference_count()

```
cl_uint trisycl::detail::host_event::get_reference_count ( ) const  [inline], [override],
[virtual]
```

Implements trisycl::detail::event.

Definition at line 35 of file host_event.hpp.

```
00035                                              {
00036     return 0;
00037   }
```

12.37.2.6 is_host()

```
bool trisycl::detail::host_event::is_host ( ) const  [inline], [override], [virtual]
```

Implements trisycl::detail::event.

Definition at line 31 of file host_event.hpp.

```
00031                                    {
00032     return true;
00033   }
```

12.37.2.7 wait()

```
void trisycl::detail::host_event::wait ( ) const  [inline], [override], [virtual]
```

Implements trisycl::detail::event.

Definition at line 47 of file host_event.hpp.

```
00047                                   {
00048   }
```

The documentation for this class was generated from the following file:

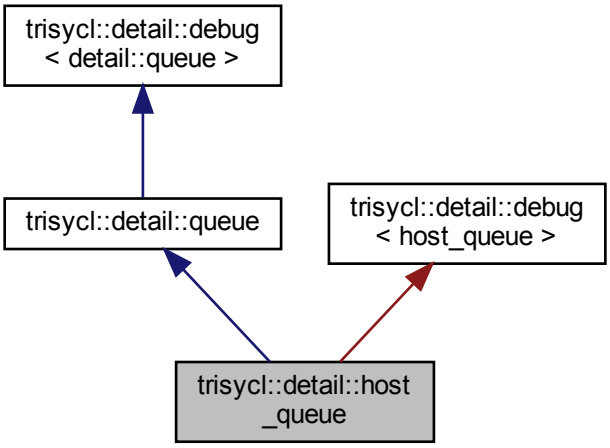
- include/triSYCL/event/detail/host_event.hpp

12.38 trisycl::detail::host_queue Class Reference

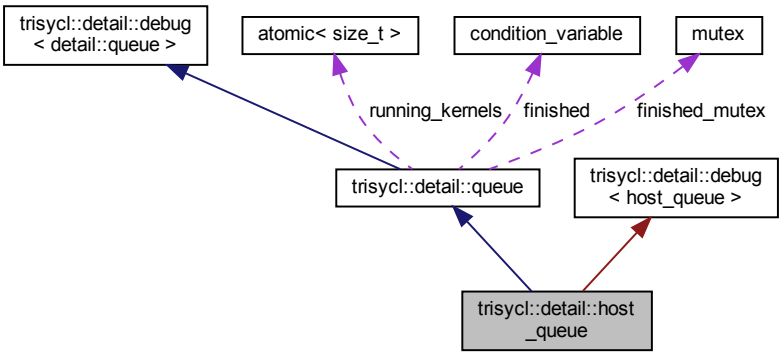
Some implementation details about the SYCL queue.

`#include <host_queue.hpp>`

Inheritance diagram for trisycl::detail::host_queue:



Collaboration diagram for trisycl::detail::host_queue:



Private Member Functions

- cl_command_queue get () const override

  Return the cl_command_queue of the underlying OpenCL queue.
- boost::compute::command_queue & get_boost_compute () override

Return the underlying Boost.Compute command queue.

- trisycl::context get_context () const override

  Return the SYCL host queue's host context.

- trisycl::device get_device () const override

  Return the SYCL host device the host queue is associated with.

- bool is_host () const override

  Claim proudly that the queue is executing on the SYCL host device.

Additional Inherited Members

12.38.1 Detailed Description

Some implementation details about the SYCL queue.

Note that a host queue is not a singleton, compared to host device or host platform, for example.

Definition at line 28 of file host_queue.hpp.

12.38.2 Member Function Documentation

12.38.2.1 get()

```
cl_command_queue trisycl::detail::host_queue::get ( ) const  [inline], [override], [private],
[virtual]
```

Return the cl_command_queue of the underlying OpenCL queue.

This throws an error since there is no OpenCL queue associated to the host queue.

Implements trisycl::detail::queue.

Definition at line 37 of file host_queue.hpp.

```
00037                                       {
00038     throw non_cl_error("The host queue has no OpenCL command queue");
00039   }
```

12.38.2.2 get_boost_compute()

```
boost::compute::command_queue & trisycl::detail::host_queue::get_boost_compute ( )  [inline],
[override], [private], [virtual]
```

Return the underlying Boost.Compute command queue.

This throws an error since there is no OpenCL queue associated to the host queue.

Implements trisycl::detail::queue.

Definition at line 47 of file host_queue.hpp.

```
00047                                                       {
00048     throw non_cl_error("The host queue has no OpenCL command queue");
00049   }
```

12.38.2.3 get_context()

```
trisycl::context trisycl::detail::host_queue::get_context ( ) const  [inline], [override],
[private], [virtual]
```

Return the SYCL host queue's host context.

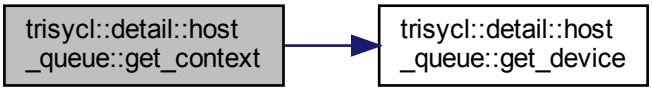
Implements trisycl::detail::queue.

Definition at line 54 of file host_queue.hpp.

```
00054                                          {
00055     // Return the default context for the default device
00056     // which is the host context
00057     return { get_device() };
00058   }
```

References get_device().

Here is the call graph for this function:



12.38.2.4 get_device()

```
trisycl::device trisycl::detail::host_queue::get_device ( ) const  [inline], [override], [private],
[virtual]
```

Return the SYCL host device the host queue is associated with.

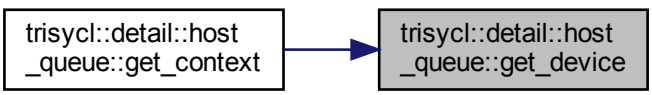
Implements trisycl::detail::queue.

Definition at line 62 of file host_queue.hpp.

```
00062                                          {
00063     // Return the default device which is the host device
00064     return {};
00065   }
```

Referenced by get_context().

Here is the caller graph for this function:



12.38.2.5 is_host()

```
bool trisycl::detail::host_queue::is_host ( ) const  [inline], [override], [private], [virtual]
```

Claim proudly that the queue is executing on the SYCL host device.

Implements trisycl::detail::queue.

Definition at line 69 of file host_queue.hpp.

```
00069                                {
00070     return true;
00071   }
```

The documentation for this class was generated from the following file:

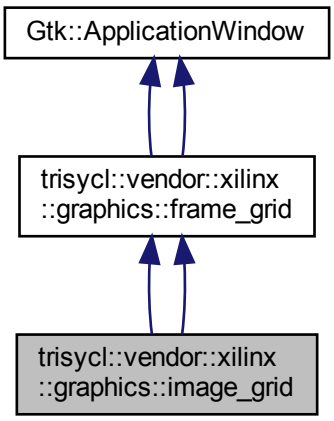
- include/triSYCL/queue/detail/host_queue.hpp

12.39 trisycl::vendor::xilinx::graphics::image_grid Struct Reference

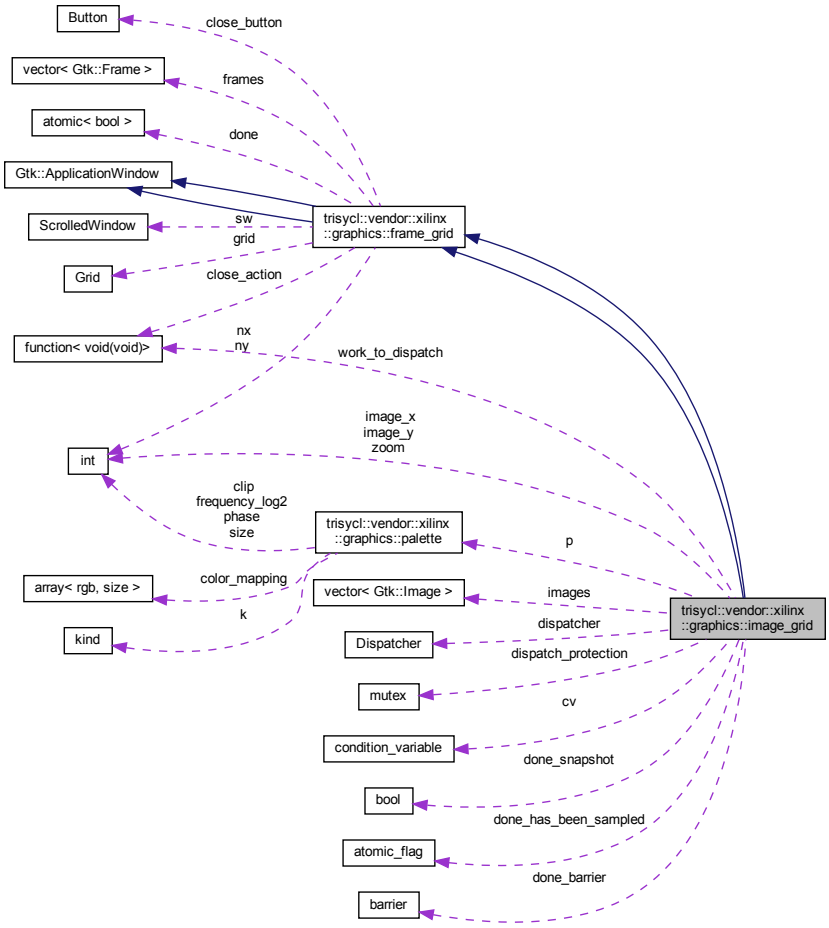
An application window displaying a grid of tiled images.

```
#include <graphics.hpp>
```

Inheritance diagram for trisycl::vendor::xilinx::graphics::image_grid:



Collaboration diagram for trisycl::vendor::xilinx::graphics::image_grid:



Public Member Functions

- image_grid (int nx, int ny, int image_x, int image_y, int zoom)

  Create a grid of tiled images.
- bool on_key_press_event (GdkEventKey ∗event) override

  Deal with the key pressed.
- void submit (std::function< void(void)> f)

  Submit some work to the graphics thread.
- template<typename MDspan , typename RangeValue , typename = typename MDspan::extents_type>
  void update_tile_data_image (int x, int y, const MDspan &data, RangeValue min_value, RangeValue max_↩
  value)

  Update the image of a tile.
- template<typename DataType , typename RangeValue >
  void update_tile_data_image (int x, int y, const DataType ∗data, RangeValue min_value, RangeValue max↩
  _value)

  Update the image of a tile of size image_y by image_x from a pointer.
- palette & get_palette ()

  Return the palette used to render the value.
- bool is_done_barrier ()

Test if the window has been closed after synchronizing with a barrier accross all the tiles.

- image_grid (int nx, int ny, int image_x, int image_y, int zoom)

  Create a grid of tiled images.
- bool on_key_press_event (GdkEventKey *event) override

  Deal with the key pressed.
- void submit (std::function< void(void)> f)

  Submit some work to the graphics thread.
- template<typename MDspan , typename RangeValue , typename = typename MDspan::extents_type>
  void update_tile_data_image (int x, int y, const MDspan &data, RangeValue min_value, RangeValue max_↩
  value)

  Update the image of a tile.
- template<typename DataType , typename RangeValue >
  void update_tile_data_image (int x, int y, const DataType *data, RangeValue min_value, RangeValue max↩
  _value)

  Update the image of a tile of size image_y by image_x from a pointer.
- palette & get_palette ()

  Return the palette used to render the value.
- bool is_done_barrier ()

  Test if the window has been closed after synchronizing with a barrier accross all the tiles.

Public Attributes

- std::vector< Gtk::Image > images

  The linearized 2D vector of images.
- int image_x

  Horizontal image size in pixels.
- int image_y

  Vertical image size in pixels.
- int zoom

  The image pixel zooming factor for both dimension.
- Glib::Dispatcher dispatcher

  Dispatcher to invoke something in the graphics thread in a safe way.
- std::mutex dispatch_protection

  Protection against concurrent update.
- std::condition_variable cv

  The condition variable used to queue up graphics update submission.
- std::function< void(void)> work_to_dispatch

  What to dispatch.
- palette p

  The RGB palette used to render the image values.
- bool done_snapshot

  A global done state across all the tile to allow a global shutdown.
- std::atomic_flag done_has_been_sampled

  Set to true once "done" state has been sampled.
- boost::barrier done_barrier { static_cast<unsigned int>(nx*ny) }

  Barrier to compute the global done state across the tiles.

12.39.1 Detailed Description

An application window displaying a grid of tiled images.

Definition at line 372 of file graphics.hpp.

12.39.2 Constructor & Destructor Documentation

12.39.2.1 image_grid() [1/2]

```
trisycl::vendor::xilinx::graphics::image_grid::image_grid (
            int nx,
            int ny,
            int image_x,
            int image_y,
            int zoom )  [inline]
```

Create a grid of tiled images.

Parameters

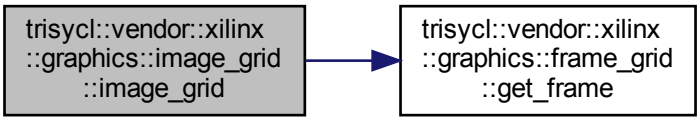
in	*nx*	is the numer of tiles horizontally
in	*ny*	is the numer of tiles vertically
in	*image_x*	is the horizontal image pixel size
in	*image_y*	is the vertical image pixel size
in	*zoom*	is the zooming factor applied to image pixels, both horizontally and vertically

Definition at line 422 of file graphics.hpp.

```
00423     :  frame_grid { nx , ny }
00424     , image_x { image_x }
00425     , image_y { image_y }
00426     , zoom { zoom }
00427   {
00428     for (int y = 0; y < ny; ++y)
00429       for (int x = 0; x < nx; ++x) {
00430         auto &f = get_frame(x, y);
00431         auto pb = Gdk::Pixbuf::create(Gdk::Colorspace::COLORSPACE_RGB
00432                                       , false //< has_alpha
00433                                       , 8 //< bits_per_sample
00434                                       , image_x*zoom //< width
00435                                       , image_y*zoom //< height
00436                                         );
00437         images.emplace_back(pb);
00438         // Display the frame with the lower y to the South
00439         f.add(images.back());
00440       }
00441
00442     // Hook a generic dispatcher
00443     dispatcher.connect([&] {
00444         {
00445           // Only 1 customer at a time
00446           std::lock_guard lock { dispatch_protection };
00447           // Skip the work when done to avoid dead lock
00448           if (done)
00449             // Wake-up everybody waiting for sending some work to
00450             // realize that they have to give up on their hope
00451             cv.notify_all();
00452           else
00453             work_to_dispatch();
00454           // Discard the previous work
00455           work_to_dispatch = nullptr;
00456         }
00457         // We can serve the next customer
00458         cv.notify_one();
00459       });
00460
00461     // Go live
00462     show_all_children();
00463   }
```

References cv, dispatch_protection, dispatcher, trisycl::vendor::xilinx::graphics::frame_grid::done, trisycl::vendor::xilinx::graphics::fram
image_x, image_y, images, trisycl::vendor::xilinx::graphics::frame_grid::nx, trisycl::vendor::xilinx::graphics::frame_grid::ny,
work_to_dispatch, x, trisycl::y, and zoom.

Here is the call graph for this function:



12.39.2.2 image_grid() [2/2]

```
trisycl::vendor::xilinx::graphics::image_grid::image_grid (
              int nx,
              int ny,
              int image_x,
              int image_y,
              int zoom )  [inline]
```

Create a grid of tiled images.

Parameters

in	*nx*	is the numer of tiles horizontally
in	*ny*	is the numer of tiles vertically
in	*image_x*	is the horizontal image pixel size
in	*image_y*	is the vertical image pixel size
in	*zoom*	is the zooming factor applied to image pixels, both horizontally and vertically

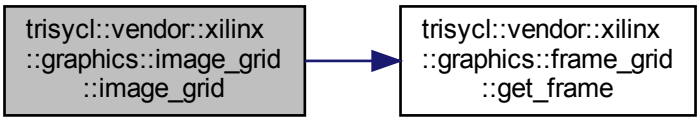
Definition at line 422 of file graphics.hpp.

```
00423     :  frame_grid { nx , ny }
00424     , image_x { image_x }
00425     , image_y { image_y }
00426     , zoom { zoom }
00427   {
00428     for (int y = 0; y < ny; ++y)
00429       for (int x = 0; x < nx; ++x) {
00430         auto &f = get_frame(x, y);
00431         auto pb = Gdk::Pixbuf::create(Gdk::Colorspace::COLORSPACE_RGB
00432                                       , false //< has_alpha
00433                                       , 8 //< bits_per_sample
00434                                       , image_x*zoom //< width
00435                                       , image_y*zoom //< height
00436                                         );
00437         images.emplace_back(pb);
00438         // Display the frame with the lower y to the South
00439         f.add(images.back());
```

```
00440       }
00441
00442     // Hook a generic dispatcher
00443     dispatcher.connect([&] {
00444         {
00445           // Only 1 customer at a time
00446           std::lock_guard lock { dispatch_protection };
00447           // Skip the work when done to avoid dead lock
00448           if (done)
00449             // Wake-up everybody waiting for sending some work to
00450             // realize that they have to give up on their hope
00451             cv.notify_all();
00452           else
00453             work_to_dispatch();
00454           // Discard the previous work
00455           work_to_dispatch = nullptr;
00456         }
00457         // We can serve the next customer
00458         cv.notify_one();
00459       });
00460
00461     // Go live
00462     show_all_children();
00463   }
```

References cv, dispatch_protection, dispatcher, trisycl::vendor::xilinx::graphics::frame_grid::done, trisycl::vendor::xilinx::graphics::fram
image_x, image_y, images, trisycl::vendor::xilinx::graphics::frame_grid::nx, trisycl::vendor::xilinx::graphics::frame_grid::ny,
work_to_dispatch, x, trisycl::y, and zoom.

Here is the call graph for this function:



12.39.3 Member Function Documentation

12.39.3.1 get_palette() [1/2]

```
palette & trisycl::vendor::xilinx::graphics::image_grid::get_palette ( )  [inline]
```

Return the palette used to render the value.

Definition at line 571 of file graphics.hpp.

```
00571                                {
00572     return p;
00573   }
```

References p.

12.39.3.2 get_palette() [2/2]

```
palette & trisycl::vendor::xilinx::graphics::image_grid::get_palette ( )  [inline]
```

Return the palette used to render the value.

Definition at line 571 of file graphics.hpp.

```
00571                                {
00572     return p;
00573   }
```

References p.

12.39.3.3 is_done_barrier() [1/2]

```
bool trisycl::vendor::xilinx::graphics::image_grid::is_done_barrier ( )  [inline]
```

Test if the window has been closed after synchronizing with a barrier accross all the tiles.

Definition at line 578 of file graphics.hpp.

```
00578                                {
00579     // Only 1 thread sample the graphics status
00580     if (done_has_been_sampled.test_and_set())
00581       done_snapshot = done;
00582     // Then wait for everybody to synchronize
00583     done_barrier.count_down_and_wait();
00584     // Reinitialize the flag for the next iteration
00585     done_has_been_sampled.clear();
00586     return done_snapshot;
00587   }
```

References trisycl::vendor::xilinx::graphics::frame_grid::done, done_barrier, done_has_been_sampled, and done_snapshot.

12.39.3.4 is_done_barrier() [2/2]

```
bool trisycl::vendor::xilinx::graphics::image_grid::is_done_barrier ( )  [inline]
```

Test if the window has been closed after synchronizing with a barrier accross all the tiles.

Definition at line 578 of file graphics.hpp.

```
00578                                {
00579     // Only 1 thread sample the graphics status
00580     if (done_has_been_sampled.test_and_set())
00581       done_snapshot = done;
00582     // Then wait for everybody to synchronize
00583     done_barrier.count_down_and_wait();
00584     // Reinitialize the flag for the next iteration
00585     done_has_been_sampled.clear();
00586     return done_snapshot;
00587   }
```

References trisycl::vendor::xilinx::graphics::frame_grid::done, done_barrier, done_has_been_sampled, and done_snapshot.

12.39.3.5 on_key_press_event() [1/2]

```
bool trisycl::vendor::xilinx::graphics::image_grid::on_key_press_event (
            GdkEventKey * event )  [inline], [override]
```

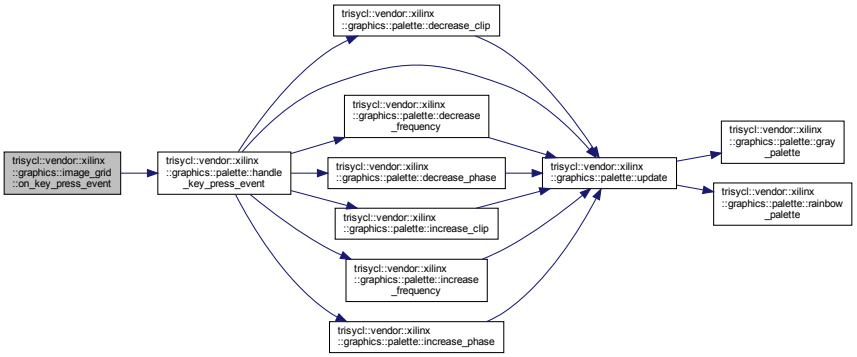
Deal with the key pressed.

Definition at line 467 of file graphics.hpp.

```
00467                                                   {
00468     // Forward to the palette for action
00469     return p.handle_key_press_event(*event);
00470   }
```

References trisycl::vendor::xilinx::graphics::palette::handle_key_press_event(), and p.

Here is the call graph for this function:



12.39.3.6 on_key_press_event() [2/2]

```
bool trisycl::vendor::xilinx::graphics::image_grid::on_key_press_event (
            GdkEventKey * event )  [inline], [override]
```

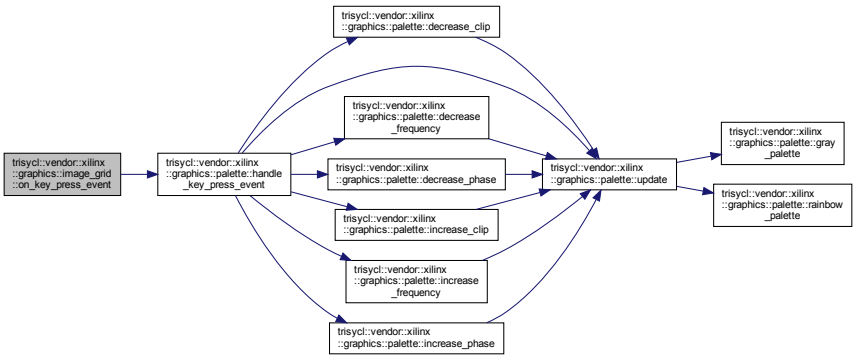
Deal with the key pressed.

Definition at line 467 of file graphics.hpp.

```
00467                                                   {
00468     // Forward to the palette for action
00469     return p.handle_key_press_event(*event);
00470   }
```

References trisycl::vendor::xilinx::graphics::palette::handle_key_press_event(), and p.

Here is the call graph for this function:



12.39.3.7 submit() [1/2]

```
void trisycl::vendor::xilinx::graphics::image_grid::submit (
            std::function< void(void)> f )  [inline]
```

Submit some work to the graphics thread.

Definition at line 473 of file graphics.hpp.

```
00473                                        {
00474     std::unique_lock lock { dispatch_protection };
00475     // Wait for no work being dispatched or the end
00476     cv.wait(lock, [&] { return !work_to_dispatch || done; } );
00477     // Do not submit anything if we are in the shutdown process already
00478     if (!done) {
00479       work_to_dispatch = std::move(f);
00480       // Ask the graphics thread for some work
00481       dispatcher.emit();
00482     }
00483   };
```

References cv, dispatch_protection, dispatcher, trisycl::vendor::xilinx::graphics::frame_grid::done, and work_to_dispatch.

Referenced by update_tile_data_image().

Here is the caller graph for this function:



12.39.3.8 submit() [2/2]

```
void trisycl::vendor::xilinx::graphics::image_grid::submit (
            std::function< void(void)> f )  [inline]
```

Submit some work to the graphics thread.

Definition at line 473 of file graphics.hpp.

```
00473                                          {
00474     std::unique_lock lock { dispatch_protection };
00475     // Wait for no work being dispatched or the end
00476     cv.wait(lock, [&] { return !work_to_dispatch || done; } );
00477     // Do not submit anything if we are in the shutdown process already
00478     if (!done) {
00479       work_to_dispatch = std::move(f);
00480       // Ask the graphics thread for some work
00481       dispatcher.emit();
00482     }
00483   };
```

References cv, dispatch_protection, dispatcher, trisycl::vendor::xilinx::graphics::frame_grid::done, and work_to_dispatch.

12.39.3.9 update_tile_data_image() [1/4]

```
template<typename DataType , typename RangeValue >
void trisycl::vendor::xilinx::graphics::image_grid::update_tile_data_image (
            int x,
            int y,
            const DataType * data,
            RangeValue min_value,
            RangeValue max_value )  [inline]
```

Update the image of a tile of size image_y by image_x from a pointer.

Parameters

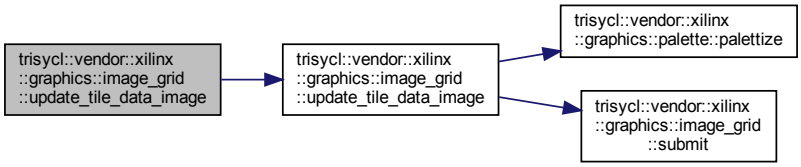
in	*x*	is the tile horizontal id
in	*y*	is the tile vertical id
in	*data*	is a pointer to the pixels to be drawns
in	*min_value*	is the value represented with minimum of graphics palette color
in	*max_value*	is the value represented with maximum of graphics palette color

Definition at line 560 of file graphics.hpp.

```
00563                                                  {
00564     // Wrap the pointed area into an MDspan
00565     const std::mdspan md { data, image_y, image_x };
00566     update_tile_data_image(x, y, md, min_value, max_value);
00567   }
```

References image_x, image_y, update_tile_data_image(), x, and trisycl::y.

Here is the call graph for this function:



12.39.3.10 update_tile_data_image() [2/4]

```
template<typename DataType , typename RangeValue >
void trisycl::vendor::xilinx::graphics::image_grid::update_tile_data_image (
            int x,
            int y,
            const DataType * data,
            RangeValue min_value,
            RangeValue max_value )  [inline]
```

Update the image of a tile of size image_y by image_x from a pointer.

Parameters

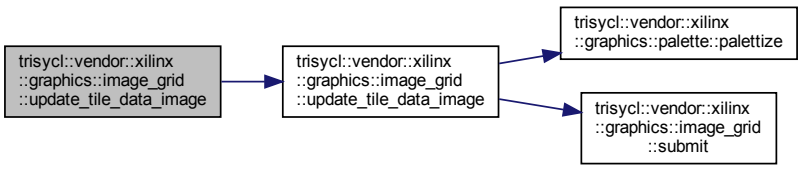
in	*x*	is the tile horizontal id
in	*y*	is the tile vertical id
in	*data*	is a pointer to the pixels to be drawns
in	*min_value*	is the value represented with minimum of graphics palette color
in	*max_value*	is the value represented with maximum of graphics palette color

Definition at line 560 of file graphics.hpp.

```
00563                                                 {
00564     // Wrap the pointed area into an MDspan
00565     const std::mdspan md { data, image_y, image_x };
00566     update_tile_data_image(x, y, md, min_value, max_value);
00567   }
```

References image_x, image_y, update_tile_data_image(), x, and trisycl::y.

Here is the call graph for this function:



12.39.3.11 update_tile_data_image() [3/4]

```
template<typename MDspan , typename RangeValue , typename = typename MDspan::extents_type>
void trisycl::vendor::xilinx::graphics::image_grid::update_tile_data_image (
            int x,
            int y,
            const MDspan & data,
            RangeValue min_value,
            RangeValue max_value )  [inline]
```

Update the image of a tile.

Parameters

in	*x*	is the tile horizontal id
in	*y*	is the tile vertical id
in	*data*	is a 2D MDspan of extent at most image_y by image_x. Only the pixels of the extents are drawn
in	*min_value*	is the value represented with minimum of graphics palette color
in	*max_value*	is the value represented with maximum of graphics palette color

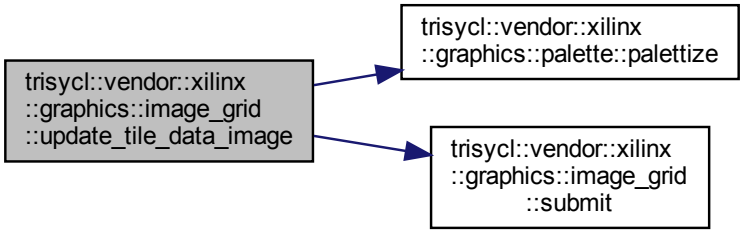
Definition at line 506 of file graphics.hpp.

```
00509                                                   {
00510     /* Painful:  first I cannot use a std::unique_ptr because the
00511 std::function below needs to be copy-constructible, then the
00512 std::make_shared taking an array size comes only in C++20 while
00513 it is available for std::make_unique in C++17...  */
00514     std::shared_ptr<std::uint8_t[]> d { new std::uint8_t[3*image_x*image_y] };
00515     std::mdspan output { reinterpret_cast<rgb*>(d.get()), image_y, image_x };
00516     // For each pixel of the md_span or of the image, which one is smaller
00517     for (int j = 0; j < std::min(static_cast<int>(data.extent(0)), image_y);
00518          ++j)
00519       for (int i = 0; i < std::min(static_cast<int>(data.extent(1)), image_x);
00520            ++i) {
00521         /* Mirror the image vertically to display the pixels in a
00522 mathematical sense */
00523         output[image_y - 1 - j, i] =
00524             p.palettize(data[j, i], min_value, max_value);
00525       }
00526     // Send the graphics updating code
00527     submit([=, this] {
00528         // Create a first buffer, allowing later zooming
00529         auto pb = Gdk::Pixbuf::create_from_data(d.get()
00530                                                 , Gdk::Colorspace::COLORSPACE_RGB
00531                                                 , false//< has_alpha
00532                                                 , 8 //< bits_per_sample
00533                                                 , image_x //< width
00534                                                 , image_y //< height
00535                                                 , image_x*3 //< rowstride
00536                                                );
00537         // Update the pixel buffer of the image with some zooming
00538         images.at(x + nx*y).set(pb->scale_simple(image_x*zoom,
00539                                                  image_y*zoom,
00540                                                  Gdk::INTERP_NEAREST));
00541       });
00542   }
```

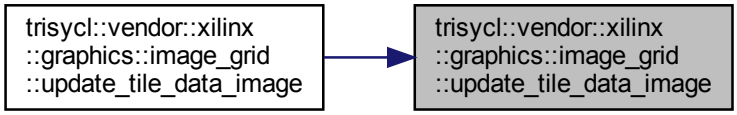
References i, image_x, image_y, images, trisycl::vendor::xilinx::graphics::frame_grid::nx, p, trisycl::vendor::xilinx::graphics::palette::pa submit(), x, trisycl::y, and zoom.

Referenced by update_tile_data_image().

Here is the call graph for this function:



Here is the caller graph for this function:



12.39.3.12 update_tile_data_image() [4/4]

```
template<typename MDspan , typename RangeValue , typename = typename MDspan::extents_type>
void trisycl::vendor::xilinx::graphics::image_grid::update_tile_data_image (
            int x,
            int y,
            const MDspan & data,
            RangeValue min_value,
            RangeValue max_value )  [inline]
```

Update the image of a tile.

Parameters

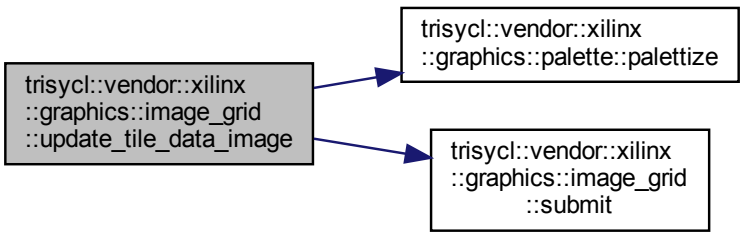
in	*x*	is the tile horizontal id
in	*y*	is the tile vertical id
in	*data*	is a 2D MDspan of extent at most image_y by image_x. Only the pixels of the extents are drawn
in	*min_value*	is the value represented with minimum of graphics palette color
in	*max_value*	is the value represented with maximum of graphics palette color

Definition at line 506 of file graphics.hpp.

```
00509                                                 {
00510     /* Painful:  first I cannot use a std::unique_ptr because the
00511 std::function below needs to be copy-constructible, then the
00512 std::make_shared taking an array size comes only in C++20 while
00513 it is available for std::make_unique in C++17...  */
00514     std::shared_ptr<std::uint8_t[]> d { new std::uint8_t[3*image_x*image_y] };
00515     std::mdspan output { reinterpret_cast<rgb*>(d.get()), image_y, image_x };
00516     // For each pixel of the md_span or of the image, which one is smaller
00517     for (int j = 0; j < std::min(static_cast<int>(data.extent(0)), image_y);
00518          ++j)
00519       for (int i = 0; i < std::min(static_cast<int>(data.extent(1)), image_x);
00520            ++i) {
00521         /* Mirror the image vertically to display the pixels in a
00522 mathematical sense */
00523         output[image_y - 1 - j, i] =
00524             p.palettize(data[j, i], min_value, max_value);
00525       }
00526     // Send the graphics updating code
00527     submit([=, this] {
00528         // Create a first buffer, allowing later zooming
00529         auto pb = Gdk::Pixbuf::create_from_data(d.get()
00530                                                 , Gdk::Colorspace::COLORSPACE_RGB
00531                                                 , false//< has_alpha
00532                                                 , 8 //< bits_per_sample
00533                                                 , image_x //< width
00534                                                 , image_y //< height
00535                                                 , image_x*3 //< rowstride
00536                                                );
00537         // Update the pixel buffer of the image with some zooming
00538         images.at(x + nx*y).set(pb->scale_simple(image_x*zoom,
00539                                                  image_y*zoom,
00540                                                  Gdk::INTERP_NEAREST));
00541       });
00542   }
```

References i, image_x, image_y, images, trisycl::vendor::xilinx::graphics::frame_grid::nx, p, trisycl::vendor::xilinx::graphics::palette::pa submit(), x, trisycl::y, and zoom.

Here is the call graph for this function:



12.39.4 Member Data Documentation

12.39.4.1 cv

`std::condition_variable trisycl::vendor::xilinx::graphics::image_grid::cv`

The condition variable used to queue up graphics update submission.

Definition at line 392 of file graphics.hpp.

Referenced by image_grid(), and submit().

12.39.4.2 dispatch_protection

```
std::mutex trisycl::vendor::xilinx::graphics::image_grid::dispatch_protection
```

Protection against concurrent update.

Definition at line 389 of file graphics.hpp.

Referenced by image_grid(), and submit().

12.39.4.3 dispatcher

```
Glib::Dispatcher trisycl::vendor::xilinx::graphics::image_grid::dispatcher
```

Dispatcher to invoke something in the graphics thread in a safe way.

Definition at line 386 of file graphics.hpp.

Referenced by image_grid(), and submit().

12.39.4.4 done_barrier

```
boost::barrier trisycl::vendor::xilinx::graphics::image_grid::done_barrier { static_cast<unsigned
int>(nx*ny) }
```

Barrier to compute the global done state across the tiles.

Definition at line 407 of file graphics.hpp.

Referenced by is_done_barrier().

12.39.4.5 done_has_been_sampled

```
std::atomic_flag trisycl::vendor::xilinx::graphics::image_grid::done_has_been_sampled
```

Set to true once "done" state has been sampled.

Definition at line 404 of file graphics.hpp.

Referenced by is_done_barrier().

12.39.4.6 done_snapshot

```
bool trisycl::vendor::xilinx::graphics::image_grid::done_snapshot
```

A global done state across all the tile to allow a global shutdown.

Definition at line 401 of file graphics.hpp.

Referenced by is_done_barrier().

12.39.4.7 image_x

```
int trisycl::vendor::xilinx::graphics::image_grid::image_x
```

Horizontal image size in pixels.

Definition at line 377 of file graphics.hpp.

Referenced by image_grid(), and update_tile_data_image().

12.39.4.8 image_y

```
int trisycl::vendor::xilinx::graphics::image_grid::image_y
```

Vertical image size in pixels.

Definition at line 380 of file graphics.hpp.

Referenced by image_grid(), and update_tile_data_image().

12.39.4.9 images

```
std::vector< Gtk::Image > trisycl::vendor::xilinx::graphics::image_grid::images
```

The linearized 2D vector of images.

Definition at line 374 of file graphics.hpp.

Referenced by image_grid(), and update_tile_data_image().

12.39.4.10 p

```
palette trisycl::vendor::xilinx::graphics::image_grid::p
```

The RGB palette used to render the image values.

Definition at line 398 of file graphics.hpp.

Referenced by get_palette(), on_key_press_event(), and update_tile_data_image().

12.39.4.11 work_to_dispatch

```
std::function< void(void)> trisycl::vendor::xilinx::graphics::image_grid::work_to_dispatch
```

What to dispatch.

Definition at line 395 of file graphics.hpp.

Referenced by image_grid(), and submit().

12.39.4.12 zoom

```
int trisycl::vendor::xilinx::graphics::image_grid::zoom
```

The image pixel zooming factor for both dimension.

Definition at line 383 of file graphics.hpp.

Referenced by image_grid(), and update_tile_data_image().

The documentation for this struct was generated from the following files:

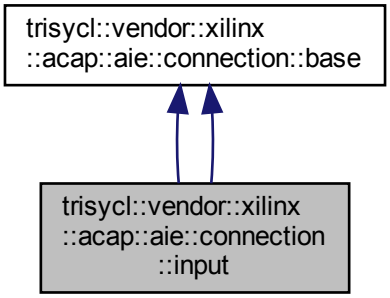
- include/sycl/vendor/Xilinx/graphics.hpp
- include/triSYCL/vendor/Xilinx/graphics.hpp

12.40 trisycl::vendor::xilinx::acap::aie::connection::input Struct Reference

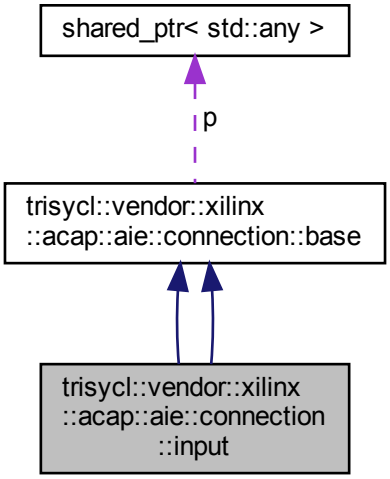
Abstract an input port.

`#include <connection.hpp>`

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::connection::input:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::connection::input:



Public Member Functions

- template<typename InputT , access::target Target>
  auto in ()

  Get the input port.
- template<typename InputT , access::target Target>
  auto in ()

  Get the input port.
- base (const connection &c)

  Recycle the base constructors.
- base ()=default

  Recycle the base constructors.
- base (const connection &c)

  Recycle the base constructors.
- base ()=default

  Recycle the base constructors.
- base (const connection &c)

  Recycle the base constructors.
- base ()=default

  Recycle the base constructors.
- base (const connection &c)

  Recycle the base constructors.
- base ()=default

  Recycle the base constructors.

Additional Inherited Members

12.40.1 Detailed Description

Abstract an input port.

Definition at line 321 of file connection.hpp.

12.40.2 Member Function Documentation

12.40.2.1 base() [1/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base ( )  [default]
```

Recycle the base constructors.

12.40.2.2 base() [2/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base ( )  [default]
```

Recycle the base constructors.

12.40.2.3 base() [3/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base ( )  [default]
```

Recycle the base constructors.

12.40.2.4 base() [4/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base ( )  [default]
```

Recycle the base constructors.

12.40.2.5 base() [5/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base (
            const connection & c )  [inline]
```

Recycle the base constructors.

Definition at line 292 of file connection.hpp.

```
00293       : p { c.p } {}
```

12.40.2.6 base() [6/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base (
            const connection & c )  [inline]
```

Recycle the base constructors.

Definition at line 292 of file connection.hpp.

```
00293       : p { c.p } {}
```

12.40.2.7 base() [7/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base (
            const connection & c )  [inline]
```

Recycle the base constructors.

Definition at line 292 of file connection.hpp.

```
00293       :  p { c.p } {}
```

12.40.2.8 base() [8/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base (
            const connection & c )  [inline]
```

Recycle the base constructors.

Definition at line 292 of file connection.hpp.

```
00293       :  p { c.p } {}
```

12.40.2.9 in() [1/2]

```
template<typename InputT , access::target Target>
auto trisycl::vendor::xilinx::acap::aie::connection::input::in ( )  [inline]
```

Get the input port.

Parameters

in	*InputT*	is the data type to be used in the transfers
in	*Target*	specifies if the connection is blocking or not

Exceptions

trisycl::runtime_error	if the input is not connected

Definition at line 335 of file connection.hpp.

```
00335               {
00336       if (!p)
00337         throw ::trisycl::runtime_error { "This input is not connected" };
00338       return
00339         pipe_of<InputT>().template get_access<access::mode::read, Target>();
00340     }
```

References trisycl::vendor::xilinx::acap::aie::connection::base::p.

12.40.2.10 in() [2/2]

```
template<typename InputT , access::target Target>
auto trisycl::vendor::xilinx::acap::aie::connection::input::in ( )  [inline]
```

Get the input port.

Parameters

in	*InputT*	is the data type to be used in the transfers
in	*Target*	specifies if the connection is blocking or not

Exceptions

trisycl::runtime_error	if the input is not connected

Definition at line 335 of file connection.hpp.

```
00335               {
00336       if (!p)
00337         throw ::trisycl::runtime_error { "This input is not connected" };
00338       return
00339         pipe_of<InputT>().template get_access<access::mode::read, Target>();
00340     }
```

References trisycl::vendor::xilinx::acap::aie::connection::base::p.

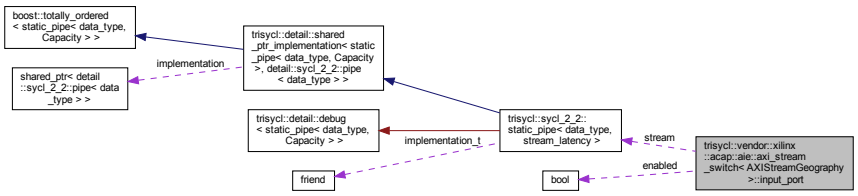
The documentation for this struct was generated from the following files:

- include/sycl/vendor/Xilinx/acap/aie/connection.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/connection.hpp

12.41 trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input_port Struct Reference

```
#include <axi_stream_switch.hpp>
```

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input_↩ port:



Public Attributes

- ::trisycl::sycl_2_2::static_pipe< data_type, stream_latency > stream
- bool enabled

12.41.1 Detailed Description

template<typename AXIStreamGeography>
struct trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input_port

Definition at line 337 of file axi_stream_switch.hpp.

12.41.2 Member Data Documentation

12.41.2.1 enabled

```
template<typename AXIStreamGeography >
bool trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input_port↩
::enabled
```

Definition at line 339 of file axi_stream_switch.hpp.

12.41.2.2 stream

```
template<typename AXIStreamGeography >
trisycl::sycl_2_2::static_pipe< data_type, stream_latency > trisycl::vendor::xilinx::acap::aie::axi_stream_sw
AXIStreamGeography >::input_port::stream
```

Definition at line 338 of file axi_stream_switch.hpp.

The documentation for this struct was generated from the following files:

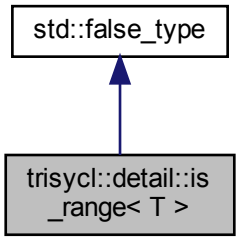
- include/sycl/vendor/Xilinx/acap/aie/axi_stream_switch.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/axi_stream_switch.hpp

12.42 trisycl::detail::is_range< T > Struct Template Reference

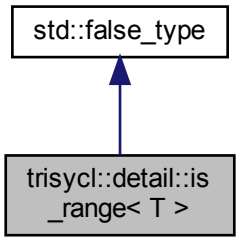
A type trait to check if a type it a sycl::range or not.

```
#include <range.hpp>
```

Inheritance diagram for trisycl::detail::is_range< T >:



Collaboration diagram for trisycl::detail::is_range< T >:



12.42.1 Detailed Description

template<typename T>
struct trisycl::detail::is_range< T >

A type trait to check if a type it a sycl::range or not.

Definition at line 83 of file range.hpp.

The documentation for this struct was generated from the following file:

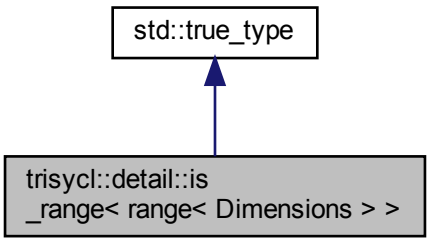
- include/triSYCL/range.hpp

12.43 trisycl::detail::is_range< range< Dimensions > > Struct Template Reference

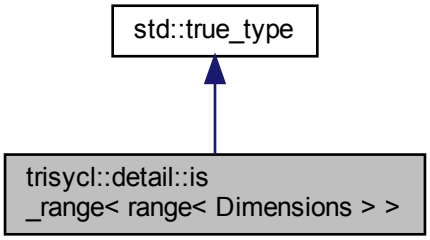
Return true if template instantiating match a sycl::range.

`#include <range.hpp>`

Inheritance diagram for trisycl::detail::is_range< range< Dimensions > >:



Collaboration diagram for trisycl::detail::is_range< range< Dimensions > >:



12.43.1 Detailed Description

template<int Dimensions>
struct trisycl::detail::is_range< range< Dimensions > >

Return true if template instantiating match a sycl::range.

Definition at line 87 of file range.hpp.

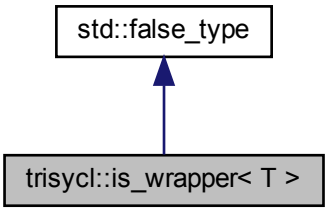
The documentation for this struct was generated from the following file:

- include/triSYCL/range.hpp

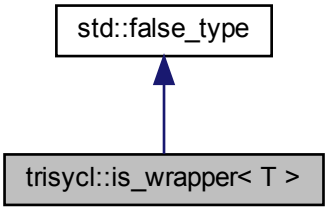
12.44 trisycl::is_wrapper< T > Struct Template Reference

`#include <opencl_types.hpp>`

Inheritance diagram for trisycl::is_wrapper< T >:



Collaboration diagram for trisycl::is_wrapper< T >:



12.44.1 Detailed Description

template<class T>
struct trisycl::is_wrapper< T >

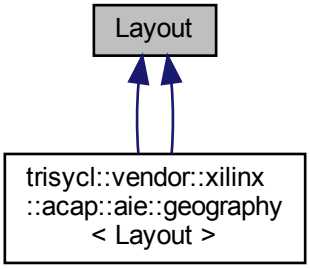
Definition at line 83 of file opencl_types.hpp.

The documentation for this struct was generated from the following file:

- include/triSYCL/opencl_types.hpp

12.45 Layout Class Reference

Inheritance diagram for Layout:



The documentation for this class was generated from the following file:

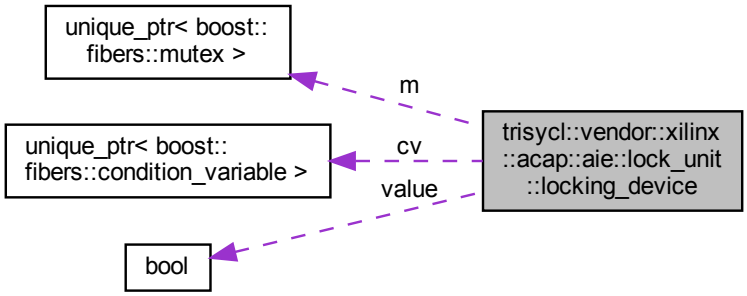
- include/sycl/vendor/Xilinx/acap/aie/geography.hpp

12.46 trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device Struct Reference

The individual locking system.

`#include <lock.hpp>`

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device:



Public Types

- using value_t = bool

  The type of the value stored in the locking device.
- using value_t = bool

  The type of the value stored in the locking device.

Public Member Functions

- void acquire ()

  Lock the mutex.
- void release ()

  Unlock the mutex.
- void acquire_with_value (value_t expectation)

  Wait until the internal value has the expectation.
- void release_with_value (value_t new_value)

  Release and update with a new internal value.
- void acquire ()

  Lock the mutex.
- void release ()

  Unlock the mutex.
- void acquire_with_value (value_t expectation)

  Wait until the internal value has the expectation.
- void release_with_value (value_t new_value)

  Release and update with a new internal value.

Public Attributes

- std::unique_ptr< boost::fibers::mutex > m { new boost::fibers::mutex { } }

  The mutex to provide the basic protection mechanism.
- std::unique_ptr< boost::fibers::condition_variable > cv

  The condition variable to wait/notify for some value.
- value_t value = false

  The value to be waited for, initialized to false on reset.

12.46.1 Detailed Description

The individual locking system.

Definition at line 44 of file lock.hpp.

12.46.2 Member Typedef Documentation

12.46.2.1 value_t [1/2]

```
using trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device::value_t = bool
```

The type of the value stored in the locking device.

Definition at line 46 of file lock.hpp.

12.46.2.2 value_t [2/2]

```
using trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device::value_t = bool
```

The type of the value stored in the locking device.

Definition at line 46 of file lock.hpp.

12.46.3 Member Function Documentation

12.46.3.1 acquire() [1/2]

```
void trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device::acquire ( )  [inline]
```

Lock the mutex.

Definition at line 66 of file lock.hpp.

```
00066                 {
00067       m->lock();
00068     }
```

References m.

12.46.3.2 acquire() [2/2]

```
void trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device::acquire ( )  [inline]
```

Lock the mutex.

Definition at line 66 of file lock.hpp.

```
00066                 {
00067       m->lock();
00068     }
```

References m.

12.46.3.3 acquire_with_value() [1/2]

```
void trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device::acquire_with_value (
            value_t expectation )  [inline]
```

Wait until the internal value has the expectation.

Definition at line 78 of file lock.hpp.

```
00078                                                {
00079       std::unique_lock lk { *m };
00080       cv->wait(lk, [&] { return expectation == value; });
00081     }
```

References cv, m, and value.

12.46.3.4 acquire_with_value() [2/2]

```
void trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device::acquire_with_value (
            value_t expectation )  [inline]
```

Wait until the internal value has the expectation.

Definition at line 78 of file lock.hpp.

```
00078                                                {
00079       std::unique_lock lk { *m };
00080       cv->wait(lk, [&] { return expectation == value; });
00081     }
```

References cv, m, and value.

12.46.3.5 release() [1/2]

```
void trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device::release ( )  [inline]
```

Unlock the mutex.

Definition at line 72 of file lock.hpp.

```
00072                    {
00073       m->unlock();
00074     }
```

References m.

12.46.3.6 release() [2/2]

```
void trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device::release ( )  [inline]
```

Unlock the mutex.

Definition at line 72 of file lock.hpp.

```
00072                    {
00073       m->unlock();
00074     }
```

References m.

12.46.3.7 release_with_value() [1/2]

```
void trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device::release_with_value (
            value_t new_value )  [inline]
```

Release and update with a new internal value.

Definition at line 85 of file lock.hpp.

```
00085                                           {
00086       {
00087         std::unique_lock lk { *m };
00088         value = new_value;
00089       }
00090       // By construction there should be only one client waiting for it
00091       cv->notify_one();
00092     }
```

References cv, m, and value.

12.46.3.8 release_with_value() [2/2]

```
void trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device::release_with_value (
            value_t new_value )  [inline]
```

Release and update with a new internal value.

Definition at line 85 of file lock.hpp.

```
00085                                           {
00086       {
00087         std::unique_lock lk { *m };
00088         value = new_value;
00089       }
00090       // By construction there should be only one client waiting for it
00091       cv->notify_one();
00092     }
```

References cv, m, and value.

12.46.4 Member Data Documentation

12.46.4.1 cv

```
std::unique_ptr< boost::fibers::condition_variable > trisycl::vendor::xilinx::acap::aie↩
::lock_unit::locking_device::cv
```

Initial value:

```
{
      new boost::fibers::condition_variable { } }
```

The condition variable to wait/notify for some value.

Definition at line 59 of file lock.hpp.

Referenced by acquire_with_value(), and release_with_value().

12.46.4.2 m

```
std::unique_ptr< boost::fibers::mutex > trisycl::vendor::xilinx::acap::aie::lock_unit::locking↩
_device::m { new boost::fibers::mutex { } }
```

The mutex to provide the basic protection mechanism.

Definition at line 56 of file lock.hpp.

Referenced by acquire(), acquire_with_value(), release(), and release_with_value().

12.46.4.3 value

```
value_t trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device::value = false
```

The value to be waited for, initialized to false on reset.

Definition at line 63 of file lock.hpp.

Referenced by acquire_with_value(), and release_with_value().

The documentation for this struct was generated from the following files:

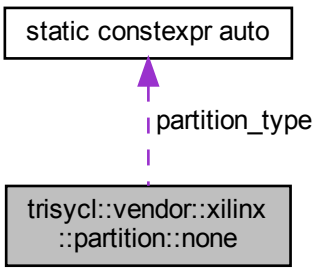
- include/sycl/vendor/Xilinx/acap/aie/lock.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/lock.hpp

12.47 trisycl::vendor::xilinx::partition::none Struct Reference

Represent a none partition.

```
#include <partition_array.hpp>
```

Collaboration diagram for trisycl::vendor::xilinx::partition::none:



Static Public Attributes

- static constexpr auto partition_type = type::none

12.47.1 Detailed Description

Represent a none partition.

The single array would be the same as std::array.

Definition at line 129 of file partition_array.hpp.

12.47.2 Member Data Documentation

12.47.2.1 partition_type

```
static constexpr auto trisycl::vendor::xilinx::partition::none::partition_type = type::none
[static], [constexpr]
```

Definition at line 130 of file partition_array.hpp.

The documentation for this struct was generated from the following files:

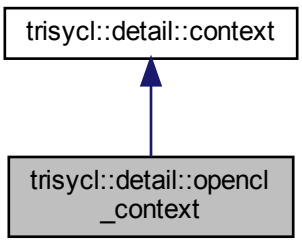
- include/sycl/vendor/Xilinx/fpga/partition_array.hpp
- include/triSYCL/vendor/Xilinx/fpga/partition_array.hpp

12.48 trisycl::detail::opencl_context Class Reference

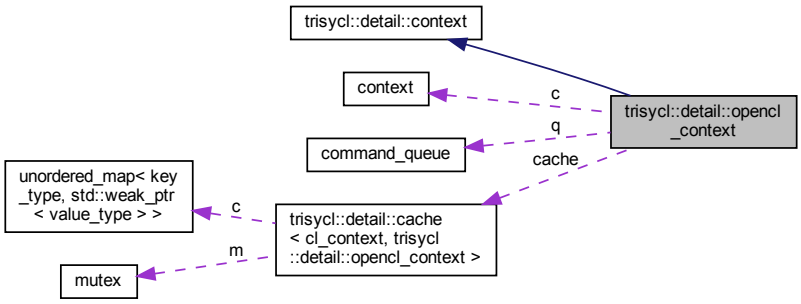
SYCL OpenCL context.

```
#include <opencl_context.hpp>
```

Inheritance diagram for trisycl::detail::opencl_context:



Collaboration diagram for trisycl::detail::opencl_context:



Public Member Functions

- cl_context get () const override

  Return the underlying `cl_context` of the `trisycl::context`.
- boost::compute::context & get_boost_compute () override

  Return the underlying `boost::compute::context` of the `trisycl::context`.
- boost::compute::command_queue & get_boost_queue () override

  Return the queue that is associated to the context.
- bool is_host () const override

  Return false because the context is not a SYCL host context.
- trisycl::platform get_platform () const override

  Return the platform of the context.
- vector_class< trisycl::device > get_devices () const override

  Returns the set of devices that are part of this context.
- trisycl::cl_uint get_reference_count () const override

  Returns the reference count of underlying CL context, or 0 on host.
- ~opencl_context () override

  Unregister from the cache on destruction.

Static Public Member Functions

- static std::shared_ptr< opencl_context > instance (const boost::compute::context &c)

  Get a singleton instance of the `opencl_context`.

Private Member Functions

- opencl_context (const boost::compute::context &c)

  Only the instance factory can build it.

Private Attributes

- boost::compute::context c

  User the Boost Compute abstraction of the OpenCL context.
- boost::compute::command_queue q

  A boost `command_queue` associated to an OpenCL context for when we need to transfer data but no queue is given (eg.

Static Private Attributes

- static detail::cache< cl_context, detail::opencl_context > cache

  A cache to always return the same alive context for a given OpenCL context.

12.48.1 Detailed Description

SYCL OpenCL context.

Definition at line 28 of file opencl_context.hpp.

12.48.2 Constructor & Destructor Documentation

12.48.2.1 opencl_context()

```
trisycl::detail::opencl_context::opencl_context (
            const boost::compute::context & c )  [inline], [private]
```

Only the instance factory can build it.

Definition at line 126 of file opencl_context.hpp.

```
00126                                          :
00127     c { c },
00128     q { boost::compute::command_queue { c, c.get_device() } } {}
```

12.48.2.2 ~opencl_context()

```
trisycl::detail::opencl_context::~opencl_context ( )  [inline], [override]
```

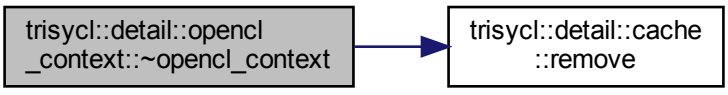
Unregister from the cache on destruction.

Definition at line 133 of file opencl_context.hpp.

```
00133                                  {
00134     cache.remove(c.get());
00135   }
```

References c, and trisycl::detail::cache< Key, Value >::remove().

Here is the call graph for this function:



12.48.3 Member Function Documentation

12.48.3.1 get()

```
cl_context trisycl::detail::opencl_context::get ( ) const  [inline], [override], [virtual]
```

Return the underlying `cl_context` of the `trisycl::context`.

Implements trisycl::detail::context.

Definition at line 49 of file opencl_context.hpp.

```
00049                                    {
00050     return c.get();
00051   }
```

References c.

12.48.3.2 get_boost_compute()

```
boost::compute::context & trisycl::detail::opencl_context::get_boost_compute ( )  [inline],
[override], [virtual]
```

Return the underlying `boost::compute::context` of the `trisycl::context`.

Implements trisycl::detail::context.

Definition at line 57 of file opencl_context.hpp.

```
00057                                                {
00058     return c;
00059   }
```

References c.

12.48.3.3 get_boost_queue()

```
boost::compute::command_queue & trisycl::detail::opencl_context::get_boost_queue ( )  [inline],
[override], [virtual]
```

Return the queue that is associated to the context.

Implements trisycl::detail::context.

Definition at line 63 of file opencl_context.hpp.

```
00063                                                    {
00064     return q;
00065   }
```

References q.

12.48.3.4 get_devices()

```
vector_class< trisycl::device > trisycl::detail::opencl_context::get_devices ( ) const  [inline],
[override], [virtual]
```

Returns the set of devices that are part of this context.

Todo To be implemented

Implements trisycl::detail::context.

Definition at line 103 of file opencl_context.hpp.

```
00103                                                  {
00104     vector_class<trisycl::device> devices;
00105
00106     for (const auto &d :  c.get_devices())
00107       devices.push_back(d);
00108     return devices;
00109   }
```

References c.

12.48.3.5 get_platform()

```
trisycl::platform trisycl::detail::opencl_context::get_platform ( ) const  [inline], [override],
[virtual]
```

Return the platform of the context.

Return synchronous errors via the SYCL exception class.

Todo To be implemented

Implements trisycl::detail::context.

Definition at line 94 of file opencl_context.hpp.

```
00094                                              {
00095     return trisycl::platform(c.get_devices()[0].platform());
00096   }
```

References c.

12.48.3.6 get_reference_count()

```
trisycl::cl_uint trisycl::detail::opencl_context::get_reference_count ( ) const  [inline],
[override], [virtual]
```

Returns the reference count of underlying CL context, or 0 on host.

Implements trisycl::detail::context.

Definition at line 111 of file opencl_context.hpp.

```
00111                                                  {
00112     return c.get_info<CL_CONTEXT_REFERENCE_COUNT>();
00113   }
```

References c.

12.48.3.7 instance()

```
static std::shared_ptr< opencl_context > trisycl::detail::opencl_context::instance (
            const boost::compute::context & c )  [inline], [static]
```

Get a singleton instance of the opencl_context.

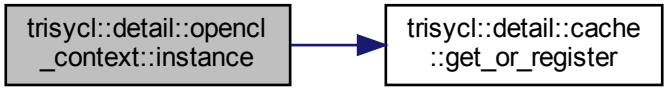
Definition at line 117 of file opencl_context.hpp.

```
00117                                                {
00118     return cache.get_or_register(c.get(),
00119                                  [&] { return new opencl_context { c }; });
00120   }
```

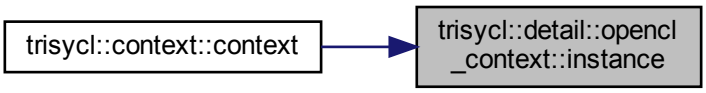
References c, and trisycl::detail::cache< Key, Value >::get_or_register().

Referenced by trisycl::context::context().

Here is the call graph for this function:



Here is the caller graph for this function:



12.48.3.8 is_host()

```
bool trisycl::detail::opencl_context::is_host ( ) const  [inline], [override], [virtual]
```

Return false because the context is not a SYCL host context.

Implements trisycl::detail::context.

Definition at line 69 of file opencl_context.hpp.

```
00069                                  {
00070     return false;
00071   }
```

12.48.4 Member Data Documentation

12.48.4.1 c

```
boost::compute::context trisycl::detail::opencl_context::c  [private]
```

User the Boost Compute abstraction of the OpenCL context.

Definition at line 31 of file opencl_context.hpp.

Referenced by get(), get_boost_compute(), get_devices(), get_platform(), get_reference_count(), and instance().

12.48.4.2 cache

```
detail::cache<cl_context, detail::opencl_context> trisycl::detail::opencl_context::cache
[static], [private]
```

A cache to always return the same alive context for a given OpenCL context.

C++11 guaranties the static construction is thread-safe

Definition at line 44 of file opencl_context.hpp.

12.48.4.3 q

```
boost::compute::command_queue trisycl::detail::opencl_context::q  [private]
```

A boost `command_queue` associated to an OpenCL context for when we need to transfer data but no queue is given (eg.

When an buffer accessor is created)

Definition at line 37 of file opencl_context.hpp.

Referenced by get_boost_queue().

The documentation for this class was generated from the following file:

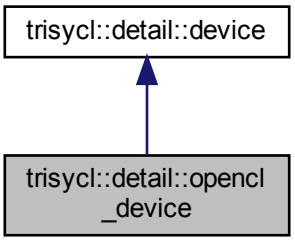
- include/triSYCL/context/detail/opencl_context.hpp

12.49 trisycl::detail::opencl_device Class Reference

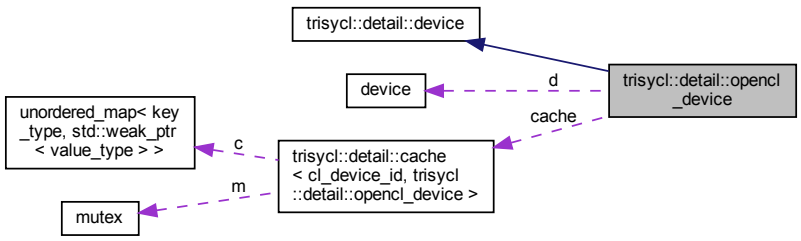
SYCL OpenCL device.

`#include <opencl_device.hpp>`

Inheritance diagram for trisycl::detail::opencl_device:



Collaboration diagram for trisycl::detail::opencl_device:



Public Member Functions

- cl_device_id get () const override

  Return the cl_device_id of the underlying OpenCL device.
- boost::compute::device & get_boost_compute () override

  Return the underlying Boost.Compute device.
- bool is_host () const override

  Return false since an OpenCL device is not the SYCL host device.
- bool is_cpu () const override

  Test if the OpenCL is a CPU device.
- bool is_gpu () const override

  Test if the OpenCL is a GPU device.
- bool is_accelerator () const override

  Test if the OpenCL is an accelerator device.

- trisycl::platform get_platform () const override
  Return the platform of device.
- std::any get_info (info::device param) const override
  Query the device for OpenCL info::device info.
- bool has_extension (const string_class &extension) const override
  Specify whether a specific extension is supported on the device.
- ~opencl_device () override
  Unregister from the cache on destruction.

Static Public Member Functions

- static std::shared_ptr< opencl_device > instance (const boost::compute::device &d)

Private Member Functions

- template<info::device param>
  std::any get_info () const
- opencl_device (const boost::compute::device &d)
  Only the instance factory can build it.

Private Attributes

- boost::compute::device d
  Use the Boost Compute abstraction of the OpenCL device.

Static Private Attributes

- static detail::cache< cl_device_id, detail::opencl_device > cache
  A cache to always return the same alive device for a given OpenCL device.

12.49.1 Detailed Description

SYCL OpenCL device.

Definition at line 28 of file opencl_device.hpp.

12.49.2 Constructor & Destructor Documentation

12.49.2.1 opencl_device()

```
trisycl::detail::opencl_device::opencl_device (
            const boost::compute::device & d )  [inline], [private]
```

Only the instance factory can build it.

Definition at line 114 of file opencl_device.hpp.

```
00114 :  d { d } {}
```

12.49.2.2 ~opencl_device()

```
trisycl::detail::opencl_device::~opencl_device ( )  [inline], [override]
```

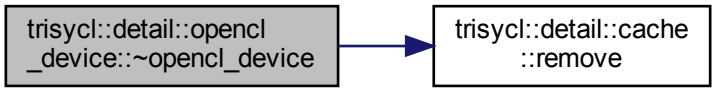
Unregister from the cache on destruction.

Definition at line 119 of file opencl_device.hpp.

```
00119                                {
00120     cache.remove(d.id());
00121   }
```

References trisycl::detail::cache< Key, Value >::remove().

Here is the call graph for this function:



12.49.3 Member Function Documentation

12.49.3.1 get()

```
cl_device_id trisycl::detail::opencl_device::get ( ) const  [inline], [override], [virtual]
```

Return the cl_device_id of the underlying OpenCL device.

Implements trisycl::detail::device.

Definition at line 43 of file opencl_device.hpp.

```
00043                                      {
00044     return d.id();
00045   }
```

References d.

12.49.3.2 get_boost_compute()

```
boost::compute::device & trisycl::detail::opencl_device::get_boost_compute ( )  [inline],
[override], [virtual]
```

Return the underlying Boost.Compute device.

Implements trisycl::detail::device.

Definition at line 49 of file opencl_device.hpp.

```
00049                                                {
00050     return d;
00051   }
```

References d.

12.49.3.3 get_info() [1/2]

```
template<info::device param>
std::any trisycl::detail::opencl_device::get_info ( ) const  [inline], [private]
```

12.49.3.4 get_info() [2/2]

```
std::any trisycl::detail::opencl_device::get_info (
            info::device param ) const  [inline], [override], [virtual]
```

Query the device for OpenCL info::device info.

Return synchronous errors via the SYCL exception class.

Implements trisycl::detail::device.

Definition at line 12 of file opencl_device_tail.hpp.

```
00012                                                    {
00013   switch (param) {
00014 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(
00015   device_type,
00016   d.get_info<CL_DEVICE_TYPE>() == CL_DEVICE_TYPE_CPU
00017   ?  info::device_type::cpu :  info::device_type::gpu);
00018 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(vendor_id, VENDOR_ID)
00019 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_compute_units, MAX_COMPUTE_UNITS )
00020 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_work_item_dimensions,
00021                                        MAX_WORK_ITEM_DIMENSIONS)
00022 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(
00023   max_work_item_sizes,
00024   (::trisycl::id<3> { d.get_info<CL_DEVICE_MAX_WORK_ITEM_SIZES>()[0]
00025                     , d.get_info<CL_DEVICE_MAX_WORK_ITEM_SIZES>()[1]
00026                     , d.get_info<CL_DEVICE_MAX_WORK_ITEM_SIZES>()[2] }))
00027 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_work_group_size, MAX_WORK_GROUP_SIZE)
00028
00029 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_vector_width_char,
00030                                        PREFERRED_VECTOR_WIDTH_CHAR)
00031 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_vector_width_short,
00032                                        PREFERRED_VECTOR_WIDTH_SHORT)
00033 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_vector_width_int,
00034                                        PREFERRED_VECTOR_WIDTH_INT)
00035 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_vector_width_long,
00036                                        PREFERRED_VECTOR_WIDTH_LONG)
00037 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_vector_width_float,
00038                                        PREFERRED_VECTOR_WIDTH_FLOAT)
00039 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_vector_width_double,
00040                                        PREFERRED_VECTOR_WIDTH_DOUBLE)
00041 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(preferred_vector_width_half,
00042                                     ::trisycl::cl_uint { 0 })
00043
00044 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(native_vector_width_char,
00045                                        NATIVE_VECTOR_WIDTH_CHAR)
00046 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(native_vector_width_short,
00047                                        NATIVE_VECTOR_WIDTH_SHORT)
00048 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(native_vector_width_int,
00049                                        NATIVE_VECTOR_WIDTH_INT)
00050 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(native_vector_width_long,
00051                                        NATIVE_VECTOR_WIDTH_LONG)
00052 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(native_vector_width_float,
00053                                        NATIVE_VECTOR_WIDTH_FLOAT)
00054 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(native_vector_width_double,
00055                                        NATIVE_VECTOR_WIDTH_DOUBLE)
00056 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(native_vector_width_half,
00057                                     ::trisycl::cl_uint { 0 })
00058
00059 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_clock_frequency, MAX_CLOCK_FREQUENCY)
00060 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(address_bits, ADDRESS_BITS)
00061 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_mem_alloc_size, MAX_MEM_ALLOC_SIZE)
00062 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(image_support, IMAGE_SUPPORT )
00063 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_read_image_args, MAX_READ_IMAGE_ARGS)
00064 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_write_image_args,
00065                                        MAX_WRITE_IMAGE_ARGS)
00066 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(image2d_max_width, IMAGE2D_MAX_WIDTH)
00067 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(image2d_max_height, IMAGE2D_MAX_HEIGHT)
```

```
00068 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(image3d_max_width, IMAGE3D_MAX_WIDTH)
00069 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(image3d_max_height, IMAGE3D_MAX_HEIGHT)
00070 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(image3d_max_depth, IMAGE3D_MAX_DEPTH)
00071 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(image_max_buffer_size, size_t { 0 })
00072 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(image_max_array_size, size_t { 0 })
00073 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_samplers, MAX_SAMPLERS)
00074 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_parameter_size, MAX_PARAMETER_SIZE)
00075 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(mem_base_addr_align, MEM_BASE_ADDR_ALIGN)
00076 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(half_fp_config,
00077                                     vector_class<info::fp_config>{} )
00078 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(single_fp_config,
00079                                     vector_class<info::fp_config>{} )
00080 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(double_fp_config,
00081                                     vector_class<info::fp_config>{} )
00082 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(global_mem_cache_type,
00083                                     info::global_mem_cache_type::none )
00084 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(
00085   global_mem_cache_line_size,
00086   ::trisycl::cl_uint { static_cast<::trisycl::cl_uint>
00087       (d.get_info<CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE>()) })
00088 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(global_mem_cache_size,
00089                                        GLOBAL_MEM_CACHE_SIZE)
00090 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(global_mem_size, GLOBAL_MEM_SIZE)
00091 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(max_constant_buffer_size,
00092                                     ::trisycl::cl_ulong { 0 } )
00093 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(max_constant_args,
00094                                     ::trisycl::cl_uint { 0 } )
00095 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(local_mem_type, info::local_mem_type::local)
00096 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(local_mem_size, LOCAL_MEM_SIZE)
00097 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(error_correction_support,
00098                                        ERROR_CORRECTION_SUPPORT)
00099 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(host_unified_memory,
00100                                        HOST_UNIFIED_MEMORY)
00101 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(profiling_timer_resolution,
00102                                        PROFILING_TIMER_RESOLUTION)
00103 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(is_endian_little, ENDIAN_LITTLE)
00104 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(is_available, AVAILABLE)
00105 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(is_compiler_available,
00106                                        COMPILER_AVAILABLE)
00107 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(is_linker_available, false)
00108 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(execution_capabilities,
00109                                     vector_class<info::execution_capability>{} )
00110 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(queue_profiling, false)
00111 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(built_in_kernels,
00112                                     vector_class<string_class>{} )
00113 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(platform, ::trisycl::platform{} )
00114 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(name, NAME)
00115 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(vendor, VENDOR)
00116 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(driver_version,
00117                                     d.get_info<CL_DRIVER_VERSION>())
00118 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(profile, PROFILE)
00119 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(version, VERSION)
00120 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(opencl_c_version, OPENCL_C_VERSION )
00121 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(extensions, vector_class<string_class>{} )
00122 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(printf_buffer_size, PRINTF_BUFFER_SIZE)
00123 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_interop_user_sync,
00124                                        PREFERRED_INTEROP_USER_SYNC)
00125 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(parent_device, ::trisycl::device{})
00126 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(partition_max_sub_devices,
00127                                     ::trisycl::cl_uint { 0 })
00128 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(partition_properties,
00129                                     vector_class<info::partition_property>{})
00130 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(
00131   partition_affinity_domains,
00132   vector_class<info::partition_affinity_domain>{})
00133 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(partition_type_property,
00134                                     info::partition_property::no_partition)
00135 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(
00136   partition_type_affinity_domain,
00137   info::partition_affinity_domain::not_applicable)
00138 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(reference_count, REFERENCE_COUNT)
00139   default:
00140     return 0;
00141   }
00142 }
```

References trisycl::info::cpu, d, trisycl::info::gpu, trisycl::info::local, trisycl::info::no_partition, trisycl::info::none, trisycl::info::not_applicable, TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE, and TRISYCL_DEFINE_DEVICE_INFO_TEMPLAT

12.49.3.5 get_platform()

trisycl::platform trisycl::detail::opencl_device::get_platform ( ) const [inline], [override],

[virtual]

Return the platform of device.

Return synchronous errors via the SYCL exception class.

Implements trisycl::detail::device.

Definition at line 85 of file opencl_device.hpp.

```
00085                                                {
00086     return d.platform();
00087   }
```

References d.

12.49.3.6 has_extension()

```
bool trisycl::detail::opencl_device::has_extension (
            const string_class & extension ) const  [inline], [override], [virtual]
```

Specify whether a specific extension is supported on the device.

Implements trisycl::detail::device.

Definition at line 97 of file opencl_device.hpp.

```
00097                                                                {
00098     return d.supports_extension(extension);
00099   }
```

References d.

12.49.3.7 instance()

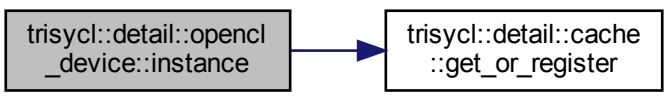
```
static std::shared_ptr< opencl_device > trisycl::detail::opencl_device::instance (
            const boost::compute::device & d )  [inline], [static]
```

Definition at line 104 of file opencl_device.hpp.

```
00104                                          {
00105     return cache.get_or_register(d.id(),
00106                                  [&] { return new opencl_device { d }; });
00107   }
```

References d, and trisycl::detail::cache< Key, Value >::get_or_register().

Here is the call graph for this function:



12.49.3.8 is_accelerator()

```
bool trisycl::detail::opencl_device::is_accelerator ( ) const  [inline], [override], [virtual]
```

Test if the OpenCL is an accelerator device.

Implements trisycl::detail::device.

Definition at line 75 of file opencl_device.hpp.

```
00075                                        {
00076     // Even in Boost.Compute the type is a bit-field, so use & instead of ==
00077     return d.type() & boost::compute::device::accelerator;
00078   }
```

References d.

12.49.3.9 is_cpu()

```
bool trisycl::detail::opencl_device::is_cpu ( ) const  [inline], [override], [virtual]
```

Test if the OpenCL is a CPU device.

Implements trisycl::detail::device.

Definition at line 61 of file opencl_device.hpp.

```
00061                                    {
00062     // Even in Boost.Compute the type is a bit-field, so use & instead of ==
00063     return d.type() & boost::compute::device::cpu;
00064   }
```

References d.

12.49.3.10 is_gpu()

```
bool trisycl::detail::opencl_device::is_gpu ( ) const  [inline], [override], [virtual]
```

Test if the OpenCL is a GPU device.

Implements trisycl::detail::device.

Definition at line 68 of file opencl_device.hpp.

```
00068                                    {
00069     // Even in Boost.Compute the type is a bit-field, so use & instead of ==
00070     return d.type() & boost::compute::device::gpu;
00071   }
```

References d.

12.49.3.11 is_host()

```
bool trisycl::detail::opencl_device::is_host ( ) const  [inline], [override], [virtual]
```

Return false since an OpenCL device is not the SYCL host device.

Implements trisycl::detail::device.

Definition at line 55 of file opencl_device.hpp.

```
00055                                {
00056     return false;
00057   }
```

12.49.4 Member Data Documentation

12.49.4.1 cache

```
detail::cache<cl_device_id, detail::opencl_device> trisycl::detail::opencl_device::cache
[static], [private]
```

A cache to always return the same alive device for a given OpenCL device.

C++11 guaranties the static construction is thread-safe

Definition at line 38 of file opencl_device.hpp.

12.49.4.2 d

```
boost::compute::device trisycl::detail::opencl_device::d  [private]
```

Use the Boost Compute abstraction of the OpenCL device.

Definition at line 31 of file opencl_device.hpp.

Referenced by get(), get_boost_compute(), get_info(), get_platform(), has_extension(), instance(), is_accelerator(), is_cpu(), and is_gpu().

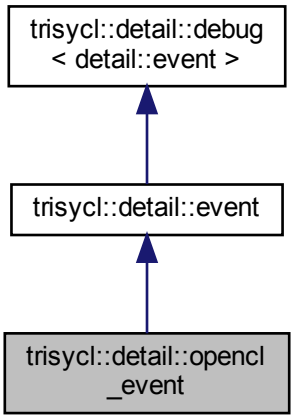
The documentation for this class was generated from the following files:

- include/triSYCL/device/detail/opencl_device.hpp
- include/triSYCL/device/detail/opencl_device_tail.hpp

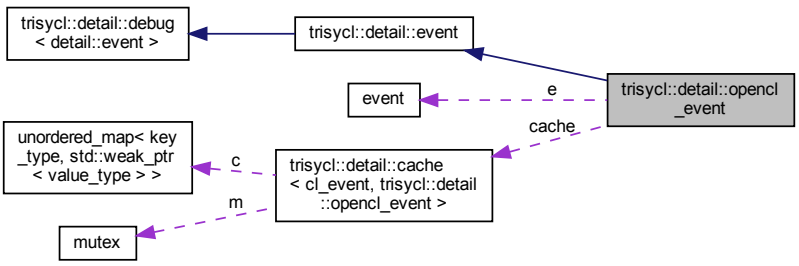
12.50 trisycl::detail::opencl_event Class Reference

`#include <opencl_event.hpp>`

Inheritance diagram for trisycl::detail::opencl_event:



Collaboration diagram for trisycl::detail::opencl_event:



Public Member Functions

- cl_event get () const override

  Return the cl_event of the underlying OpenCL event.
- const boost::compute::event & get_boost_compute () const override

  Return the underlying Boost.Compute event, if any.
- bool is_host () const override
- info::event_command_status get_command_execution_status () const override
- cl_ulong get_profiling_info (info::event_profiling param) const override
- trisycl::cl_uint get_reference_count () const override
- void wait () const override
- ~opencl_event () override

Static Public Member Functions

- static std::shared_ptr< opencl_event > instance (const boost::compute::event &e)

  Get a singleton instance of the `opencl_event`.

Private Member Functions

- opencl_event (const boost::compute::event &e)

  Only the instance factory can build it.

Private Attributes

- boost::compute::event e

Static Private Attributes

- static detail::cache< cl_event, detail::opencl_event > cache

  A cache to always return the same alive event for a given OpenCL event.

12.50.1 Detailed Description

Definition at line 15 of file opencl_event.hpp.

12.50.2 Constructor & Destructor Documentation

12.50.2.1 opencl_event()

```
trisycl::detail::opencl_event::opencl_event (
            const boost::compute::event & e )  [inline], [private]
```

Only the instance factory can build it.

Definition at line 61 of file opencl_event.hpp.

```
00061                                          :
00062     e { e } {}
```

12.50.2.2 ~opencl_event()

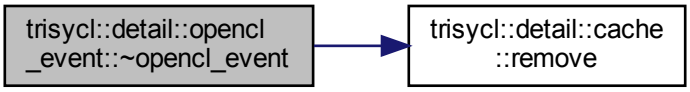
```
trisycl::detail::opencl_event::~opencl_event ( )  [inline], [override]
```

Definition at line 65 of file opencl_event.hpp.

```
00065                             {
00066     cache.remove(e.get());
00067   }
```

References trisycl::detail::cache< Key, Value >::remove().

Here is the call graph for this function:



12.50.3 Member Function Documentation

12.50.3.1 get()

```
cl_event trisycl::detail::opencl_event::get ( ) const  [inline], [override], [virtual]
```

Return the cl_event of the underlying OpenCL event.

Implements trisycl::detail::event.

Definition at line 25 of file opencl_event.hpp.

```
00025                                 {
00026     return e.get();
00027   }
```

References e.

12.50.3.2 get_boost_compute()

```
const boost::compute::event & trisycl::detail::opencl_event::get_boost_compute ( ) const  [inline],
[override], [virtual]
```

Return the underlying Boost.Compute event, if any.

Implements trisycl::detail::event.

Definition at line 29 of file opencl_event.hpp.

```
00029                                                       {
00030     return e;
00031   }
```

References e.

12.50.3.3 get_command_execution_status()

```
info::event_command_status trisycl::detail::opencl_event::get_command_execution_status ( )
const  [inline], [override], [virtual]
```

Implements trisycl::detail::event.

Definition at line 37 of file opencl_event.hpp.

```
00037                                                                    {
00038     return static_cast<info::event_command_status>(e.status());
00039   }
```

References e.

12.50.3.4 get_profiling_info()

```
cl_ulong trisycl::detail::opencl_event::get_profiling_info (
            info::event_profiling param ) const  [inline], [override], [virtual]
```

Implements trisycl::detail::event.

Definition at line 41 of file opencl_event.hpp.

```
00041                                                                                {
00042     return e.get_profiling_info<cl_ulong>(static_cast<cl_profiling_info>(param));
00043   }
```

References e.

12.50.3.5 get_reference_count()

```
trisycl::cl_uint trisycl::detail::opencl_event::get_reference_count ( ) const  [inline], [override],
[virtual]
```

Implements trisycl::detail::event.

Definition at line 45 of file opencl_event.hpp.

```
00045                                                {
00046     return e.get_info<CL_EVENT_REFERENCE_COUNT>();
00047   }
```

References e.

12.50.3.6 instance()

```
static std::shared_ptr< opencl_event > trisycl::detail::opencl_event::instance (
            const boost::compute::event & e )  [inline], [static]
```

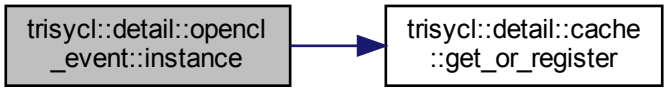
Get a singleton instance of the opencl_event.

Definition at line 55 of file opencl_event.hpp.

```
00055                                        {
00056     return cache.get_or_register(e.get(),
00057                                  [&] { return new opencl_event { e }; });
00058   }
```

References e, and trisycl::detail::cache< Key, Value >::get_or_register().

Here is the call graph for this function:



12.50.3.7 is_host()

```
bool trisycl::detail::opencl_event::is_host ( ) const  [inline], [override], [virtual]
```

Implements trisycl::detail::event.

Definition at line 33 of file opencl_event.hpp.

```
00033                                  {
00034     return false;
00035   }
```

12.50.3.8 wait()

```
void trisycl::detail::opencl_event::wait ( ) const  [inline], [override], [virtual]
```

Implements trisycl::detail::event.

Definition at line 49 of file opencl_event.hpp.

```
00049                                {
00050     e.wait();
00051   }
```

References e.

12.50.4 Member Data Documentation

12.50.4.1 cache

`detail::cache<cl_event, detail::opencl_event> trisycl::detail::opencl_event::cache [static], [private]`

A cache to always return the same alive event for a given OpenCL event.

C++11 guaranties the static construction is thread-safe

Definition at line 23 of file opencl_event.hpp.

12.50.4.2 e

`boost::compute::event trisycl::detail::opencl_event::e [private]`

Definition at line 16 of file opencl_event.hpp.

Referenced by get(), get_boost_compute(), get_command_execution_status(), get_profiling_info(), get_reference_count(), instance(), and wait().

The documentation for this class was generated from the following file:

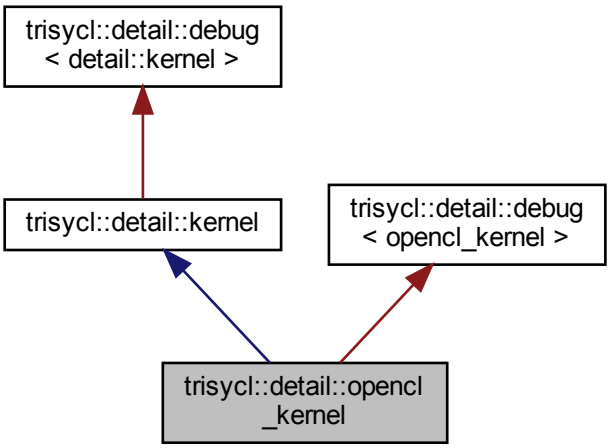
- include/triSYCL/event/detail/opencl_event.hpp

12.51 trisycl::detail::opencl_kernel Class Reference

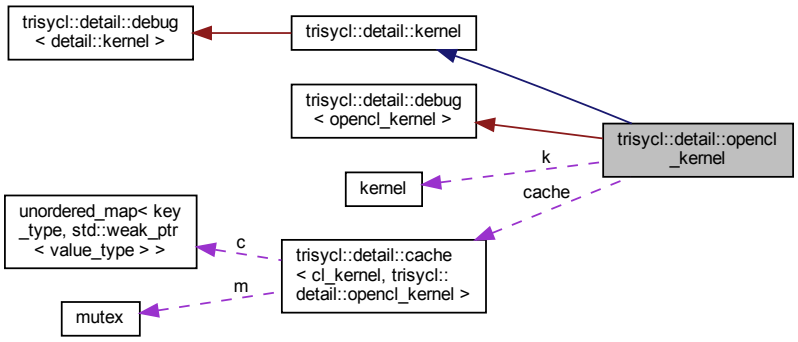
An abstraction of the OpenCL kernel.

`#include <opencl_kernel.hpp>`

Inheritance diagram for trisycl::detail::opencl_kernel:



Collaboration diagram for trisycl::detail::opencl_kernel:



Public Member Functions

- cl_kernel get () const override

  Return the underlying OpenCL object.
- boost::compute::kernel & get_boost_compute () override

  Return the Boost.Compute OpenCL kernel object for this kernel.
- void single_task (std::shared_ptr< detail::task > task, std::shared_ptr< detail::queue > q) override

  Launch a single task of the OpenCL kernel.
- TRISYCL_ParallelForKernel_RANGE (1) TRISYCL_ParallelForKernel_RANGE(2) TRISYCL_ParallelFor↩ Kernel_RANGE(3) ~opencl_kernel() override

  Unregister from the cache on destruction.

Static Public Member Functions

- static std::shared_ptr< opencl_kernel > instance (const boost::compute::kernel &k)

Private Member Functions

- opencl_kernel (const boost::compute::kernel &k)

Private Attributes

- boost::compute::kernel k

  Use the Boost Compute abstraction of the OpenCL kernel.

Static Private Attributes

- static detail::cache< cl_kernel, detail::opencl_kernel > cache

  A cache to always return the same alive kernel for a given OpenCL kernel.

Additional Inherited Members

12.51.1 Detailed Description

An abstraction of the OpenCL kernel.

Definition at line 27 of file opencl_kernel.hpp.

12.51.2 Constructor & Destructor Documentation

12.51.2.1 opencl_kernel()

```
trisycl::detail::opencl_kernel::opencl_kernel (
            const boost::compute::kernel & k )  [inline], [private]
```

Definition at line 40 of file opencl_kernel.hpp.

```
00040 :  k { k } {}
```

12.51.3 Member Function Documentation

12.51.3.1 get()

```
cl_kernel trisycl::detail::opencl_kernel::get ( ) const  [inline], [override], [virtual]
```

Return the underlying OpenCL object.

Todo Improve the spec to deprecate C OpenCL host API and move to C++ instead to avoid this ugly ownership management

Todo Test error and throw. Externalize this feature in Boost.Compute?

Implements trisycl::detail::kernel.

Definition at line 56 of file opencl_kernel.hpp.

```
00056                                {
00057 /// \todo Test error and throw.  Externalize this feature in Boost.Compute?
00058     clRetainKernel(k);
00059     return k.get();
00060   }
```

12.51.3.2 get_boost_compute()

```
boost::compute::kernel & trisycl::detail::opencl_kernel::get_boost_compute ( )  [inline],
[override], [virtual]
```

Return the Boost.Compute OpenCL kernel object for this kernel.

This is an extension.

Implements trisycl::detail::kernel.

Definition at line 67 of file opencl_kernel.hpp.

```
00067                                                {
00068     return k;
00069   }
```

12.51.3.3 instance()

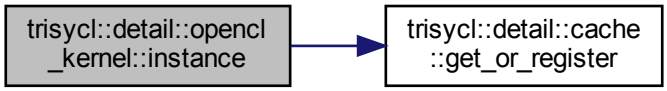
```
static std::shared_ptr< opencl_kernel > trisycl::detail::opencl_kernel::instance (
            const boost::compute::kernel & k )  [inline], [static]
```

Definition at line 46 of file opencl_kernel.hpp.

```
00046                                              {
00047     return cache.get_or_register(k.get(),
00048                                  [&] { return new opencl_kernel { k }; });
00049   }
```

References trisycl::detail::cache< Key, Value >::get_or_register(), and k.

Here is the call graph for this function:



12.51.3.4 single_task()

```
void trisycl::detail::opencl_kernel::single_task (
            std::shared_ptr< detail::task > task,
            std::shared_ptr< detail::queue > q )  [inline], [override], [virtual]
```

Launch a single task of the OpenCL kernel.

Todo Remove either task or q

Implements trisycl::detail::kernel.

Definition at line 89 of file opencl_kernel.hpp.

```
00090                                                    {
00091     q->get_boost_compute().enqueue_task(k);
00092     /* For now use a crude synchronization mechanism to map directly a
00093 host task to an accelerator task */
00094     q->get_boost_compute().finish();
00095   }
```

12.51.3.5 TRISYCL_ParallelForKernel_RANGE()

```
trisycl::detail::opencl_kernel::TRISYCL_ParallelForKernel_RANGE (
            1  )  [inline], [override]
```

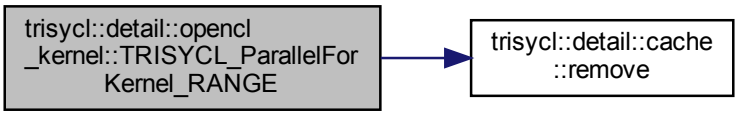
Unregister from the cache on destruction.

Definition at line 123 of file opencl_kernel.hpp.

```
00131                                {
00132     cache.remove(k.get());
```

References trisycl::detail::cache< Key, Value >::remove().

Here is the call graph for this function:



12.51.4 Member Data Documentation

12.51.4.1 cache

```
detail::cache<cl_kernel, detail::opencl_kernel> trisycl::detail::opencl_kernel::cache  [static],
[private]
```

A cache to always return the same alive kernel for a given OpenCL kernel.

C++11 guaranties the static construction is thread-safe

Definition at line 38 of file opencl_kernel.hpp.

12.51.4.2 k

```
boost::compute::kernel trisycl::detail::opencl_kernel::k  [private]
```

Use the Boost Compute abstraction of the OpenCL kernel.

Definition at line 31 of file opencl_kernel.hpp.

Referenced by instance().

The documentation for this class was generated from the following file:

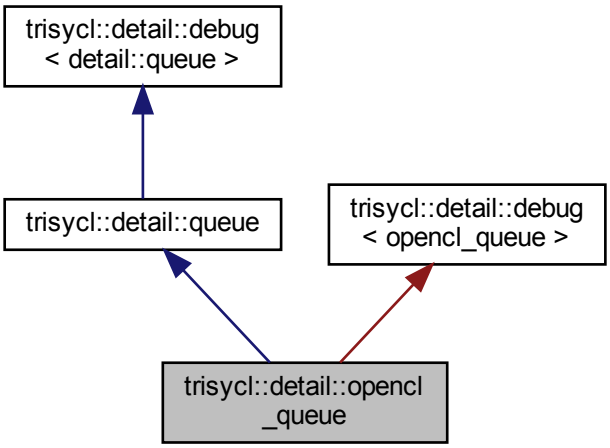
- include/triSYCL/kernel/detail/opencl_kernel.hpp

12.52 trisycl::detail::opencl_queue Class Reference

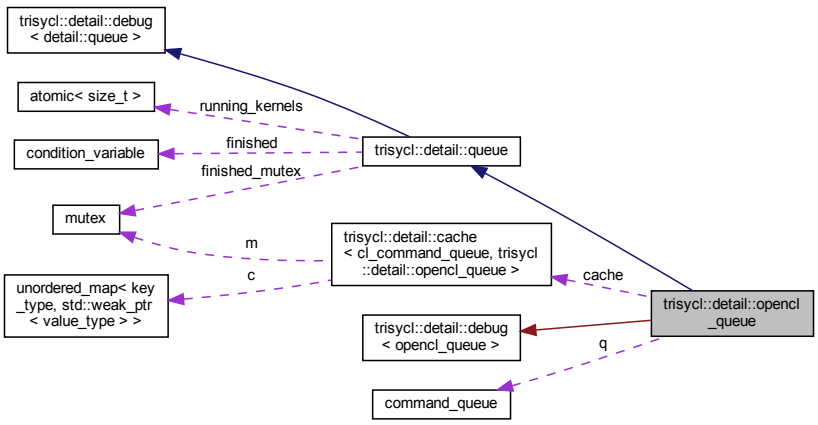
Some implementation details about the SYCL queue.

`#include <opencl_queue.hpp>`

Inheritance diagram for trisycl::detail::opencl_queue:



Collaboration diagram for trisycl::detail::opencl_queue:



Public Member Functions

- ~opencl_queue () override

  Unregister from the cache on destruction.

Static Public Member Functions

- static std::shared_ptr< opencl_queue > instance (const boost::compute::command_queue &q)

  Get a singleton instance of the opencl_queue.
- static std::shared_ptr< detail::queue > instance (const trisycl::device &d)

  Create a new queue associated to this device.

Private Member Functions

- cl_command_queue get () const override

  Return the cl_command_queue of the underlying OpenCL queue.
- boost::compute::command_queue & get_boost_compute () override

  Return the underlying Boost.Compute command queue.
- trisycl::context get_context () const override

  Return the SYCL context associated to the queue.
- trisycl::device get_device () const override

  Return the SYCL device associated to the queue.
- bool is_host () const override

  Claim proudly that an OpenCL queue cannot be the SYCL host queue.
- opencl_queue (const boost::compute::command_queue &q)

  Only the instance factory can built it.

Private Attributes

- boost::compute::command_queue q

  Use the Boost Compute abstraction of the OpenCL command queue.

Static Private Attributes

- static detail::cache< cl_command_queue, detail::opencl_queue > cache

  A cache to always return the same alive queue for a given OpenCL command queue.

Additional Inherited Members

12.52.1 Detailed Description

Some implementation details about the SYCL queue.

Definition at line 21 of file opencl_queue.hpp.

12.52.2 Constructor & Destructor Documentation

12.52.2.1 opencl_queue()

```
trisycl::detail::opencl_queue::opencl_queue (
            const boost::compute::command_queue & q )  [inline], [private]
```

Only the instance factory can built it.

Definition at line 65 of file opencl_queue.hpp.

```
00065 :  q { q } {}
```

12.52.2.2 ~opencl_queue()

```
trisycl::detail::opencl_queue::~opencl_queue ( )  [inline], [override]
```

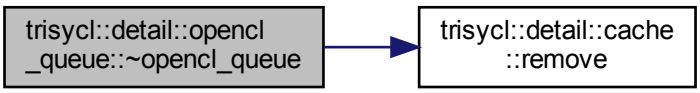
Unregister from the cache on destruction.

Definition at line 94 of file opencl_queue.hpp.

```
00094                                {
00095     cache.remove(q.get());
00096   }
```

References trisycl::detail::cache< Key, Value >::remove().

Here is the call graph for this function:



12.52.3 Member Function Documentation

12.52.3.1 get()

```
cl_command_queue trisycl::detail::opencl_queue::get ( ) const  [inline], [override], [private],
[virtual]
```

Return the cl_command_queue of the underlying OpenCL queue.

Implements trisycl::detail::queue.

Definition at line 34 of file opencl_queue.hpp.

```
00034                                        {
00035     return q.get();
00036   }
```

References q.

12.52.3.2 get_boost_compute()

```
boost::compute::command_queue & trisycl::detail::opencl_queue::get_boost_compute ( )  [inline],
[override], [private], [virtual]
```

Return the underlying Boost.Compute command queue.

Implements trisycl::detail::queue.

Definition at line 40 of file opencl_queue.hpp.

```
00040                                                    {
00041     return q;
00042   }
```

References q.

12.52.3.3 get_context()

```
trisycl::context trisycl::detail::opencl_queue::get_context ( ) const  [inline], [override],
[private], [virtual]
```

Return the SYCL context associated to the queue.

Implements trisycl::detail::queue.

Definition at line 46 of file opencl_queue.hpp.

```
00046                                              {
00047     return q.get_context();
00048   }
```

References q.

12.52.3.4 get_device()

```
trisycl::device trisycl::detail::opencl_queue::get_device ( ) const  [inline], [override],
[private], [virtual]
```

Return the SYCL device associated to the queue.

Implements trisycl::detail::queue.

Definition at line 52 of file opencl_queue.hpp.

```
00052                                            {
00053     return q.get_device();
00054   }
```

References q.

12.52.3.5 instance() [1/2]

```
static std::shared_ptr< opencl_queue > trisycl::detail::opencl_queue::instance (
            const boost::compute::command_queue & q )  [inline], [static]
```

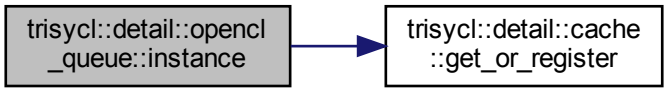
Get a singleton instance of the opencl_queue.

Definition at line 71 of file opencl_queue.hpp.

```
00071                                                {
00072     return cache.get_or_register(q.get(),
00073                                  [&] { return new opencl_queue { q }; });
00074   }
```

References trisycl::detail::cache< Key, Value >::get_or_register(), and q.

Here is the call graph for this function:



12.52.3.6 instance() [2/2]

```
static std::shared_ptr< detail::queue > trisycl::detail::opencl_queue::instance (
            const trisycl::device & d )  [inline], [static]
```

Create a new queue associated to this device.

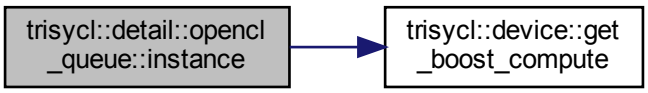
Todo Check with SYCL committee what is the expected behaviour here about the context. Is this a new context everytime, or always the same for a given device?

Definition at line 84 of file opencl_queue.hpp.

```
00084                                                {
00085     return instance (boost::compute::command_queue {
00086         // For now, create a new context every time
00087         boost::compute::context { d.get_boost_compute() },
00088         d.get_boost_compute()
00089           });
00090   }
```

References trisycl::device::get_boost_compute().

Here is the call graph for this function:



12.52.3.7 is_host()

```
bool trisycl::detail::opencl_queue::is_host ( ) const  [inline], [override], [private], [virtual]
```

Claim proudly that an OpenCL queue cannot be the SYCL host queue.

Implements trisycl::detail::queue.

Definition at line 58 of file opencl_queue.hpp.

```
00058                                {
00059     return false;
00060   }
```

12.52.4 Member Data Documentation

12.52.4.1 cache

```
detail::cache<cl_command_queue, detail::opencl_queue> trisycl::detail::opencl_queue::cache
[static], [private]
```

A cache to always return the same alive queue for a given OpenCL command queue.

C++11 guaranties the static construction is thread-safe

Definition at line 31 of file opencl_queue.hpp.

12.52.4.2 q

```
boost::compute::command_queue trisycl::detail::opencl_queue::q  [private]
```

Use the Boost Compute abstraction of the OpenCL command queue.

Definition at line 24 of file opencl_queue.hpp.

Referenced by get(), get_boost_compute(), get_context(), get_device(), and instance().

The documentation for this class was generated from the following file:

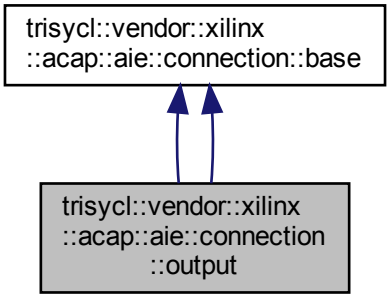
- include/triSYCL/queue/detail/opencl_queue.hpp

12.53 trisycl::vendor::xilinx::acap::aie::connection::output Struct Reference

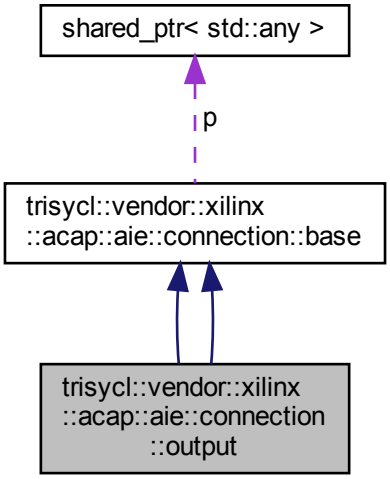
Abstract an output port.

`#include <connection.hpp>`

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::connection::output:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::connection::output:



Public Member Functions

- template<typename OutputT , access::target Target>
  auto out ()

  Get the output port.
- template<typename OutputT , access::target Target>
  auto out ()

  Get the output port.
- base (const connection &c)

  Recycle the base constructors.
- base ()=default

  Recycle the base constructors.
- base (const connection &c)

  Recycle the base constructors.
- base ()=default

  Recycle the base constructors.
- base (const connection &c)

  Recycle the base constructors.
- base ()=default

  Recycle the base constructors.
- base (const connection &c)

  Recycle the base constructors.
- base ()=default

  Recycle the base constructors.

Additional Inherited Members

12.53.1 Detailed Description

Abstract an output port.

Definition at line 345 of file connection.hpp.

12.53.2 Member Function Documentation

12.53.2.1 base() [1/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base ( )  [default]
```

Recycle the base constructors.

12.53.2.2 base() [2/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base ( )  [default]
```

Recycle the base constructors.

12.53.2.3 base() [3/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base ( )  [default]
```

Recycle the base constructors.

12.53.2.4 base() [4/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base ( )  [default]
```

Recycle the base constructors.

12.53.2.5 base() [5/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base (
            const connection & c )  [inline]
```

Recycle the base constructors.

Definition at line 292 of file connection.hpp.

```
00293       : p { c.p } {}
```

12.53.2.6 base() [6/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base (
            const connection & c )  [inline]
```

Recycle the base constructors.

Definition at line 292 of file connection.hpp.

```
00293       : p { c.p } {}
```

12.53.2.7 base() [7/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base (
            const connection & c )  [inline]
```

Recycle the base constructors.

Definition at line 292 of file connection.hpp.

```
00293       :  p { c.p } {}
```

12.53.2.8 base() [8/8]

```
trisycl::vendor::xilinx::acap::aie::connection::base::base (
            const connection & c )  [inline]
```

Recycle the base constructors.

Definition at line 292 of file connection.hpp.

```
00293       :  p { c.p } {}
```

12.53.2.9 out() [1/2]

```
template<typename OutputT , access::target Target>
auto trisycl::vendor::xilinx::acap::aie::connection::output::out ( )  [inline]
```

Get the output port.

Parameters

in	*InputT*	is the data type to be used in the transfers
in	*Target*	specifies if the connection is blocking or not

Exceptions

trisycl::runtime_error	if the output is not connected

Definition at line 359 of file connection.hpp.

```
00359               {
00360       if (!p)
00361         throw ::trisycl::runtime_error { "This output is not connected" };
00362       return
00363         pipe_of<OutputT>().template get_access<access::mode::write, Target>();
00364     }
```

References trisycl::vendor::xilinx::acap::aie::connection::base::p.

12.53.2.10 out() [2/2]

```
template<typename OutputT , access::target Target>
auto trisycl::vendor::xilinx::acap::aie::connection::output::out ( )  [inline]
```

Get the output port.

Parameters

in	*InputT*	is the data type to be used in the transfers
in	*Target*	specifies if the connection is blocking or not

Exceptions

trisycl::runtime_error	if the output is not connected

Definition at line 359 of file connection.hpp.

```
00359                {
00360       if (!p)
00361         throw ::trisycl::runtime_error { "This output is not connected" };
00362       return
00363         pipe_of<OutputT>().template get_access<access::mode::write, Target>();
00364     }
```

References trisycl::vendor::xilinx::acap::aie::connection::base::p.

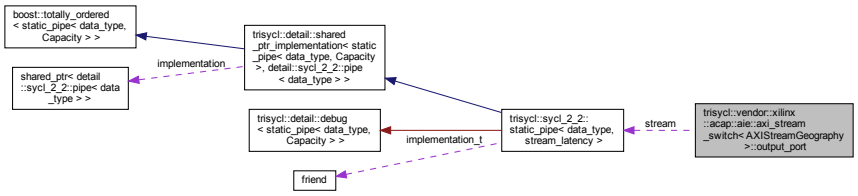
The documentation for this struct was generated from the following files:

- include/sycl/vendor/Xilinx/acap/aie/connection.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/connection.hpp

12.54 trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::output_port Struct Reference

```
#include <axi_stream_switch.hpp>
```

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::output_↩ port:



Public Attributes

- ::trisycl::sycl_2_2::static_pipe< data_type, stream_latency > stream

12.54.1 Detailed Description

template<typename AXIStreamGeography>
struct trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::output_port

Definition at line 342 of file axi_stream_switch.hpp.

12.54.2 Member Data Documentation

12.54.2.1 stream

```
template<typename AXIStreamGeography >
trisycl::sycl_2_2::static_pipe< data_type, stream_latency > trisycl::vendor::xilinx::acap::aie::axi_stream_sw
AXIStreamGeography >::output_port::stream
```

Definition at line 343 of file axi_stream_switch.hpp.

The documentation for this struct was generated from the following files:

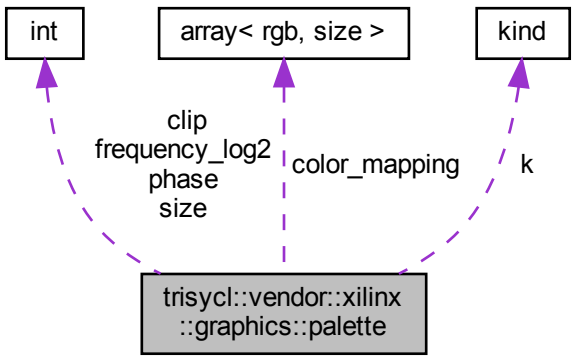
- include/sycl/vendor/Xilinx/acap/aie/axi_stream_switch.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/axi_stream_switch.hpp

12.55 trisycl::vendor::xilinx::graphics::palette Struct Reference

Color palette to project a linear data space to an RGB color space.

```
#include <graphics.hpp>
```

Collaboration diagram for trisycl::vendor::xilinx::graphics::palette:



Public Types

- enum kind { gray , rainbow , gray , rainbow }
- enum kind { gray , rainbow , gray , rainbow }

Public Member Functions

- palette (kind k=gray, int phase=0, int frequency_log2=0, int clip=-1)

  Create a palette.
- void gray_palette ()
- void rainbow_palette ()
- void update ()
- void set (kind new_k, int new_phase, int new_frequency_log2, int new_clip)

  Set palette parameters and update.
- rgb palettize (double data, double min_value, double max_value)

  Transform a value into an RGB color according to the palette.
- void decrease_clip ()
- void increase_clip ()
- void decrease_frequency ()
- void increase_frequency ()
- void decrease_phase ()
- void increase_phase ()
- bool handle_key_press_event (GdkEventKey &event)

  Handle with any key pressed.
- palette (kind k=gray, int phase=0, int frequency_log2=0, int clip=-1)

  Create a palette.
- void gray_palette ()
- void rainbow_palette ()
- void update ()
- void set (kind new_k, int new_phase, int new_frequency_log2, int new_clip)

  Set palette parameters and update.
- rgb palettize (double data, double min_value, double max_value)

  Transform a value into an RGB color according to the palette.
- void decrease_clip ()
- void increase_clip ()
- void decrease_frequency ()
- void increase_frequency ()
- void decrease_phase ()
- void increase_phase ()
- bool handle_key_press_event (GdkEventKey &event)

  Handle with any key pressed.

Public Attributes

- std::array< rgb, size > color_mapping
- kind k
- int phase
- int clip
- int frequency_log2

Static Public Attributes

- static constexpr int size = 256

12.55.1 Detailed Description

Color palette to project a linear data space to an RGB color space.

Reimplement the palette concept by Nicolas Paris in PompC & HyperC back from the 90's... :-)

Definition at line 146 of file graphics.hpp.

12.55.2 Member Enumeration Documentation

12.55.2.1 kind [1/2]

```
enum trisycl::vendor::xilinx::graphics::palette::kind
```

Enumerator

gray	
rainbow	
gray	
rainbow	

Definition at line 150 of file graphics.hpp.

```
00150 { gray, rainbow };
```

12.55.2.2 kind [2/2]

```
enum trisycl::vendor::xilinx::graphics::palette::kind
```

Enumerator

gray	
rainbow	
gray	
rainbow	

Definition at line 150 of file graphics.hpp.

```
00150 { gray, rainbow };
```

12.55.3 Constructor & Destructor Documentation

12.55.3.1 palette() [1/2]

```
trisycl::vendor::xilinx::graphics::palette::palette (
            kind k = gray,
            int phase = 0,
            int frequency_log2 = 0,
            int clip = -1 )  [inline]
```

Create a palette.

Parameters

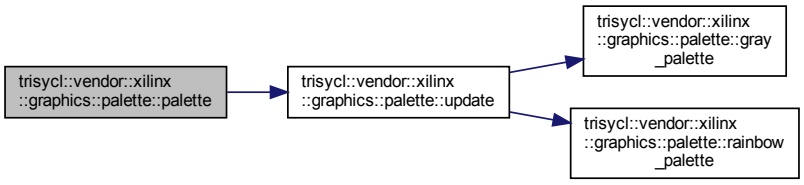
in	*k*	is the kind of palette
in	*phase*	is optional and allows the palette to have a phase shift instead of starting at 0
in	*clip*	is optional and specifies a value to be enhanced

Definition at line 165 of file graphics.hpp.

```
00166     :  k { k }, phase { phase } , clip { clip }, frequency_log2{frequency_log2} {
00167       update();
00168     }
```

References update().

Here is the call graph for this function:



12.55.3.2 palette() [2/2]

```
trisycl::vendor::xilinx::graphics::palette::palette (
            kind k = gray,
            int phase = 0,
            int frequency_log2 = 0,
            int clip = -1 )  [inline]
```

Create a palette.

Parameters

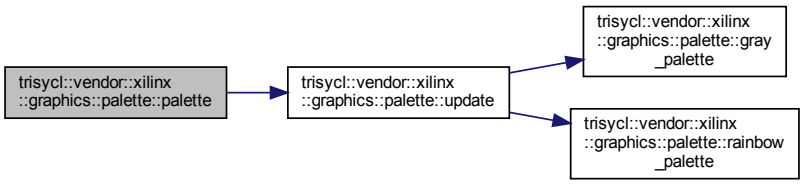
in	*k*	is the kind of palette
in	*phase*	is optional and allows the palette to have a phase shift instead of starting at 0
in	*clip*	is optional and specifies a value to be enhanced

Definition at line 165 of file graphics.hpp.

```
00166     :  k { k }, phase { phase } , clip { clip }, frequency_log2{frequency_log2} {
00167       update();
00168     }
```

References update().

Here is the call graph for this function:



12.55.4 Member Function Documentation

12.55.4.1 decrease_clip() [1/2]

```
void trisycl::vendor::xilinx::graphics::palette::decrease_clip ( )  [inline]
```

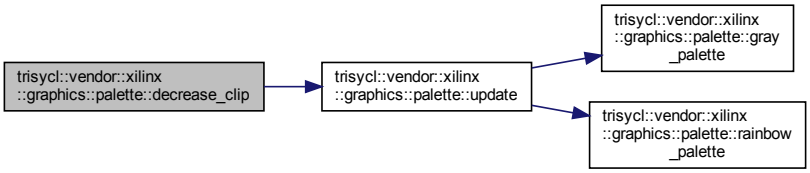
Definition at line 272 of file graphics.hpp.

```
00272                          {
00273     --clip;
00274     update();
00275   }
```

References clip, and update().

Referenced by handle_key_press_event().

Here is the call graph for this function:



Here is the caller graph for this function:



12.55.4.2 decrease_clip() [2/2]

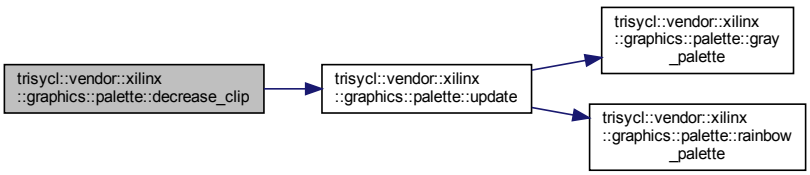
```
void trisycl::vendor::xilinx::graphics::palette::decrease_clip ( )  [inline]
```

Definition at line 272 of file graphics.hpp.

```
00272                                {
00273     --clip;
00274     update();
00275   }
```

References clip, and update().

Here is the call graph for this function:



12.55.4.3 decrease_frequency() [1/2]

```
void trisycl::vendor::xilinx::graphics::palette::decrease_frequency ( )  [inline]
```

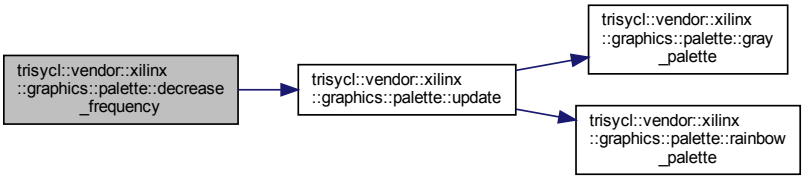
Definition at line 284 of file graphics.hpp.

```
00284                                    {
00285     --frequency_log2;
00286     update();
00287   }
```

References frequency_log2, and update().

Referenced by handle_key_press_event().

Here is the call graph for this function:



Here is the caller graph for this function:



12.55.4.4 decrease_frequency() [2/2]

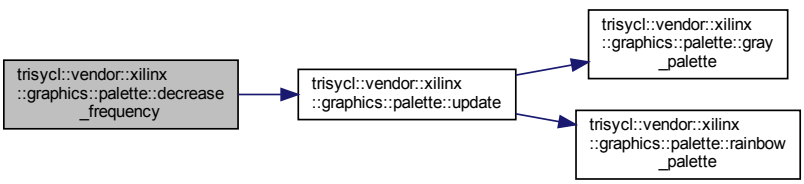
```
void trisycl::vendor::xilinx::graphics::palette::decrease_frequency ( )  [inline]
```

Definition at line 284 of file graphics.hpp.

```
00284                                {
00285     --frequency_log2;
00286     update();
00287   }
```

References frequency_log2, and update().

Here is the call graph for this function:



12.55.4.5 decrease_phase() [1/2]

```
void trisycl::vendor::xilinx::graphics::palette::decrease_phase ( )  [inline]
```

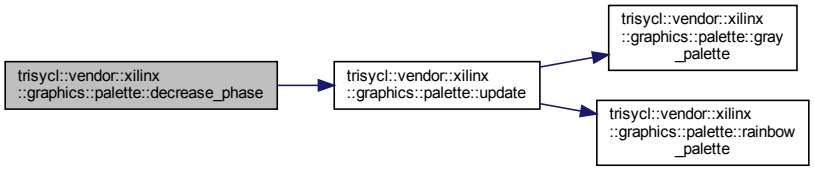
Definition at line 296 of file graphics.hpp.

```
00296                           {
00297     --phase;
00298     update();
00299   }
```

References phase, and update().

Referenced by handle_key_press_event().

Here is the call graph for this function:



Here is the caller graph for this function:



12.55.4.6 decrease_phase() [2/2]

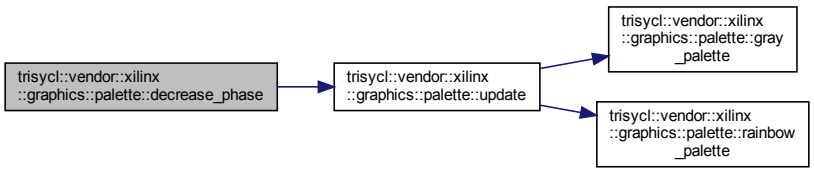
```
void trisycl::vendor::xilinx::graphics::palette::decrease_phase ( )  [inline]
```

Definition at line 296 of file graphics.hpp.

```
00296                           {
00297     --phase;
00298     update();
00299   }
```

References phase, and update().

Here is the call graph for this function:



12.55.4.7 gray_palette() [1/2]

```
void trisycl::vendor::xilinx::graphics::palette::gray_palette ( )  [inline]
```

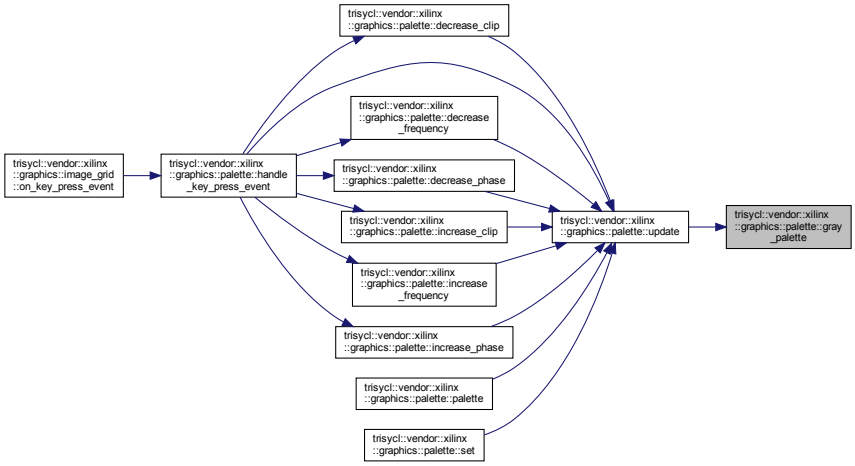
Definition at line 172 of file graphics.hpp.

```
00172                          {
00173     for (int i = 0; auto &e :  color_mapping) {
00174       if (i == clip)
00175         // Put a full red color for the selected value
00176         e = { 255, 0, 0 };
00177       else {
00178         int j;
00179         // There is no bidirectional shift operator in C++
00180         if (frequency_log2 < 0)
00181           // Divide the step by 2^{-fl}
00182           j = i » -frequency_log2;
00183         else
00184           // Multiply the step by 2^{fl}
00185           j = i « frequency_log2;
00186         auto v = static_cast<std::uint8_t>(j + phase);
00187         // A non-saturated "gray" of luminosity v
00188         e = { v,  v,  v };
00189       }
00190       ++i;
00191     }
00192   }
```

References clip, color_mapping, frequency_log2, i, and phase.

Referenced by update().

Here is the caller graph for this function:



12.55.4.8 gray_palette() [2/2]

```
void trisycl::vendor::xilinx::graphics::palette::gray_palette ( )  [inline]
```

Definition at line 172 of file graphics.hpp.

```
00172                                {
00173     for (int i = 0; auto &e :  color_mapping) {
00174       if (i == clip)
00175         // Put a full red color for the selected value
00176         e = { 255, 0, 0 };
00177       else {
00178         int j;
00179         // There is no bidirectional shift operator in C++
00180         if (frequency_log2 < 0)
00181           // Divide the step by 2^{-fl}
00182           j = i » -frequency_log2;
00183         else
00184           // Multiply the step by 2^{fl}
00185           j = i « frequency_log2;
00186         auto v = static_cast<std::uint8_t>(j + phase);
00187         // A non-saturated "gray" of luminosity v
00188         e = { v,  v,  v };
00189       }
00190       ++i;
00191     }
00192   }
```

References clip, color_mapping, frequency_log2, i, and phase.

12.55.4.9 handle_key_press_event() [1/2]

```
bool trisycl::vendor::xilinx::graphics::palette::handle_key_press_event (
            GdkEventKey & event )  [inline]
```

Handle with any key pressed.

Definition at line 309 of file graphics.hpp.

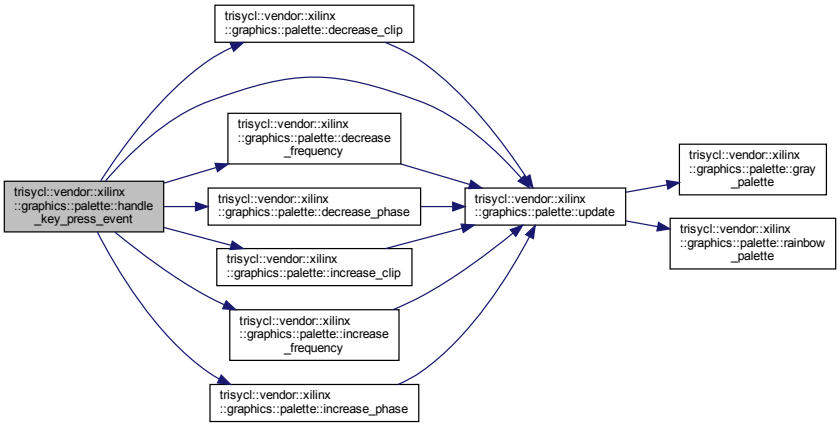
```
00309                                                {
00310     /* Only listen to keys without modifiers (no control, alt...)  but
00311 still allows shift */
00312     if (!(event.state
00313           & gtk_accelerator_get_default_mod_mask()
00314           & ~GDK_SHIFT_MASK)) {
00315       switch (event.keyval) {
00316       case GDK_KEY_0:
00317       case GDK_KEY_KP_0:
00318         k = gray;
00319         update();
00320         break;
00321       case GDK_KEY_1:
00322       case GDK_KEY_KP_1:
00323         k = rainbow;
00324         update();
00325         break;
00326       case GDK_KEY_minus:
00327       case GDK_KEY_KP_Subtract:
00328       case GDK_KEY_underscore:
00329         decrease_frequency();
00330         break;
00331       case GDK_KEY_plus:
00332       case GDK_KEY_KP_Add:
00333       case GDK_KEY_equal:
00334       case GDK_KEY_KP_Equal:
00335         increase_frequency();
00336         break;
00337       case GDK_KEY_h:
00338         increase_phase();
00339         break;
00340       case GDK_KEY_H:
```

```
00341         increase_phase();
00342         decrease_clip();
00343         break;
00344       case GDK_KEY_j:
00345         decrease_clip();
00346         break;
00347       case GDK_KEY_k:
00348         increase_clip();
00349         break;
00350       case GDK_KEY_l:
00351         decrease_phase();
00352         break;
00353       case GDK_KEY_L:
00354         decrease_phase();
00355         increase_clip();
00356         break;
00357       default:
00358         // Not a key press for us:  let the other listeners to handle
00359         return false;
00360       }
00361       // We have handled the event:  do not go on with any other listener
00362       return true;
00363     }
00364     // Not the right modifiers:  let the other listeners to handle
00365     return false;
00366   }
```

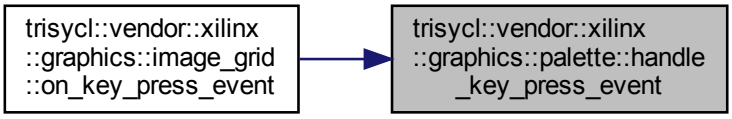
References decrease_clip(), decrease_frequency(), decrease_phase(), gray, increase_clip(), increase_frequency(), increase_phase(), k, rainbow, and update().

Referenced by trisycl::vendor::xilinx::graphics::image_grid::on_key_press_event().

Here is the call graph for this function:



Here is the caller graph for this function:



12.55.4.10 handle_key_press_event() [2/2]

```
bool trisycl::vendor::xilinx::graphics::palette::handle_key_press_event (
            GdkEventKey & event )  [inline]
```

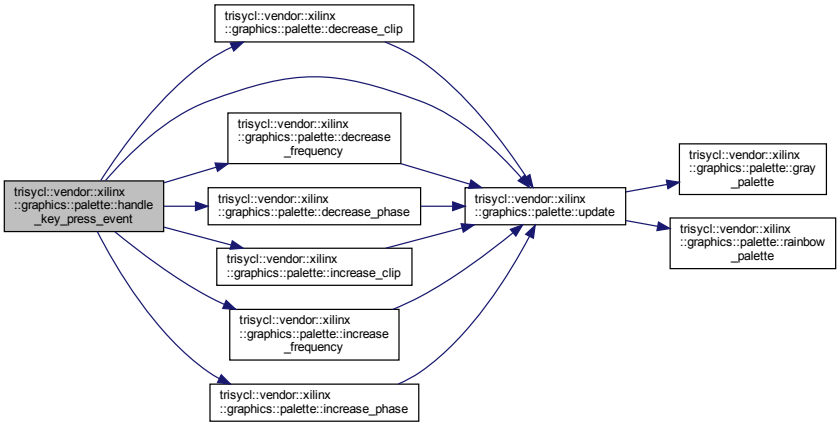
Handle with any key pressed.

Definition at line 309 of file graphics.hpp.

```
00309                                                {
00310     /* Only listen to keys without modifiers (no control, alt...)  but
00311 still allows shift */
00312     if (!(event.state
00313           & gtk_accelerator_get_default_mod_mask()
00314           & ~GDK_SHIFT_MASK)) {
00315       switch (event.keyval) {
00316       case GDK_KEY_0:
00317       case GDK_KEY_KP_0:
00318         k = gray;
00319         update();
00320         break;
00321       case GDK_KEY_1:
00322       case GDK_KEY_KP_1:
00323         k = rainbow;
00324         update();
00325         break;
00326       case GDK_KEY_minus:
00327       case GDK_KEY_KP_Subtract:
00328       case GDK_KEY_underscore:
00329         decrease_frequency();
00330         break;
00331       case GDK_KEY_plus:
00332       case GDK_KEY_KP_Add:
00333       case GDK_KEY_equal:
00334       case GDK_KEY_KP_Equal:
00335         increase_frequency();
00336         break;
00337       case GDK_KEY_h:
00338         increase_phase();
00339         break;
00340       case GDK_KEY_H:
00341         increase_phase();
00342         decrease_clip();
00343         break;
00344       case GDK_KEY_j:
00345         decrease_clip();
00346         break;
00347       case GDK_KEY_k:
00348         increase_clip();
00349         break;
00350       case GDK_KEY_l:
00351         decrease_phase();
00352         break;
00353       case GDK_KEY_L:
00354         decrease_phase();
00355         increase_clip();
00356         break;
00357       default:
00358         // Not a key press for us:  let the other listeners to handle
00359         return false;
00360       }
00361       // We have handled the event:  do not go on with any other listener
00362       return true;
00363     }
00364     // Not the right modifiers:  let the other listeners to handle
00365     return false;
00366   }
```

References decrease_clip(), decrease_frequency(), decrease_phase(), gray, increase_clip(), increase_frequency(), increase_phase(), k, rainbow, and update().

Here is the call graph for this function:



12.55.4.11 increase_clip() [1/2]

```
void trisycl::vendor::xilinx::graphics::palette::increase_clip ( )  [inline]
```

Definition at line 278 of file graphics.hpp.

```
00278                                {
00279     ++clip;
00280     update();
00281   }
```

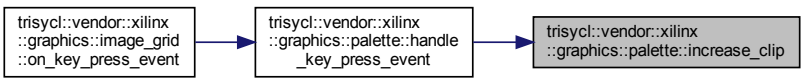
References clip, and update().

Referenced by handle_key_press_event().

Here is the call graph for this function:



Here is the caller graph for this function:



12.55.4.12 increase_clip() [2/2]

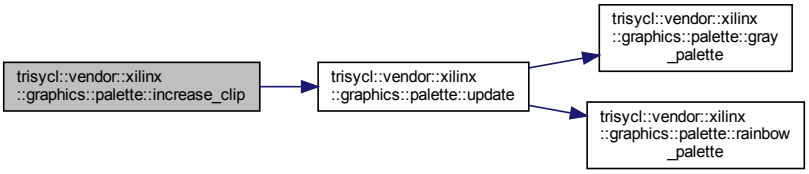
```
void trisycl::vendor::xilinx::graphics::palette::increase_clip ( )  [inline]
```

Definition at line 278 of file graphics.hpp.

```
00278                           {
00279     ++clip;
00280     update();
00281   }
```

References clip, and update().

Here is the call graph for this function:



12.55.4.13 increase_frequency() [1/2]

```
void trisycl::vendor::xilinx::graphics::palette::increase_frequency ( )  [inline]
```

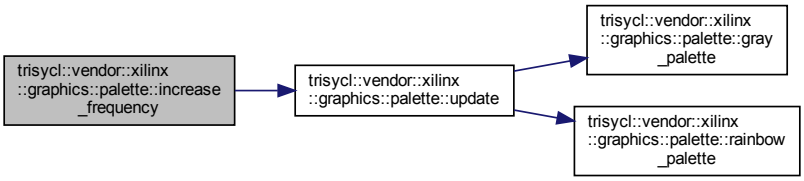
Definition at line 290 of file graphics.hpp.

```
00290                                {
00291     ++frequency_log2;
00292     update();
00293   }
```

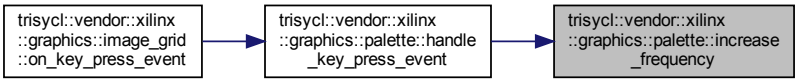
References frequency_log2, and update().

Referenced by handle_key_press_event().

Here is the call graph for this function:



Here is the caller graph for this function:



12.55.4.14 increase_frequency() [2/2]

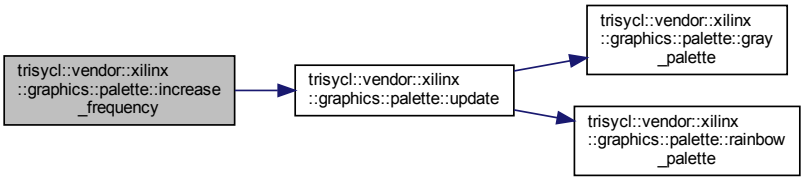
```
void trisycl::vendor::xilinx::graphics::palette::increase_frequency ( )  [inline]
```

Definition at line 290 of file graphics.hpp.

```
00290                                    {
00291     ++frequency_log2;
00292     update();
00293   }
```

References frequency_log2, and update().

Here is the call graph for this function:



12.55.4.15 increase_phase() [1/2]

```
void trisycl::vendor::xilinx::graphics::palette::increase_phase ( )  [inline]
```

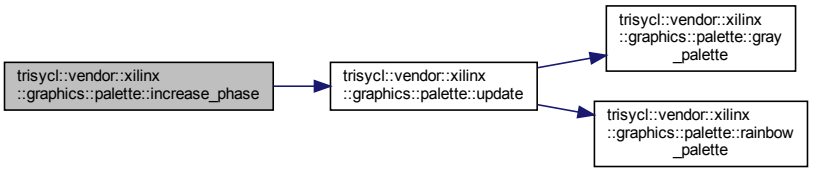
Definition at line 302 of file graphics.hpp.

```
00302                                    {
00303     ++phase;
00304     update();
00305   }
```

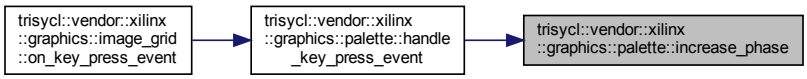
References phase, and update().

Referenced by handle_key_press_event().

Here is the call graph for this function:



Here is the caller graph for this function:



12.55.4.16 increase_phase() [2/2]

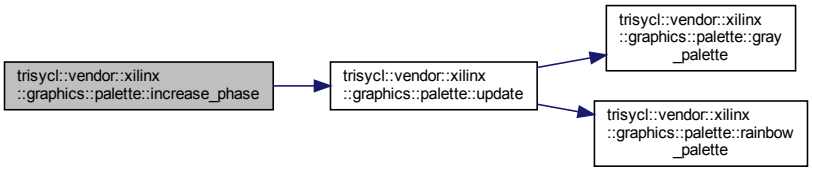
```
void trisycl::vendor::xilinx::graphics::palette::increase_phase ( )  [inline]
```

Definition at line 302 of file graphics.hpp.

```
00302                                {
00303     ++phase;
00304     update();
00305   }
```

References phase, and update().

Here is the call graph for this function:



12.55.4.17 palettize() [1/2]

```
rgb trisycl::vendor::xilinx::graphics::palette::palettize (
              double data,
              double min_value,
              double max_value )  [inline]
```

Transform a value into an RGB color according to the palette.

Definition at line 265 of file graphics.hpp.

```
00265                                                                    {
00266     std::uint8_t v = (data - min_value)*255/(max_value - min_value);
00267     // Write the same value for RGB to have a grey level
00268     return color_mapping[v];
00269   }
```

References color_mapping.

Referenced by trisycl::vendor::xilinx::graphics::image_grid::update_tile_data_image().

Here is the caller graph for this function:



12.55.4.18 palettize() [2/2]

```
rgb trisycl::vendor::xilinx::graphics::palette::palettize (
              double data,
              double min_value,
              double max_value )  [inline]
```

Transform a value into an RGB color according to the palette.

Definition at line 265 of file graphics.hpp.

```
00265                                                                    {
00266     std::uint8_t v = (data - min_value)*255/(max_value - min_value);
00267     // Write the same value for RGB to have a grey level
00268     return color_mapping[v];
00269   }
```

References color_mapping.

12.55.4.19 rainbow_palette() [1/2]

```
void trisycl::vendor::xilinx::graphics::palette::rainbow_palette ( )  [inline]
```

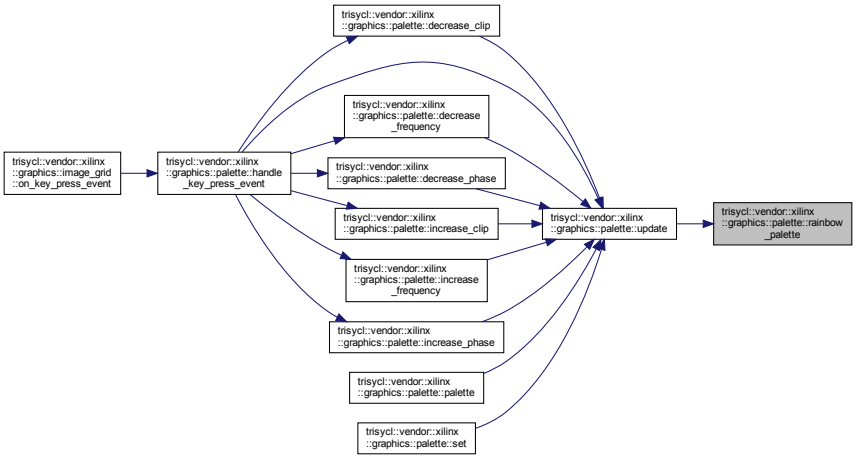
Definition at line 196 of file graphics.hpp.

```
00196                           {
00197     Gdk::RGBA color;
00198     for (int i = 0; auto &e :  color_mapping) {
00199       // Put the selected value as black
00200       double luminosity = !(i == clip);
00201       int j;
00202       // There is no bidirectional shift operator in C++
00203       if (frequency_log2 < 0)
00204         // Divide the step by 2^{-fl}
00205         j = i » -frequency_log2;
00206       else
00207         // Multiply the step by 2^{fl}
00208         j = i « frequency_log2;
00209       auto v = static_cast<std::uint8_t>(j + phase);
00210       color.set_hsv(360.*v/size, 1, luminosity);
00211       e = { static_cast<std::uint8_t>(color.get_red_u() » 8),
00212             static_cast<std::uint8_t>(color.get_green_u() » 8),
00213             static_cast<std::uint8_t>(color.get_blue_u() » 8) };
00214       ++i;
00215     }
00216   }
```

References clip, color, color_mapping, frequency_log2, i, phase, and size.

Referenced by update().

Here is the caller graph for this function:



12.55.4.20 rainbow_palette() [2/2]

```
void trisycl::vendor::xilinx::graphics::palette::rainbow_palette ( )  [inline]
```

Definition at line 196 of file graphics.hpp.

```
00196                           {
00197     Gdk::RGBA color;
00198     for (int i = 0; auto &e :  color_mapping) {
00199       // Put the selected value as black
00200       double luminosity = !(i == clip);
00201       int j;
```

```
00202       // There is no bidirectional shift operator in C++
00203       if (frequency_log2 < 0)
00204         // Divide the step by 2^{-fl}
00205         j = i » -frequency_log2;
00206       else
00207         // Multiply the step by 2^{fl}
00208         j = i « frequency_log2;
00209       auto v = static_cast<std::uint8_t>(j + phase);
00210       color.set_hsv(360.*v/size, 1, luminosity);
00211       e = { static_cast<std::uint8_t>(color.get_red_u() » 8),
00212             static_cast<std::uint8_t>(color.get_green_u() » 8),
00213             static_cast<std::uint8_t>(color.get_blue_u() » 8) };
00214       ++i;
00215     }
00216   }
```

References clip, color, color_mapping, frequency_log2, i, phase, and size.

12.55.4.21 set() [1/2]

```
void trisycl::vendor::xilinx::graphics::palette::set (
              kind new_k,
              int new_phase,
              int new_frequency_log2,
              int new_clip )  [inline]
```

Set palette parameters and update.

Parameters

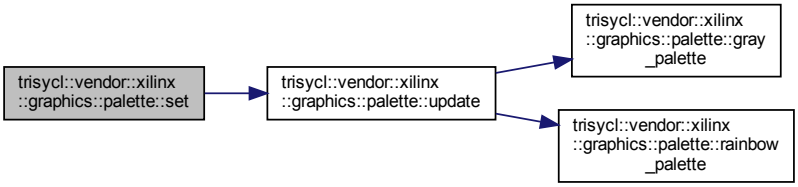
in	*new_k*	is the kind of palette
in	*new_phase*	allows the palette to have a phase shift instead of starting at 0
in	*new_clip*	specifies a value to be enhanced

Definition at line 254 of file graphics.hpp.

```
00254                                                                    {
00255     k = new_k;
00256     phase = new_phase;
00257     clip = new_clip;
00258     frequency_log2 = new_frequency_log2;
00259     update();
00260   }
```

References clip, frequency_log2, k, phase, and update().

Here is the call graph for this function:



12.55.4.22 set() [2/2]

```
void trisycl::vendor::xilinx::graphics::palette::set (
            kind new_k,
            int new_phase,
            int new_frequency_log2,
            int new_clip )  [inline]
```

Set palette parameters and update.

Parameters

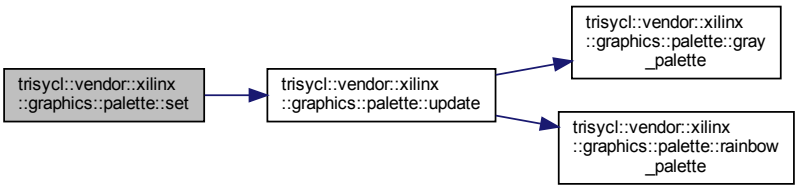
in	*new_k*	is the kind of palette
in	*new_phase*	allows the palette to have a phase shift instead of starting at 0
in	*new_clip*	specifies a value to be enhanced

Definition at line 254 of file graphics.hpp.

```
00254                                                                    {
00255     k = new_k;
00256     phase = new_phase;
00257     clip = new_clip;
00258     frequency_log2 = new_frequency_log2;
00259     update();
00260   }
```

References clip, frequency_log2, k, phase, and update().

Here is the call graph for this function:



12.55.4.23 update() [1/2]

```
void trisycl::vendor::xilinx::graphics::palette::update ( )  [inline]
```

Definition at line 221 of file graphics.hpp.

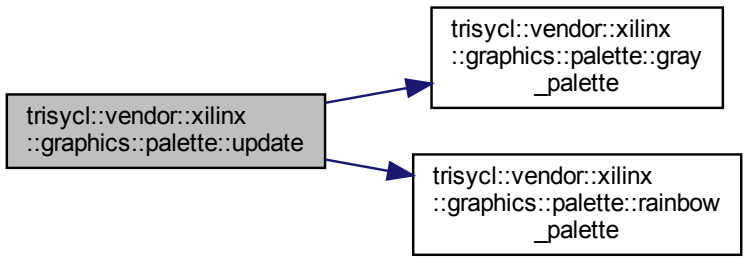
```
00221                {
00222     // Implement modulo arithmetic on interval [-1, size - 1]
00223     if (clip < -1)
00224       clip = size - 1;
00225     else if (clip >= size)
00226         clip = -1;
00227     // The frequency can only evolve in this interval
00228     frequency_log2 = std::clamp(frequency_log2, -8, 7);
00229     /* The phase can evolve modulo size.  When there are some negative
00230 numbers it is implementation dependent but that is enough for
00231 this GUI */
00232     phase = phase % size;
```

```
00233
00234     switch (k) {
00235       case gray:
00236         gray_palette();
00237         break;
00238       case rainbow:
00239         rainbow_palette();
00240         break;
00241     }
00242   }
```

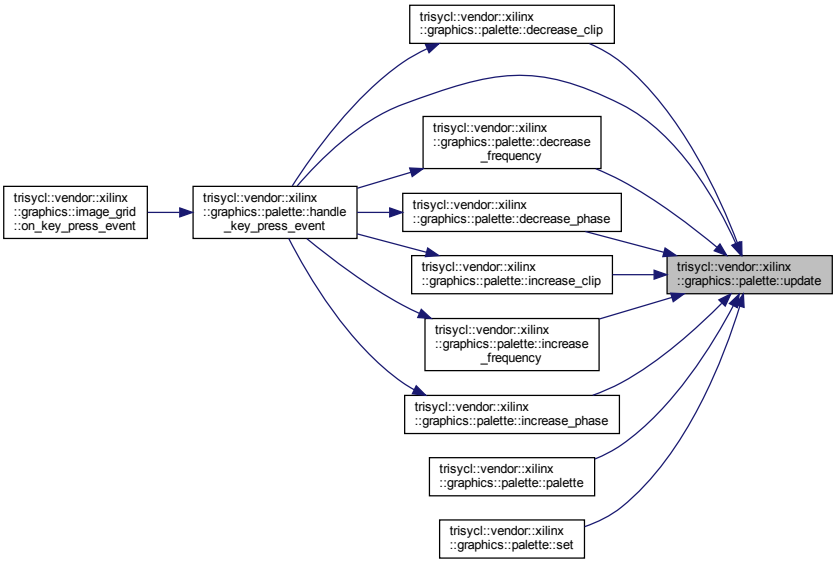
References clip, frequency_log2, gray, gray_palette(), k, phase, rainbow, rainbow_palette(), and size.

Referenced by decrease_clip(), decrease_frequency(), decrease_phase(), handle_key_press_event(), increase_clip(), increase_frequency(), increase_phase(), palette(), and set().

Here is the call graph for this function:



Here is the caller graph for this function:



12.55.4.24 update() [2/2]

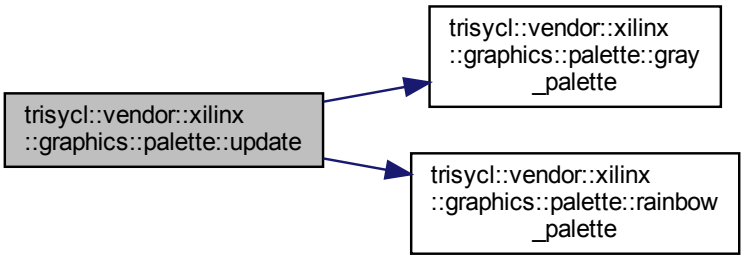
```
void trisycl::vendor::xilinx::graphics::palette::update ( )  [inline]
```

Definition at line 221 of file graphics.hpp.

```
00221                {
00222     // Implement modulo arithmetic on interval [-1, size - 1]
00223     if (clip < -1)
00224       clip = size - 1;
00225     else if (clip >= size)
00226         clip = -1;
00227     // The frequency can only evolve in this interval
00228     frequency_log2 = std::clamp(frequency_log2, -8, 7);
00229     /* The phase can evolve modulo size.  When there are some negative
00230 numbers it is implementation dependent but that is enough for
00231 this GUI */
00232     phase = phase % size;
00233
00234     switch (k) {
00235       case gray:
00236         gray_palette();
00237         break;
00238       case rainbow:
00239         rainbow_palette();
00240         break;
00241     }
00242   }
```

References clip, frequency_log2, gray, gray_palette(), k, phase, rainbow, rainbow_palette(), and size.

Here is the call graph for this function:



12.55.5 Member Data Documentation

12.55.5.1 clip

```
int trisycl::vendor::xilinx::graphics::palette::clip
```

Definition at line 153 of file graphics.hpp.

Referenced by decrease_clip(), gray_palette(), increase_clip(), rainbow_palette(), set(), and update().

12.55.5.2 color_mapping

```
std::array< rgb, size > trisycl::vendor::xilinx::graphics::palette::color_mapping
```

Definition at line 148 of file graphics.hpp.

Referenced by gray_palette(), palettize(), and rainbow_palette().

12.55.5.3 frequency_log2

```
int trisycl::vendor::xilinx::graphics::palette::frequency_log2
```

Definition at line 154 of file graphics.hpp.

Referenced by decrease_frequency(), gray_palette(), increase_frequency(), rainbow_palette(), set(), and update().

12.55.5.4 k

```
kind trisycl::vendor::xilinx::graphics::palette::k
```

Definition at line 151 of file graphics.hpp.

Referenced by handle_key_press_event(), set(), and update().

12.55.5.5 phase

```
int trisycl::vendor::xilinx::graphics::palette::phase
```

Definition at line 152 of file graphics.hpp.

Referenced by decrease_phase(), gray_palette(), increase_phase(), rainbow_palette(), set(), and update().

12.55.5.6 size

```
static constexpr int trisycl::vendor::xilinx::graphics::palette::size = 256  [static], [constexpr]
```

Definition at line 147 of file graphics.hpp.

Referenced by rainbow_palette(), and update().

The documentation for this struct was generated from the following files:

- include/sycl/vendor/Xilinx/graphics.hpp
- include/triSYCL/vendor/Xilinx/graphics.hpp

12.56 trisycl::info::param_traits< T, Param > Struct Template Reference

Implement a meta-function from (T, value) to T' to express the return type value of an OpenCL function of kind (T, value)

`#include <param_traits.hpp>`

12.56.1 Detailed Description

template<typename T, T Param>
struct trisycl::info::param_traits< T, Param >

Implement a meta-function from (T, value) to T' to express the return type value of an OpenCL function of kind (T, value)

Definition at line 18 of file param_traits.hpp.

The documentation for this struct was generated from the following file:

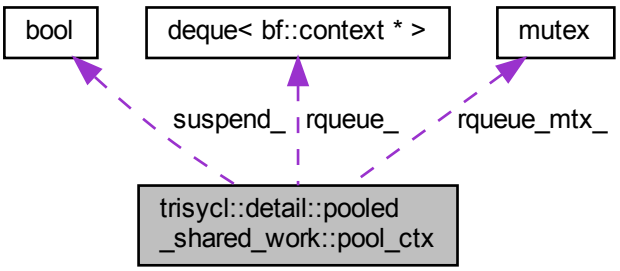
- include/triSYCL/info/param_traits.hpp

12.57 trisycl::detail::pooled_shared_work::pool_ctx Struct Reference

Shared storage among the working threads.

`#include <pooled_shared_work.hpp>`

Collaboration diagram for trisycl::detail::pooled_shared_work::pool_ctx:



Public Member Functions

- pool_ctx (bool suspend)

Public Attributes

- const bool suspend_

  Indicate if a thread without work goes to sleep instead of busy-waiting.
- rqueue_type rqueue_ {}

  The global queue storing the runnable fibers.
- std::mutex rqueue_mtx_ {}

  For concurrent access to the global queue.

12.57.1 Detailed Description

Shared storage among the working threads.

Definition at line 50 of file pooled_shared_work.hpp.

12.57.2 Constructor & Destructor Documentation

12.57.2.1 pool_ctx()

```
trisycl::detail::pooled_shared_work::pool_ctx::pool_ctx (
            bool suspend ) [inline]
```

Definition at line 51 of file pooled_shared_work.hpp.

```
00051                           : suspend_ { suspend }
00052     {}
```

12.57.3 Member Data Documentation

12.57.3.1 rqueue_

```
rqueue_type trisycl::detail::pooled_shared_work::pool_ctx::rqueue_ {}
```

The global queue storing the runnable fibers.

Definition at line 58 of file pooled_shared_work.hpp.

12.57.3.2 rqueue_mtx_

```
std::mutex trisycl::detail::pooled_shared_work::pool_ctx::rqueue_mtx_ {}
```

For concurrent access to the global queue.

Definition at line 61 of file pooled_shared_work.hpp.

12.57.3.3 suspend_

```
const bool trisycl::detail::pooled_shared_work::pool_ctx::suspend_
```

Indicate if a thread without work goes to sleep instead of busy-waiting.

Definition at line 55 of file pooled_shared_work.hpp.

The documentation for this struct was generated from the following file:

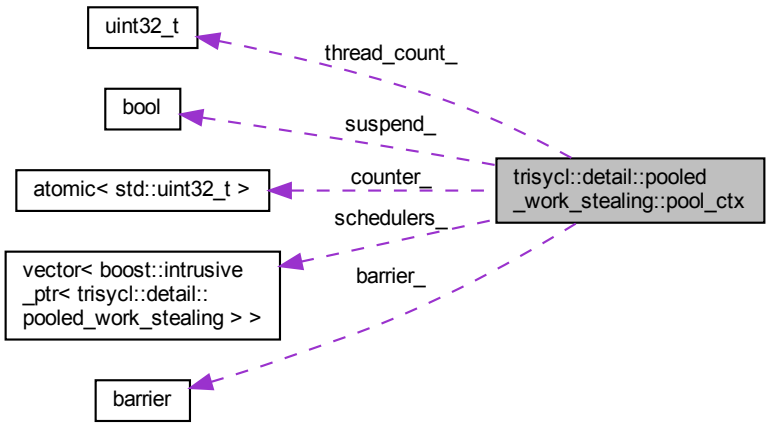
- include/triSYCL/detail/fiber/pooled_shared_work.hpp

12.58 trisycl::detail::pooled_work_stealing::pool_ctx Struct Reference

Shared storage among the working threads.

```
#include <pooled_work_stealing.hpp>
```

Collaboration diagram for trisycl::detail::pooled_work_stealing::pool_ctx:



Public Member Functions

- pool_ctx (std::uint32_t thread_count, bool suspend)

Public Attributes

- const std::uint32_t thread_count_

  Number of threads in the worker pool.
- const bool suspend_

  Indicate if a thread without work goes to sleep instead of busy-waiting.
- std::atomic< std::uint32_t > counter_ = 0

  Counter used to give a unique id_ to the worker.
- std::vector< boost::intrusive_ptr< pooled_work_stealing > > schedulers_

  Keep track of each worker scheduler.
- boost::barrier barrier_

  Synchronize all the working thread after starting and before finishing.

12.58.1 Detailed Description

Shared storage among the working threads.

Definition at line 47 of file pooled_work_stealing.hpp.

12.58.2 Constructor & Destructor Documentation

12.58.2.1 pool_ctx()

```
trisycl::detail::pooled_work_stealing::pool_ctx::pool_ctx (
            std::uint32_t thread_count,
            bool suspend )  [inline]
```

Definition at line 48 of file pooled_work_stealing.hpp.

```
00049       :  thread_count_ { thread_count }
00050       , suspend_ { suspend }
00051       , schedulers_ { thread_count, nullptr }
00052       , barrier_ { thread_count }
00053     {}
```

12.58.3 Member Data Documentation

12.58.3.1 barrier_

```
boost::barrier trisycl::detail::pooled_work_stealing::pool_ctx::barrier_
```

Synchronize all the working thread after starting and before finishing.

Definition at line 68 of file pooled_work_stealing.hpp.

12.58.3.2 counter_

```
std::atomic<std::uint32_t> trisycl::detail::pooled_work_stealing::pool_ctx::counter_ = 0
```

Counter used to give a unique id_ to the worker.

Definition at line 62 of file pooled_work_stealing.hpp.

12.58.3.3 schedulers_

```
std::vector<boost::intrusive_ptr<pooled_work_stealing> > trisycl::detail::pooled_work_↩
stealing::pool_ctx::schedulers_
```

Keep track of each worker scheduler.

Definition at line 65 of file pooled_work_stealing.hpp.

12.58.3.4 suspend_

```
const bool trisycl::detail::pooled_work_stealing::pool_ctx::suspend_
```

Indicate if a thread without work goes to sleep instead of busy-waiting.

Definition at line 59 of file pooled_work_stealing.hpp.

12.58.3.5 thread_count_

```
const std::uint32_t trisycl::detail::pooled_work_stealing::pool_ctx::thread_count_
```

Number of threads in the worker pool.

Definition at line 56 of file pooled_work_stealing.hpp.

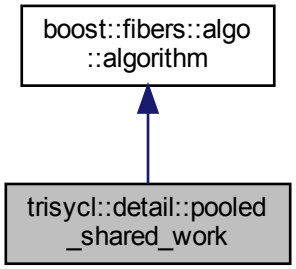
The documentation for this struct was generated from the following file:

- include/triSYCL/detail/fiber/pooled_work_stealing.hpp

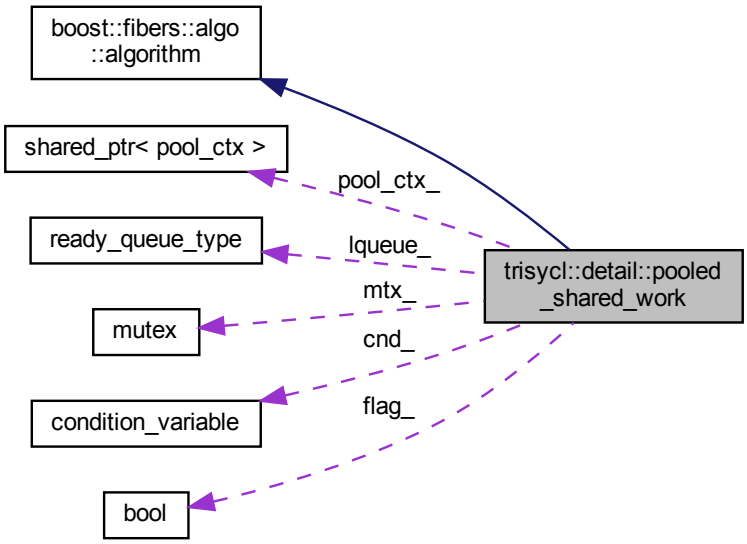
12.59 trisycl::detail::pooled_shared_work Class Reference

```
#include <pooled_shared_work.hpp>
```

Inheritance diagram for trisycl::detail::pooled_shared_work:



Collaboration diagram for trisycl::detail::pooled_shared_work:



Classes

- struct pool_ctx

  Shared storage among the working threads.

Public Types

- using ctx = std::shared_ptr< pool_ctx >

  Type tracking the common worker data.

Public Member Functions

- pooled_shared_work (const ctx &pc)
- pooled_shared_work (pooled_shared_work const &)=delete
- pooled_shared_work (pooled_shared_work &&)=delete
- pooled_shared_work & operator= (pooled_shared_work const &)=delete
- pooled_shared_work & operator= (pooled_shared_work &&)=delete
- void awakened (bf::context *ctx) noexcept override
- bf::context * pick_next () noexcept override
- bool has_ready_fibers () const noexcept override
- void suspend_until (std::chrono::steady_clock::time_point const &time_point) noexcept override
- void notify () noexcept override

Static Public Member Functions

- static ctx create_pool_ctx (bool suspend)

Private Types

- using rqueue_type = std::deque< bf::context * >
- using lqueue_type = bf::scheduler::ready_queue_type

Private Attributes

- ctx pool_ctx_

  Some shared datastructure among the working threads.
- lqueue_type lqueue_ {}

  The runnable local fibers bound to the thread.
- std::mutex mtx_ {}

  The thread-local suspend/notify mechanics.
- std::condition_variable cnd_ {}
- bool flag_ { false }

12.59.1 Detailed Description

Definition at line 42 of file pooled_shared_work.hpp.

12.59.2 Member Typedef Documentation

12.59.2.1 ctx

```
using trisycl::detail::pooled_shared_work::ctx = std::shared_ptr<pool_ctx>
```

Type tracking the common worker data.

Definition at line 65 of file pooled_shared_work.hpp.

12.59.2.2 lqueue_type

```
using trisycl::detail::pooled_shared_work::lqueue_type = bf::scheduler::ready_queue_type  [private]
```

Definition at line 45 of file pooled_shared_work.hpp.

12.59.2.3 rqueue_type

```
using trisycl::detail::pooled_shared_work::rqueue_type = std::deque<bf::context *>  [private]
```

Definition at line 44 of file pooled_shared_work.hpp.

12.59.3 Constructor & Destructor Documentation

12.59.3.1 pooled_shared_work() [1/3]

```
trisycl::detail::pooled_shared_work::pooled_shared_work (
            const ctx & pc )  [inline]
```

Definition at line 88 of file pooled_shared_work.hpp.

```
00088                                   : pool_ctx_ { pc }
00089   {}
```

12.59.3.2 pooled_shared_work() [2/3]

```
trisycl::detail::pooled_shared_work::pooled_shared_work (
            pooled_shared_work const &  )  [delete]
```

12.59.3.3 pooled_shared_work() [3/3]

```
trisycl::detail::pooled_shared_work::pooled_shared_work (
            pooled_shared_work &&  )  [delete]
```

12.59.4 Member Function Documentation

12.59.4.1 awakened()

```
void trisycl::detail::pooled_shared_work::awakened (
            bf::context * ctx )  [inline], [override], [noexcept]
```

Definition at line 97 of file pooled_shared_work.hpp.

```
00097                                                {
00098     if (ctx->is_context(bf::type::pinned_context)) { /*<
00099 recognize when we're passed this thread's main fiber (or an
00100 implicit library helper fiber):  never put those on the shared
00101 queue
00102 >*/
00103       lqueue_.push_back(*ctx);
00104     } else {
00105       ctx->detach();
00106       std::unique_lock lk { pool_ctx_->rqueue_mtx_ }; /*<
00107 worker fiber, enqueue on shared queue
00108 >*/
00109       pool_ctx_->rqueue_.push_back(ctx);
00110     }
00111   }
```

References lqueue_, and pool_ctx_.

12.59.4.2 create_pool_ctx()

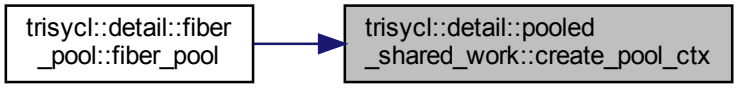
```
static ctx trisycl::detail::pooled_shared_work::create_pool_ctx (
            bool suspend )  [inline], [static]
```

Definition at line 83 of file pooled_shared_work.hpp.

```
00083                                    {
00084     return std::make_shared<pool_ctx>(suspend);
00085   }
```

Referenced by trisycl::detail::fiber_pool::fiber_pool().

Here is the caller graph for this function:



12.59.4.3 has_ready_fibers()

```
bool trisycl::detail::pooled_shared_work::has_ready_fibers ( ) const  [inline], [override],
[noexcept]
```

Definition at line 142 of file pooled_shared_work.hpp.

```
00142                                                {
00143     std::unique_lock lock { pool_ctx_->rqueue_mtx_ };
00144     return !pool_ctx_->rqueue_.empty() || !lqueue_.empty();
00145   }
```

References lqueue_, and pool_ctx_.

12.59.4.4 notify()

```
void trisycl::detail::pooled_shared_work::notify ( )  [inline], [override], [noexcept]
```

Definition at line 164 of file pooled_shared_work.hpp.

```
00164                                    {
00165     if (pool_ctx_->suspend_) {
00166       std::unique_lock lk { mtx_ };
00167       flag_ = true;
00168       lk.unlock();
00169       cnd_.notify_all();
00170     }
00171   }
```

References cnd_, flag_, mtx_, and pool_ctx_.

12.59.4.5 operator=() [1/2]

```
pooled_shared_work & trisycl::detail::pooled_shared_work::operator= (
            pooled_shared_work &&  )  [delete]
```

12.59.4.6 operator=() [2/2]

```
pooled_shared_work & trisycl::detail::pooled_shared_work::operator= (
            pooled_shared_work const &  )  [delete]
```

12.59.4.7 pick_next()

```
bf::context * trisycl::detail::pooled_shared_work::pick_next ( )  [inline], [override], [noexcept]
```

Definition at line 114 of file pooled_shared_work.hpp.

```
00114                                                {
00115     bf::context * ctx = nullptr;
00116     std::unique_lock lk { pool_ctx_->rqueue_mtx_ };
00117     auto &rq = pool_ctx_->rqueue_;
00118     if (!rq.empty()) { /*<
00119 pop an item from the ready queue
00120 >*/
00121       ctx = rq.front();
00122       rq.pop_front();
00123       lk.unlock();
00124       BOOST_ASSERT(nullptr != ctx);
00125       bf::context::active()->attach(ctx); /*<
00126 attach context to current scheduler via the active fiber
00127 of this thread
00128 >*/
00129     } else {
00130       lk.unlock();
00131       if (!lqueue_.empty()) { /*<
00132 nothing in the ready queue, return main or dispatcher fiber
00133 >*/
00134         ctx = & lqueue_.front();
00135         lqueue_.pop_front();
00136       }
00137     }
00138     return ctx;
00139   }
```

References lqueue_, and pool_ctx_.

12.59.4.8 suspend_until()

```
void trisycl::detail::pooled_shared_work::suspend_until (
            std::chrono::steady_clock::time_point const & time_point )  [inline], [override],
[noexcept]
```

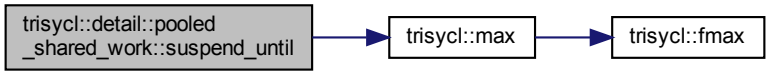
Definition at line 148 of file pooled_shared_work.hpp.

```
00149                       {
00150     if (pool_ctx_->suspend_) {
00151       if (std::chrono::steady_clock::time_point::max() == time_point) {
00152         std::unique_lock lk { mtx_ };
00153         cnd_.wait(lk, [this](){ return flag_; });
00154         flag_ = false;
00155       } else {
```

```
00156         std::unique_lock lk { mtx_ };
00157         cnd_.wait_until(lk, time_point, [this](){ return flag_; });
00158         flag_ = false;
00159       }
00160     }
00161   }
```

References cnd_, flag_, trisycl::max(), mtx_, and pool_ctx_.

Here is the call graph for this function:



12.59.5 Member Data Documentation

12.59.5.1 cnd_

```
std::condition_variable trisycl::detail::pooled_shared_work::cnd_ {}  [private]
```

Definition at line 77 of file pooled_shared_work.hpp.

Referenced by notify(), and suspend_until().

12.59.5.2 flag_

```
bool trisycl::detail::pooled_shared_work::flag_ { false }  [private]
```

Definition at line 78 of file pooled_shared_work.hpp.

Referenced by notify(), and suspend_until().

12.59.5.3 lqueue_

```
lqueue_type trisycl::detail::pooled_shared_work::lqueue_ {}  [private]
```

The runnable local fibers bound to the thread.

Definition at line 73 of file pooled_shared_work.hpp.

Referenced by awakened(), has_ready_fibers(), and pick_next().

12.59.5.4 mtx_

```
std::mutex trisycl::detail::pooled_shared_work::mtx_ {}  [private]
```

The thread-local suspend/notify mechanics.

Definition at line 76 of file pooled_shared_work.hpp.

Referenced by notify(), and suspend_until().

12.59.5.5 pool_ctx_

```
ctx trisycl::detail::pooled_shared_work::pool_ctx_  [private]
```

Some shared datastructure among the working threads.

Definition at line 70 of file pooled_shared_work.hpp.

Referenced by awakened(), has_ready_fibers(), notify(), pick_next(), and suspend_until().

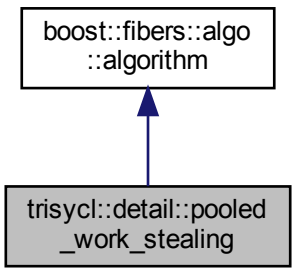
The documentation for this class was generated from the following file:

- include/triSYCL/detail/fiber/pooled_shared_work.hpp

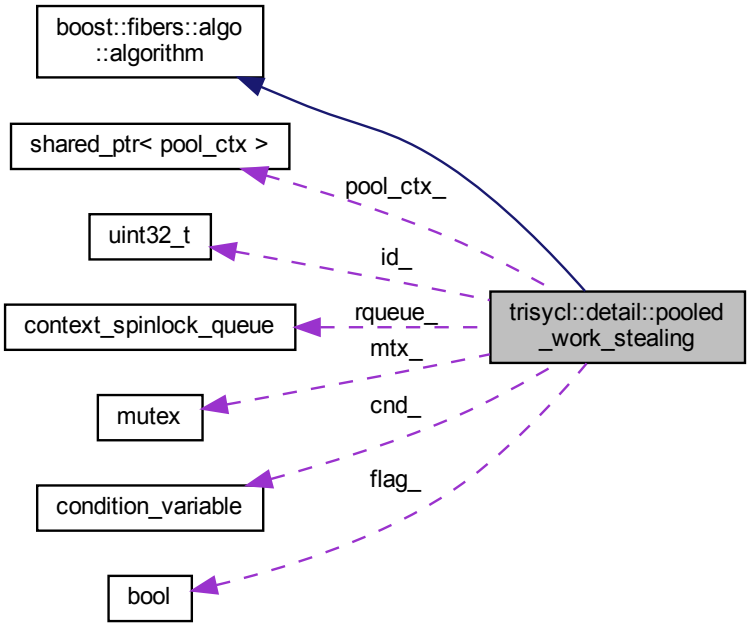
12.60 trisycl::detail::pooled_work_stealing Class Reference

```
#include <pooled_work_stealing.hpp>
```

Inheritance diagram for trisycl::detail::pooled_work_stealing:



Collaboration diagram for trisycl::detail::pooled_work_stealing:



Classes

- struct pool_ctx

  Shared storage among the working threads.

Public Types

- using ctx = std::shared_ptr< pool_ctx >

  Type tracking the common worker data.

Public Member Functions

- pooled_work_stealing (const ctx &pc)
- ~pooled_work_stealing ()=default
- pooled_work_stealing (pooled_work_stealing const &)=delete
- pooled_work_stealing (pooled_work_stealing &&)=delete
- pooled_work_stealing & operator= (pooled_work_stealing const &)=delete
- pooled_work_stealing & operator= (pooled_work_stealing &&)=delete
- void awakened (boost::fibers::context *ctx) noexcept override
- bf::context * pick_next () noexcept override
- virtual boost::fibers::context * steal () noexcept
- bool has_ready_fibers () const noexcept override
- void suspend_until (std::chrono::steady_clock::time_point const &time_point) noexcept override
- void notify () noexcept override

Static Public Member Functions

- static ctx create_pool_ctx (std::uint32_t thread_count, bool suspend)

Private Attributes

- ctx pool_ctx_

  Some shared datastructure among the working threads.
- std::uint32_t id_

  The thread order in the working pool. 0 is first starting thread.
- boost::fibers::detail::context_spinlock_queue rqueue_ {}

  The queue of thread-local runnable fibers.
- std::mutex mtx_ {}

  The thread-local suspend/notify mechanics.
- std::condition_variable cnd_ {}
- bool flag_ { false }

12.60.1 Detailed Description

Definition at line 42 of file pooled_work_stealing.hpp.

12.60.2 Member Typedef Documentation

12.60.2.1 ctx

```
using trisycl::detail::pooled_work_stealing::ctx = std::shared_ptr<pool_ctx>
```

Type tracking the common worker data.

Definition at line 72 of file pooled_work_stealing.hpp.

12.60.3 Constructor & Destructor Documentation

12.60.3.1 pooled_work_stealing() [1/3]

```
trisycl::detail::pooled_work_stealing::pooled_work_stealing (
            const ctx & pc )  [inline]
```

Definition at line 102 of file pooled_work_stealing.hpp.

```
00103     :  pool_ctx_ { pc }
00104     , id_ { pool_ctx_->counter_++ } {
00105       pool_ctx_->schedulers_[id_] = this;
00106       pool_ctx_->barrier_.wait();
00107     }
```

References id_, and pool_ctx_.

12.60.3.2 ~pooled_work_stealing()

```
trisycl::detail::pooled_work_stealing::~pooled_work_stealing ( )  [default]
```

12.60.3.3 pooled_work_stealing() [2/3]

```
trisycl::detail::pooled_work_stealing::pooled_work_stealing (
            pooled_work_stealing const &  )  [delete]
```

12.60.3.4 pooled_work_stealing() [3/3]

```
trisycl::detail::pooled_work_stealing::pooled_work_stealing (
            pooled_work_stealing &&  )  [delete]
```

12.60.4 Member Function Documentation

12.60.4.1 awakened()

```
void trisycl::detail::pooled_work_stealing::awakened (
            boost::fibers::context * ctx )  [inline], [override], [noexcept]
```

Definition at line 120 of file pooled_work_stealing.hpp.

```
00120                                                                {
00121     if (!ctx->is_context(boost::fibers::type::pinned_context))
00122       ctx->detach();
00123     rqueue_.push(ctx);
00124   }
```

References rqueue_.

12.60.4.2 create_pool_ctx()

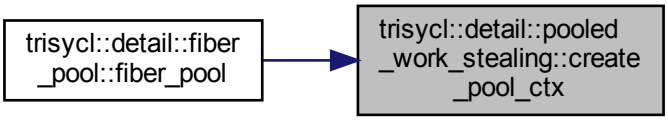
```
static ctx trisycl::detail::pooled_work_stealing::create_pool_ctx (
            std::uint32_t thread_count,
            bool suspend )  [inline], [static]
```

Definition at line 97 of file pooled_work_stealing.hpp.

```
00097                                                                {
00098     return std::make_shared<pool_ctx>(thread_count, suspend);
00099   }
```

Referenced by trisycl::detail::fiber_pool::fiber_pool().

Here is the caller graph for this function:



12.60.4.3 has_ready_fibers()

```
bool trisycl::detail::pooled_work_stealing::has_ready_fibers ( ) const  [inline], [override],
[noexcept]
```

Definition at line 171 of file pooled_work_stealing.hpp.

```
00171                                                  {
00172     return !rqueue_.empty();
00173   }
```

References rqueue_.

12.60.4.4 notify()

```
void trisycl::detail::pooled_work_stealing::notify ( )  [inline], [override], [noexcept]
```

Definition at line 193 of file pooled_work_stealing.hpp.

```
00193                                  {
00194     if (pool_ctx_->suspend_) {
00195       std::unique_lock lk { mtx_ };
00196       flag_ = true;
00197       lk.unlock();
00198       cnd_.notify_all();
00199     }
00200   }
```

References cnd_, flag_, mtx_, and pool_ctx_.

12.60.4.5 operator=() [1/2]

```
pooled_work_stealing & trisycl::detail::pooled_work_stealing::operator= (
            pooled_work_stealing &&  )  [delete]
```

12.60.4.6 operator=() [2/2]

```
pooled_work_stealing & trisycl::detail::pooled_work_stealing::operator= (
            pooled_work_stealing const &  )  [delete]
```

12.60.4.7 pick_next()

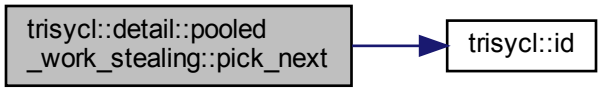
```
bf::context * trisycl::detail::pooled_work_stealing::pick_next ( )  [inline], [override], [noexcept]
```

Definition at line 127 of file pooled_work_stealing.hpp.

```
00127                                          {
00128     bf::context * victim = rqueue_.pop();
00129     if (nullptr != victim) {
00130       boost::context::detail::prefetch_range(victim, sizeof(*victim));
00131       if (!victim->is_context(bf::type::pinned_context)) {
00132         bf::context::active()->attach(victim);
00133       }
00134     }
00135     else {
00136       std::size_t size = pool_ctx_->thread_count_;
00137       //  Work stealing is only possible with more than 1 thread
00138       if (BOOST_LIKELY(size > 1)) {
00139         std::size_t count = 0;
00140         static thread_local std::minstd_rand generator { std::random_device{}() };
00141         std::uniform_int_distribution< std::uint32_t > distribution
00142           { 0, static_cast<std::uint32_t>(size - 1) };
00143         do {
00144           std::uint32_t id = 0;
00145           do {
00146             ++count;
00147             // Random selection of one logical CPU that belongs to the
00148             // local NUMA node
00149             id = distribution(generator);
00150             // Prevent stealing from own scheduler
00151           } while (id == id_);
00152           // Steal context from other scheduler
00153           victim = pool_ctx_->schedulers_[id]->steal();
00154         } while (nullptr == victim && count < size);
00155         if (nullptr != victim) {
00156           boost::context::detail::prefetch_range(victim, sizeof(*victim));
00157           BOOST_ASSERT(!victim->is_context(bf::type::pinned_context));
00158           bf::context::active()->attach(victim);
00159         }
00160       }
00161     }
00162     return victim;
00163   }
```

References trisycl::id(), id_, pool_ctx_, and rqueue_.

Here is the call graph for this function:



12.60.4.8 steal()

```
virtual boost::fibers::context * trisycl::detail::pooled_work_stealing::steal ( )  [inline],
[virtual], [noexcept]
```

Definition at line 166 of file pooled_work_stealing.hpp.

```
00166                                                {
00167     return rqueue_.steal();
00168   }
```

References rqueue_.

12.60.4.9 suspend_until()

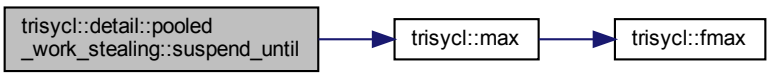
```
void trisycl::detail::pooled_work_stealing::suspend_until (
             std::chrono::steady_clock::time_point const & time_point )  [inline], [override],
[noexcept]
```

Definition at line 176 of file pooled_work_stealing.hpp.

```
00177                                                {
00178     if (pool_ctx_->suspend_) {
00179       if (std::chrono::steady_clock::time_point::max() == time_point) {
00180         std::unique_lock lk { mtx_ };
00181         cnd_.wait(lk, [&] { return flag_; });
00182         flag_ = false;
00183       }
00184       else {
00185         std::unique_lock lk { mtx_ };
00186         cnd_.wait_until(lk, time_point, [&] { return flag_; });
00187         flag_ = false;
00188       }
00189     }
00190   }
```

References cnd_, flag_, trisycl::max(), mtx_, and pool_ctx_.

Here is the call graph for this function:



12.60.5 Member Data Documentation

12.60.5.1 cnd_

```
std::condition_variable trisycl::detail::pooled_work_stealing::cnd_ {}  [private]
```

Definition at line 91 of file pooled_work_stealing.hpp.

Referenced by notify(), and suspend_until().

12.60.5.2 flag_

```
bool trisycl::detail::pooled_work_stealing::flag_ { false }  [private]
```

Definition at line 92 of file pooled_work_stealing.hpp.

Referenced by notify(), and suspend_until().

12.60.5.3 id_

```
std::uint32_t trisycl::detail::pooled_work_stealing::id_  [private]
```

The thread order in the working pool. 0 is first starting thread.

Definition at line 80 of file pooled_work_stealing.hpp.

Referenced by pick_next(), and pooled_work_stealing().

12.60.5.4 mtx_

```
std::mutex trisycl::detail::pooled_work_stealing::mtx_ {}  [private]
```

The thread-local suspend/notify mechanics.

Definition at line 90 of file pooled_work_stealing.hpp.

Referenced by notify(), and suspend_until().

12.60.5.5 pool_ctx_

```
ctx trisycl::detail::pooled_work_stealing::pool_ctx_  [private]
```

Some shared datastructure among the working threads.

Definition at line 77 of file pooled_work_stealing.hpp.

Referenced by notify(), pick_next(), pooled_work_stealing(), and suspend_until().

12.60.5.6 rqueue_

```
boost::fibers::detail::context_spinlock_queue trisycl::detail::pooled_work_stealing::rqueue_
{}  [private]
```

The queue of thread-local runnable fibers.

Definition at line 86 of file pooled_work_stealing.hpp.

Referenced by awakened(), has_ready_fibers(), pick_next(), and steal().

The documentation for this class was generated from the following file:

- include/triSYCL/detail/fiber/pooled_work_stealing.hpp

12.61 trisycl::vendor::xilinx::acap::aie::port Struct Reference

Abstraction of a communication port.

```
#include <connection.hpp>
```

Classes

- struct shim

  Abstraction of a communication port inside a shim tile.
- struct tile

  Abstraction of a communication port inside a tile.

12.61.1 Detailed Description

Abstraction of a communication port.

Definition at line 245 of file connection.hpp.

The documentation for this struct was generated from the following file:

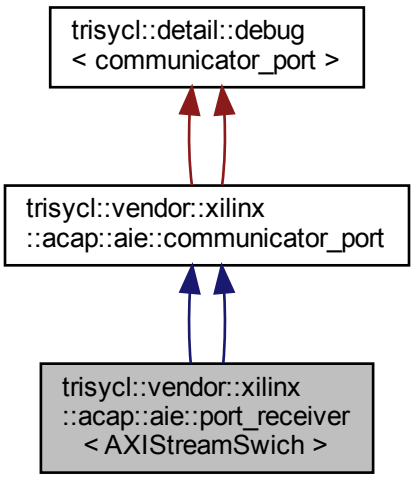
- include/sycl/vendor/Xilinx/acap/aie/connection.hpp

12.62 trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich > Class Template Reference

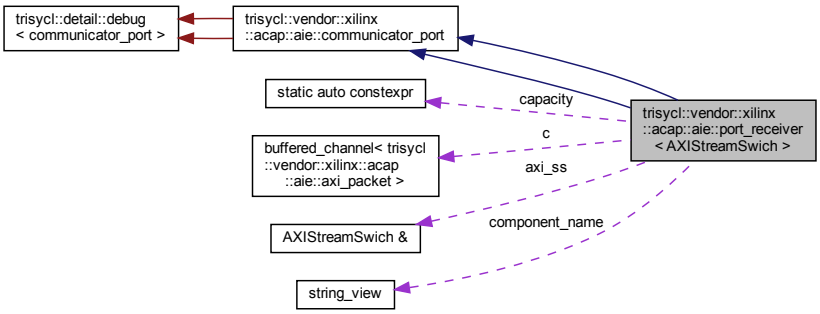
A router input port directing to an AIE core input or a BLI input.

`#include <connection.hpp>`

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >:



Public Member Functions

- port_receiver (AXIStreamSwich &axi_ss, std::string_view component_name)
- void write (const axi_packet &v) override

  Enqueue a packet (coming from the switch) to the core input.

- bool try_write (const axi_packet &v) override
  Try to enqueue a packet to the core input.
- value_type read () override
  Waiting read by a tile program on a core input port from the switch.
- bool try_read (value_type &v) override
  Non-blocking read to a core input port.
- port_receiver (AXIStreamSwich &axi_ss, std::string_view component_name)
- void write (const axi_packet &v) override
  Enqueue a packet (coming from the switch) to the core input.
- bool try_write (const axi_packet &v) override
  Try to enqueue a packet to the core input.
- value_type read () override
  Waiting read by a tile program on a core input port from the switch.
- bool try_read (value_type &v) override
  Non-blocking read to a core input port.

Private Types

- using value_type = axi_packet::value_type
  Payload data type.
- using value_type = axi_packet::value_type
  Payload data type.

Private Attributes

- boost::fibers::buffered_channel< axi_packet > c { capacity }
- AXIStreamSwich & axi_ss
  Keep track of the AXI stream switch owning this port for debugging.
- std::string_view component_name
  The component model during debug.

Static Private Attributes

- static auto constexpr capacity = 2
  Router ingress capacity queue.

12.62.1 Detailed Description

template<typename AXIStreamSwich>
class trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >

A router input port directing to an AIE core input or a BLI input.

Todo factorize with fifo_port

Todo factorize with fifo_port

Definition at line 168 of file connection.hpp.

12.62.2 Member Typedef Documentation

12.62.2.1 value_type [1/2]

```
template<typename AXIStreamSwich >
using trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::value_type = axi_packet::value_typ
[private]
```

Payload data type.

Definition at line 174 of file connection.hpp.

12.62.2.2 value_type [2/2]

```
template<typename AXIStreamSwich >
using trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::value_type = axi_packet::value_typ
[private]
```

Payload data type.

Definition at line 174 of file connection.hpp.

12.62.3 Constructor & Destructor Documentation

12.62.3.1 port_receiver() [1/2]

```
template<typename AXIStreamSwich >
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::port_receiver (
            AXIStreamSwich & axi_ss,
            std::string_view component_name )  [inline]
```

Definition at line 191 of file connection.hpp.

```
00192     :  axi_ss { axi_ss }, component_name { component_name } {}
```

12.62.3.2 port_receiver() [2/2]

```
template<typename AXIStreamSwich >
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::port_receiver (
            AXIStreamSwich & axi_ss,
            std::string_view component_name )  [inline]
```

Definition at line 191 of file connection.hpp.

```
00192     :  axi_ss { axi_ss }, component_name { component_name } {}
```

12.62.4 Member Function Documentation

12.62.4.1 read() [1/2]

```
template<typename AXIStreamSwich >
value_type trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::read ( )
[inline], [override], [virtual]
```

Waiting read by a tile program on a core input port from the switch.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 217 of file connection.hpp.

```
00217                                {
00218     TRISYCL_DUMP_T(component_name « ' ' « this « " on tile("
00219                    « axi_ss.x_coordinate
00220                    « ',' « axi_ss.y_coordinate
00221                    « ") on fiber " « boost::this_fiber::get_id()
00222                    « " reading from buffered_channel " « &c « "...");
00223     return c.value_pop().data;
00224   }
```

References trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::axi_ss, trisycl::vendor::xilinx::acap::aie::port_receive
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::component_name, and TRISYCL_DUMP_T.

12.62.4.2 read() [2/2]

```
template<typename AXIStreamSwich >
value_type trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::read ( )
[inline], [override], [virtual]
```

Waiting read by a tile program on a core input port from the switch.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 217 of file connection.hpp.

```
00217                                {
00218     TRISYCL_DUMP_T(component_name « ' ' « this « " on tile("
00219                    « axi_ss.x_coordinate
00220                    « ',' « axi_ss.y_coordinate
00221                    « ") on fiber " « boost::this_fiber::get_id()
00222                    « " reading from buffered_channel " « &c « "...");
00223     return c.value_pop().data;
00224   }
```

References trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::axi_ss, trisycl::vendor::xilinx::acap::aie::port_receive
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::component_name, and TRISYCL_DUMP_T.

12.62.4.3 try_read() [1/2]

```
template<typename AXIStreamSwich >
bool trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::try_read (
            value_type & v )  [inline], [override], [virtual]
```

Non-blocking read to a core input port.

Returns

true if the value was correctly read

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 231 of file connection.hpp.

```
00231                                       {
00232     axi_packet p;
00233
00234     if (c.try_pop(p) == boost::fibers::channel_op_status::success) {
00235       v = p.data;
00236       return true;
00237     }
00238     return false;
00239   }
```

References trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::c, and p.

12.62.4.4 try_read() [2/2]

```
template<typename AXIStreamSwich >
bool trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::try_read (
            value_type & v )  [inline], [override], [virtual]
```

Non-blocking read to a core input port.

Returns

true if the value was correctly read

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 231 of file connection.hpp.

```
00231                                       {
00232     axi_packet p;
00233
00234     if (c.try_pop(p) == boost::fibers::channel_op_status::success) {
00235       v = p.data;
00236       return true;
00237     }
00238     return false;
00239   }
```

References trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::c, and p.

12.62.4.5 try_write() [1/2]

```
template<typename AXIStreamSwich >
bool trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::try_write (
            const axi_packet & v )  [inline], [override], [virtual]
```

Try to enqueue a packet to the core input.

Returns

true if the packet is correctly enqueued

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 211 of file connection.hpp.

```
00211                                          {
00212     return c.try_push(v) == boost::fibers::channel_op_status::success;
00213   }
```

References trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::c.

12.62.4.6 try_write() [2/2]

```
template<typename AXIStreamSwich >
bool trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::try_write (
            const axi_packet & v )  [inline], [override], [virtual]
```

Try to enqueue a packet to the core input.

Returns

true if the packet is correctly enqueued

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 211 of file connection.hpp.

```
00211                                          {
00212     return c.try_push(v) == boost::fibers::channel_op_status::success;
00213   }
```

References trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::c.

12.62.4.7 write() [1/2]

```
template<typename AXIStreamSwich >
void trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::write (
            const axi_packet & v )  [inline], [override], [virtual]
```

Enqueue a packet (coming from the switch) to the core input.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 196 of file connection.hpp.

```
00196                                       {
00197     TRISYCL_DUMP_T(component_name « ' ' « this « " on tile("
00198                    « axi_ss.x_coordinate
00199                    « ',' « axi_ss.y_coordinate
00200                    « ") on fiber " « boost::this_fiber::get_id()
00201                    « " write data value " « v.data
00202                    « " to buffered_channel " « &c);
00203     c.push(v);
00204   }
```

References trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::axi_ss, trisycl::vendor::xilinx::acap::aie::port_receive
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::component_name, trisycl::vendor::xilinx::acap::aie::axi_packet::c
and TRISYCL_DUMP_T.

12.62.4.8 write() [2/2]

```
template<typename AXIStreamSwich >
void trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::write (
            const axi_packet & v )  [inline], [override], [virtual]
```

Enqueue a packet (coming from the switch) to the core input.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 196 of file connection.hpp.

```
00196                                       {
00197     TRISYCL_DUMP_T(component_name « ' ' « this « " on tile("
00198                    « axi_ss.x_coordinate
00199                    « ',' « axi_ss.y_coordinate
00200                    « ") on fiber " « boost::this_fiber::get_id()
00201                    « " write data value " « v.data
00202                    « " to buffered_channel " « &c);
00203     c.push(v);
00204   }
```

References trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::axi_ss, trisycl::vendor::xilinx::acap::aie::port_receive
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::component_name, and TRISYCL_DUMP_T.

12.62.5 Member Data Documentation

12.62.5.1 axi_ss

```
template<typename AXIStreamSwich >
AXIStreamSwich & trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::axi_ss
[private]
```

Keep track of the AXI stream switch owning this port for debugging.

Definition at line 184 of file connection.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::read(), and trisycl::vendor::xilinx::acap::aie::port_

12.62.5.2 c

```
template<typename AXIStreamSwich >
boost::fibers::buffered_channel< axi_packet > trisycl::vendor::xilinx::acap::aie::port_receiver<
AXIStreamSwich >::c { capacity }  [private]
```

Definition at line 181 of file connection.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::read(), trisycl::vendor::xilinx::acap::aie::port_rece
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::try_write(), and trisycl::vendor::xilinx::acap::aie::port_receiver< A

12.62.5.3 capacity

```
template<typename AXIStreamSwich >
static auto constexpr trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >↩
::capacity = 2  [static], [constexpr], [private]
```

Router ingress capacity queue.

Todo check with hardware team for the value

Todo check with hardware team for the value

Definition at line 171 of file connection.hpp.

12.62.5.4 component_name

```
template<typename AXIStreamSwich >
std::string_view trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::component↩
_name  [private]
```

The component model during debug.

Definition at line 187 of file connection.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::read(), and trisycl::vendor::xilinx::acap::aie::port_

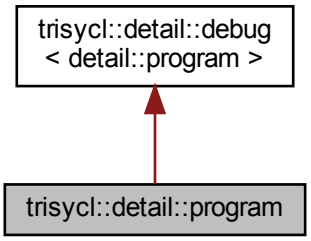
The documentation for this class was generated from the following files:

- include/sycl/vendor/Xilinx/acap/aie/connection.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/connection.hpp

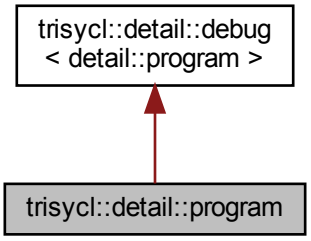
12.63 trisycl::detail::program Class Reference

`#include <program.hpp>`

Inheritance diagram for trisycl::detail::program:



Collaboration diagram for trisycl::detail::program:



Public Member Functions

- virtual cl_program get () const =0

  Return the OpenCL kernel object for this kernel.
- virtual ~program ()

Additional Inherited Members

12.63.1 Detailed Description

Definition at line 20 of file program.hpp.

12.63.2 Constructor & Destructor Documentation

12.63.2.1 ~program()

```
virtual trisycl::detail::program::~program ( )  [inline], [virtual]
```

Definition at line 33 of file program.hpp.

```
00033 {}
```

12.63.3 Member Function Documentation

12.63.3.1 get()

```
virtual cl_program trisycl::detail::program::get ( ) const  [pure virtual]
```

Return the OpenCL kernel object for this kernel.

Retains a reference to the returned cl_kernel object. Caller should release it when finished.

The documentation for this class was generated from the following file:

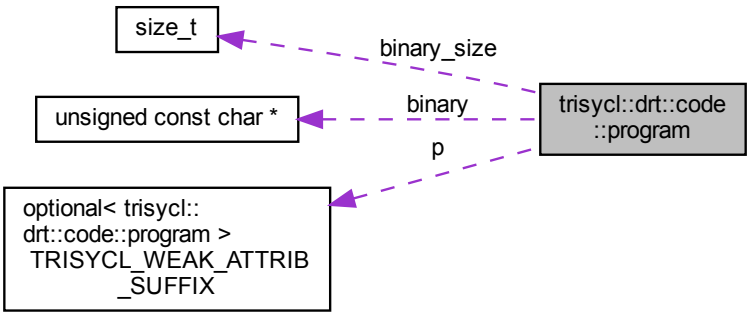
- include/triSYCL/program/detail/program.hpp

12.64 trisycl::drt::code::program Struct Reference

A binary program containing some kernels.

```
#include <device_runtime.hpp>
```

Collaboration diagram for trisycl::drt::code::program:



Public Member Functions

- program (std::size_t binary_size, const char *binary)

Public Attributes

- std::size_t binary_size

  The size of a binary program.
- unsigned const char * binary

  The bytes of program. Use this type for `boost::compute`.

Static Public Attributes

- static TRISYCL_WEAK_ATTRIB_PREFIX boost::optional< program > TRISYCL_WEAK_ATTRIB_SUFFIX p

  The current program.

12.64.1 Detailed Description

A binary program containing some kernels.

Definition at line 156 of file device_runtime.hpp.

12.64.2 Constructor & Destructor Documentation

12.64.2.1 program()

```
trisycl::drt::code::program::program (
              std::size_t binary_size,
              const char * binary )  [inline]
```

Definition at line 169 of file device_runtime.hpp.

```
00170       :  binary_size { binary_size }
00171       , binary { reinterpret_cast<unsigned const char *>(binary) } {
00172         p = *this;
00173         TRISYCL_DUMP_T("Create program with binary size = 0x" « binary_size);
00174       }
```

References binary_size, p, and TRISYCL_DUMP_T.

12.64.3 Member Data Documentation

12.64.3.1 binary

```
unsigned const char* trisycl::drt::code::program::binary
```

The bytes of program. Use this type for `boost::compute`.

Definition at line 160 of file device_runtime.hpp.

12.64.3.2 binary_size

```
std::size_t trisycl::drt::code::program::binary_size
```

The size of a binary program.

Definition at line 158 of file device_runtime.hpp.

Referenced by program().

12.64.3.3 p

```
TRISYCL_WEAK_ATTRIB_PREFIX boost::optional< program > TRISYCL_WEAK_ATTRIB_SUFFIX trisycl↩
::drt::code::program::p  [static]
```

The current program.

Todo Deal with several programs

Definition at line 167 of file device_runtime.hpp.

Referenced by program(), and trisycl::drt::set_kernel().

The documentation for this struct was generated from the following file:

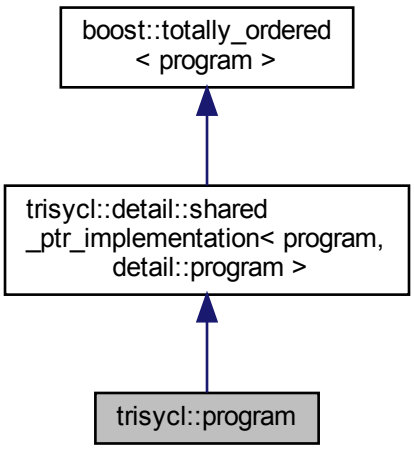
- include/triSYCL/device_runtime.hpp

12.65 trisycl::program Class Reference

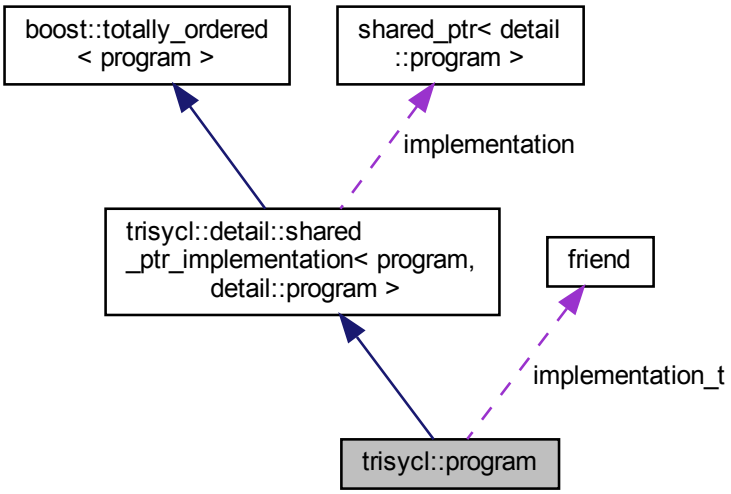
SYCL program.

`#include <program.hpp>`

Inheritance diagram for trisycl::program:



Collaboration diagram for trisycl::program:



Public Member Functions

- program ()=delete
- program (const context &context, cl_program p)
- cl_program get () const
- template<typename kernelT >
  bool has_kernel () const
- bool has_kernel (string_class kernelName) const
- template<typename kernelT >
  void compile_with_kernel_type (string_class compileOptions="")
- template<typename kernelT >
  void build_with_kernel_type (string_class buildOptions="")
- template<typename kernelT >
  ::trisycl::kernel get_kernel () const
- kernel get_kernel (string_class kernelName) const

Private Types

- using implementation_t = typename program::shared_ptr_implementation

Private Attributes

- friend implementation_t

Friends

- class handler

Additional Inherited Members

12.65.1 Detailed Description

SYCL program.

Todo To be implemented

Definition at line 36 of file program.hpp.

12.65.2 Member Typedef Documentation

12.65.2.1 implementation_t

```
using trisycl::program::implementation_t = typename program::shared_ptr_implementation  [private]
```

Definition at line 42 of file program.hpp.

12.65.3 Constructor & Destructor Documentation

12.65.3.1 program() [1/2]

```
trisycl::program::program ( )  [delete]
```

12.65.3.2 program() [2/2]

```
trisycl::program::program (
            const context & context,
            cl_program p )  [inline]
```

Definition at line 59 of file program.hpp.

```
00059 {}
```

12.65.4 Member Function Documentation

12.65.4.1 build_with_kernel_type()

```
template<typename kernelT >
void trisycl::program::build_with_kernel_type (
            string_class buildOptions = "" )  [inline]
```

Definition at line 77 of file program.hpp.

```
00078   {
00079   }
```

12.65.4.2 compile_with_kernel_type()

```
template<typename kernelT >
void trisycl::program::compile_with_kernel_type (
            string_class compileOptions = "" )  [inline]
```

Definition at line 71 of file program.hpp.

```
00072   {
00073
00074   }
```

12.65.4.3 get()

```
cl_program trisycl::program::get ( ) const  [inline]
```

Definition at line 61 of file program.hpp.

```
00061                          {
00062     return implementation->get();
00063   }
```

References trisycl::detail::shared_ptr_implementation< program, detail::program >::implementation.

12.65.4.4 get_kernel() [1/2]

```
template<typename kernelT >
::trisycl::kernel trisycl::program::get_kernel ( ) const
```

12.65.4.5 get_kernel() [2/2]

```
kernel trisycl::program::get_kernel (
             string_class kernelName ) const  [inline]
```

Definition at line 84 of file program.hpp.

```
00084                                                {
00085     return NULL;
00086   }
```

12.65.4.6 has_kernel() [1/2]

```
template<typename kernelT >
bool trisycl::program::has_kernel ( ) const
```

12.65.4.7 has_kernel() [2/2]

```
bool trisycl::program::has_kernel (
             string_class kernelName ) const
```

12.65.5 Friends And Related Function Documentation

12.65.5.1 handler

```
friend class handler  [friend]
```

Definition at line 45 of file program.hpp.

12.65.6 Member Data Documentation

12.65.6.1 implementation_t

```
friend trisycl::program::implementation_t  [private]
```

Definition at line 48 of file program.hpp.

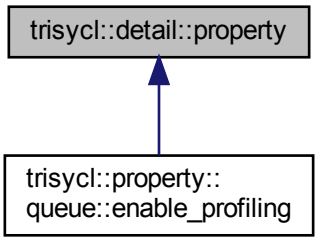
The documentation for this class was generated from the following file:

- include/triSYCL/program.hpp

12.66 trisycl::detail::property Class Reference

```
#include <property.hpp>
```

Inheritance diagram for trisycl::detail::property:



12.66.1 Detailed Description

Definition at line 6 of file property.hpp.

The documentation for this class was generated from the following file:

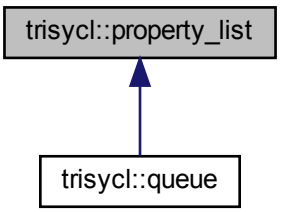
- include/triSYCL/detail/property.hpp

12.67 trisycl::property_list Class Reference

SYCL property list.

`#include <property_list.hpp>`

Inheritance diagram for trisycl::property_list:



Public Member Functions

- template<typename... propsT, typename = std::enable_if_t<detail::all_true<std::is_convertible<propsT, detail::property>::value ...>::value>>
  property_list (propsT... props)
  Construct a property list from a list of classes derived from the detail::property.
- property_list ()

Protected Member Functions

- template<typename propertyT >
  bool has_property () const
- template<typename propertyT >
  propertyT get_property () const

Private Member Functions

- TRISYCL_PROPERTY_CREATE (queue, enable_profiling)
- template<typename T , typename... propsT, typename = std::enable_if_t<detail::all_true<std::is_convertible<propsT, detail::property>::value ...>::value>>
  void addproperty (T first, propsT... next)

12.67.1 Detailed Description

SYCL property list.

Definition at line 16 of file property_list.hpp.

12.67.2 Constructor & Destructor Documentation

12.67.2.1 property_list() [1/2]

```
template<typename... propsT, typename = std::enable_if_t<detail::all_true<std::is_convertible<props↩
T, detail::property>::value ...>::value>>
trisycl::property_list::property_list (
            propsT... props ) [inline]
```

Construct a property list from a list of classes derived from the detail::property.

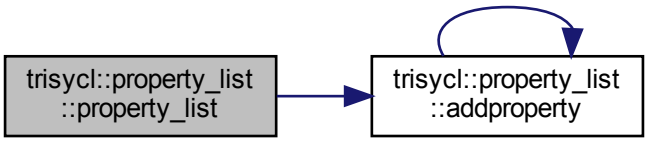
The detail::property and all_true are required to disambiguate queue/buffer constructors, so that property_lists are only created from property classes.

Definition at line 46 of file property_list.hpp.

```
00046                                {
00047     addproperty(props...);
00048   }
```

References addproperty().

Here is the call graph for this function:



12.67.2.2 property_list() [2/2]

```
trisycl::property_list::property_list ( ) [inline]
```

Definition at line 50 of file property_list.hpp.

```
00050 {}
```

12.67.3 Member Function Documentation

12.67.3.1 addproperty()

```
template<typename T , typename...  propsT, typename = std::enable_if_t<detail::all_true<std↩
::is_convertible<propsT, detail::property>::value ...>::value>>
void trisycl::property_list::addproperty (
            T first,
            propsT...  next )  [inline], [private]
```

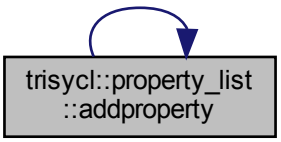
Definition at line 35 of file property_list.hpp.

```
00035                                          {
00036     addproperty(next...);
00037   }
```

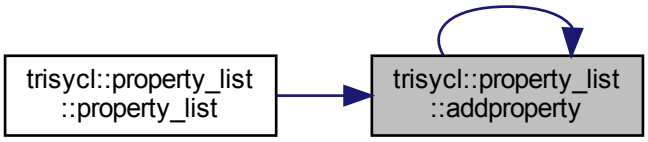
References addproperty().

Referenced by addproperty(), and property_list().

Here is the call graph for this function:



Here is the caller graph for this function:



12.67.3.2 get_property()

```
template<typename propertyT >
propertyT trisycl::property_list::get_property ( ) const  [inline], [protected]
```

12.67.3.3 has_property()

```
template<typename propertyT >
bool trisycl::property_list::has_property ( ) const  [inline], [protected]
```

12.67.3.4 TRISYCL_PROPERTY_CREATE()

```
trisycl::property_list::TRISYCL_PROPERTY_CREATE (
             queue ,
             enable_profiling  )  [private]
```

The documentation for this class was generated from the following file:

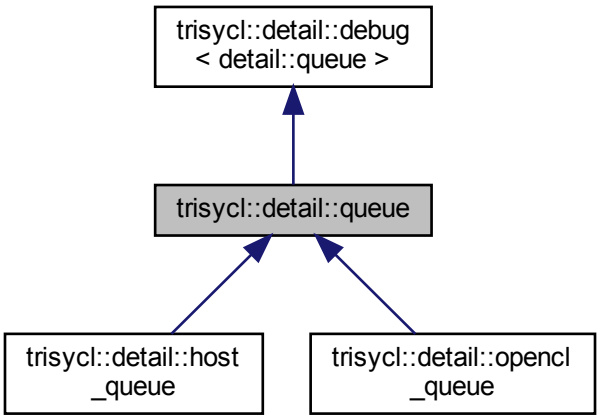
- include/triSYCL/property_list.hpp

12.68 trisycl::detail::queue Struct Reference

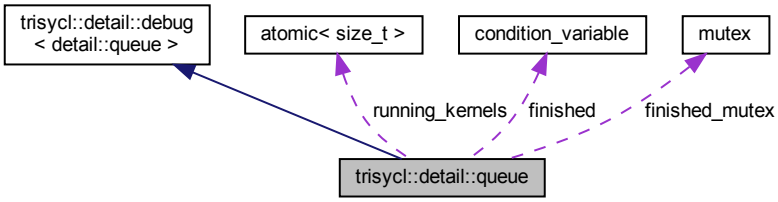
Some implementation details about the SYCL queue.

```
#include <queue.hpp>
```

Inheritance diagram for trisycl::detail::queue:



Collaboration diagram for trisycl::detail::queue:



Public Member Functions

- queue ()

  Initialize the queue with 0 running kernel.
- void wait_for_kernel_execution ()

  Wait for all kernel completion.
- void kernel_start ()

  Signal that a new kernel started on this queue.
- void kernel_end ()

  Signal that a new kernel finished on this queue.
- virtual cl_command_queue get () const =0

  Return the underlying OpenCL command queue after doing a retain.
- virtual boost::compute::command_queue & get_boost_compute ()=0

  Return the underlying Boost.Compute command queue.
- virtual trisycl::context get_context () const =0

  Return the SYCL queue's context.
- virtual trisycl::device get_device () const =0

  Return the SYCL device the queue is associated with.
- virtual bool is_host () const =0

  Return whether the queue is executing on a SYCL host device.
- virtual ~queue ()

  Wait for all kernel completion before the queue destruction.

Public Attributes

- std::atomic< size_t > running_kernels

  Track the number of kernels still running to wait for their completion.
- std::condition_variable finished

  To signal when all the kernels have completed.
- std::mutex finished_mutex

  To protect the access to the condition variable.

Additional Inherited Members

12.68.1 Detailed Description

Some implementation details about the SYCL queue.

Definition at line 28 of file queue.hpp.

12.68.2 Constructor & Destructor Documentation

12.68.2.1 queue()

```
trisycl::detail::queue::queue ( )  [inline]
```

Initialize the queue with 0 running kernel.

Definition at line 39 of file queue.hpp.

```
00039 :  running_kernels { 0 } {}
```

12.68.2.2 ~queue()

```
virtual trisycl::detail::queue::~queue ( )  [inline], [virtual]
```

Wait for all kernel completion before the queue destruction.

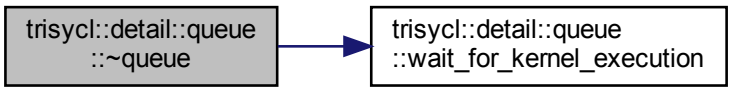
Todo Update according spec since queue destruction is non blocking

Definition at line 118 of file queue.hpp.

```
00118                   {
00119     wait_for_kernel_execution();
00120   }
```

References wait_for_kernel_execution().

Here is the call graph for this function:



12.68.3 Member Function Documentation

12.68.3.1 get()

```
virtual cl_command_queue trisycl::detail::queue::get ( ) const  [pure virtual]
```

Return the underlying OpenCL command queue after doing a retain.

This memory object is expected to be released by the developer.

Retain a reference to the returned cl_command_queue object.

Caller should release it when finished.

If the queue is a SYCL host queue then an exception is thrown.

Implemented in trisycl::detail::host_queue, and trisycl::detail::opencl_queue.

12.68.3.2 get_boost_compute()

```
virtual boost::compute::command_queue & trisycl::detail::queue::get_boost_compute ( )  [pure
virtual]
```

Return the underlying Boost.Compute command queue.

Implemented in trisycl::detail::host_queue, and trisycl::detail::opencl_queue.

12.68.3.3 get_context()

```
virtual trisycl::context trisycl::detail::queue::get_context ( ) const  [pure virtual]
```

Return the SYCL queue's context.

Report errors using SYCL exception classes.

Implemented in trisycl::detail::host_queue, and trisycl::detail::opencl_queue.

12.68.3.4 get_device()

```
virtual trisycl::device trisycl::detail::queue::get_device ( ) const  [pure virtual]
```

Return the SYCL device the queue is associated with.

Report errors using SYCL exception classes.

Implemented in trisycl::detail::host_queue, and trisycl::detail::opencl_queue.

12.68.3.5 is_host()

```
virtual bool trisycl::detail::queue::is_host ( ) const  [pure virtual]
```

Return whether the queue is executing on a SYCL host device.

Implemented in trisycl::detail::host_queue, and trisycl::detail::opencl_queue.

12.68.3.6 kernel_end()

```
void trisycl::detail::queue::kernel_end ( )  [inline]
```

Signal that a new kernel finished on this queue.

Definition at line 61 of file queue.hpp.

```
00061                    {
00062     TRISYCL_DUMP_T("A kernel of the queue ended");
00063     std::unique_lock<std::mutex> ul { finished_mutex };
00064     if (--running_kernels == 0) {
00065       // Micro-optimization:  unlock before the notification
00066       // https://en.cppreference.com/w/cpp/thread/condition_variable/notify_all
00067       ul.unlock();
00068       /* It was the last kernel running, so signal the queue just in
00069 case it was working for it for completion
00070
00071 In some cases several threads might want to wait for the
00072 same queue, because of this \c notify_one is not be enough
00073 and a \c notify_all is needed
00074 */
00075       finished.notify_all();
00076     }
00077   }
```

References finished, finished_mutex, running_kernels, and TRISYCL_DUMP_T.

12.68.3.7 kernel_start()

```
void trisycl::detail::queue::kernel_start ( )  [inline]
```

Signal that a new kernel started on this queue.

Definition at line 53 of file queue.hpp.

```
00053                      {
00054     TRISYCL_DUMP_T("A kernel has been added to the queue");
00055     // One more kernel
00056     ++running_kernels;
00057   }
```

References running_kernels, and TRISYCL_DUMP_T.

12.68.3.8 wait_for_kernel_execution()

```
void trisycl::detail::queue::wait_for_kernel_execution ( )  [inline]
```

Wait for all kernel completion.

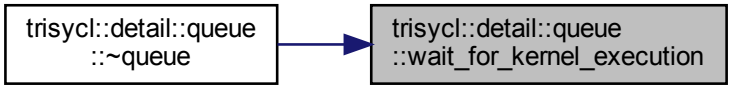
Definition at line 42 of file queue.hpp.

```
00042                                  {
00043     TRISYCL_DUMP_T("Queue waiting for kernel completion");
00044     std::unique_lock<std::mutex> ul { finished_mutex };
00045     finished.wait(ul, [&] {
00046         // When there is no kernel running in this queue, we are ready to go
00047         return running_kernels == 0;
00048       });
00049   }
```

References finished, finished_mutex, running_kernels, and TRISYCL_DUMP_T.

Referenced by ~queue().

Here is the caller graph for this function:



12.68.4 Member Data Documentation

12.68.4.1 finished

```
std::condition_variable trisycl::detail::queue::finished
```

To signal when all the kernels have completed.

Definition at line 33 of file queue.hpp.

Referenced by kernel_end(), and wait_for_kernel_execution().

12.68.4.2 finished_mutex

```
std::mutex trisycl::detail::queue::finished_mutex
```

To protect the access to the condition variable.

Definition at line 35 of file queue.hpp.

Referenced by kernel_end(), and wait_for_kernel_execution().

12.68.4.3 running_kernels

```
std::atomic<size_t> trisycl::detail::queue::running_kernels
```

Track the number of kernels still running to wait for their completion.

Definition at line 30 of file queue.hpp.

Referenced by kernel_end(), kernel_start(), and wait_for_kernel_execution().

The documentation for this struct was generated from the following file:

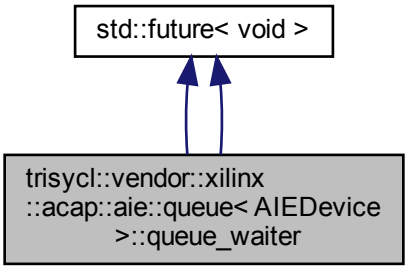
- include/triSYCL/queue/detail/queue.hpp

12.69 trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter Struct Reference

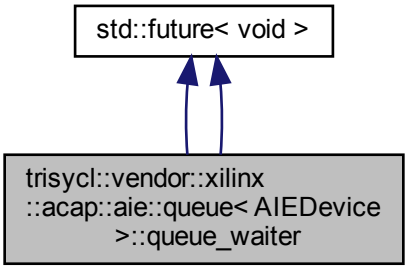
Provide a wait() function that actually can throw an exception instead of ignoring it by the std::future::wait()

```
#include <queue.hpp>
```

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter:



Public Member Functions

- void wait ()
- void wait ()

12.69.1 Detailed Description

template<typename AIEDevice>
struct trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter

Provide a wait() function that actually can throw an exception instead of ignoring it by the std::future::wait()

Definition at line 43 of file queue.hpp.

12.69.2 Member Function Documentation

12.69.2.1 wait() [1/2]

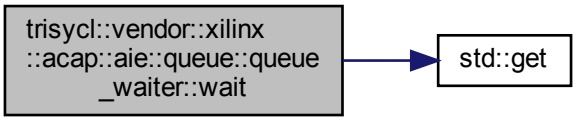
```
template<typename AIEDevice >
void trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter::wait ( )  [inline]
```

Definition at line 44 of file queue.hpp.

```
00044                 {
00045       // Just call the function unwrapping the promise, it will
00046       // rethrow if there is any exception pending
00047       get();
00048     }
```

References std::get().

Here is the call graph for this function:



12.69.2.2 wait() [2/2]

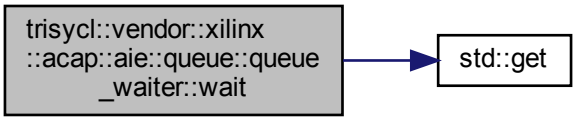
```
template<typename AIEDevice >
void trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter::wait ( )  [inline]
```

Definition at line 44 of file queue.hpp.

```
00044                 {
00045       // Just call the function unwrapping the promise, it will
00046       // rethrow if there is any exception pending
00047       get();
00048     }
```

References std::get().

Here is the call graph for this function:



The documentation for this struct was generated from the following files:

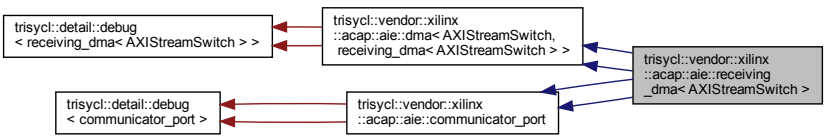
- include/sycl/vendor/Xilinx/acap/aie/queue.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/queue.hpp

12.70 trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch > Class Template Reference

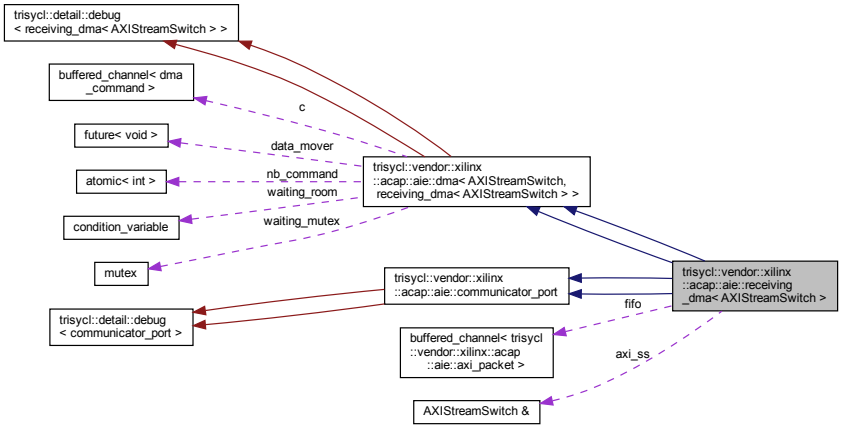
Receiving DMA.

```
#include <dma.hpp>
```

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >:



Public Member Functions

- receiving_dma (AXIStreamSwitch &axi_ss, ::trisycl::detail::fiber_pool &fe)

  Start the receiving DMA engine using an executor.
- void receive (std::span< axi_packet::value_type > sp)

  Enqueue a DMA transfer to receive a span.
- void write (const axi_packet &v) override

  The network sends some data to the DMA receiver input.
- bool try_write (const axi_packet &v) override

  The network try to send some data to the DMA receiver input.
- axi_packet::value_type read () override

  Waiting read by a tile program on a core input port from the switch.
- bool try_read (axi_packet::value_type &v) override

  Non-blocking read to a core input port.
- receiving_dma (AXIStreamSwitch &axi_ss, ::trisycl::detail::fiber_pool &fe)

  Start the receiving DMA engine using an executor.
- void receive (std::span< axi_packet::value_type > sp)

  Enqueue a DMA transfer to receive a span.
- void write (const axi_packet &v) override

  The network sends some data to the DMA receiver input.
- bool try_write (const axi_packet &v) override

  The network try to send some data to the DMA receiver input.
- axi_packet::value_type read () override

  Waiting read by a tile program on a core input port from the switch.
- bool try_read (axi_packet::value_type &v) override

  Non-blocking read to a core input port.

Private Types

- using dma_base = dma< AXIStreamSwitch, receiving_dma< AXIStreamSwitch > >
- using dma_base = dma< AXIStreamSwitch, receiving_dma< AXIStreamSwitch > >

Private Attributes

- boost::fibers::buffered_channel< axi_packet > fifo { 8 }
- AXIStreamSwitch & axi_ss

  Keep track of the AXI stream switch owning this port for debugging.

Additional Inherited Members

12.70.1 Detailed Description

template<typename AXIStreamSwitch>
class trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >

Receiving DMA.

Todo Factorize out the FIFO port

Todo Factorize out the FIFO port

Definition at line 202 of file dma.hpp.

12.70.2 Member Typedef Documentation

12.70.2.1 dma_base [1/2]

```
template<typename AXIStreamSwitch >
using trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::dma_base = dma<AXIStream↩
Switch, receiving_dma<AXIStreamSwitch> >  [private]
```

Definition at line 205 of file dma.hpp.

12.70.2.2 dma_base [2/2]

```
template<typename AXIStreamSwitch >
using trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::dma_base = dma<AXIStream↩
Switch, receiving_dma<AXIStreamSwitch> >  [private]
```

Definition at line 205 of file dma.hpp.

12.70.3 Constructor & Destructor Documentation

12.70.3.1 receiving_dma() [1/2]

```
template<typename AXIStreamSwitch >
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::receiving_dma (
            AXIStreamSwitch & axi_ss,
            ::trisycl::detail::fiber_pool & fe )  [inline]
```

Start the receiving DMA engine using an executor.

Definition at line 219 of file dma.hpp.

```
00220       :  dma_base { fe }
00221       , axi_ss { axi_ss } {}
```

12.70.3.2 receiving_dma() [2/2]

```
template<typename AXIStreamSwitch >
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::receiving_dma (
            AXIStreamSwitch & axi_ss,
            ::trisycl::detail::fiber_pool & fe )  [inline]
```

Start the receiving DMA engine using an executor.

Definition at line 219 of file dma.hpp.

```
00220       :  dma_base { fe }
00221       , axi_ss { axi_ss } {}
```

12.70.4 Member Function Documentation

12.70.4.1 read() [1/2]

```
template<typename AXIStreamSwitch >
axi_packet::value_type trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >↩
::read ( )  [inline], [override], [virtual]
```

Waiting read by a tile program on a core input port from the switch.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

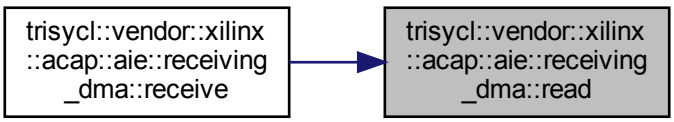
Definition at line 253 of file dma.hpp.

```
00253 { return fifo.value_pop().data; }
```

References trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::fifo.

Referenced by trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::receive().

Here is the caller graph for this function:



12.70.4.2 read() [2/2]

```
template<typename AXIStreamSwitch >
axi_packet::value_type trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >↩
::read ( )  [inline], [override], [virtual]
```

Waiting read by a tile program on a core input port from the switch.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 253 of file dma.hpp.

```
00253 { return fifo.value_pop().data; }
```

References trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::fifo.

12.70.4.3 receive() [1/2]

```
template<typename AXIStreamSwitch >
void trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::receive (
            std::span< axi_packet::value_type > sp )  [inline]
```

Enqueue a DMA transfer to receive a span.

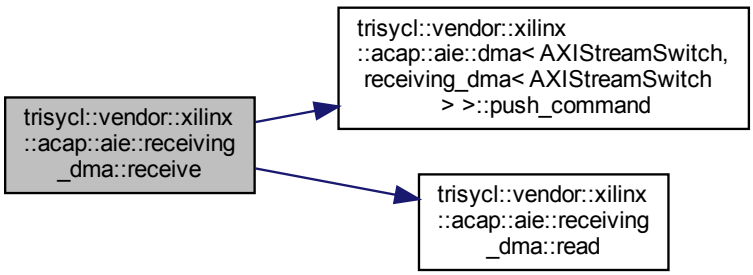
Use std::span for 1D contiguous transfers only for now

Definition at line 227 of file dma.hpp.

```
00227                                                 {
00228     this->push_command([=, this] {
00229       /* Write each element received from input router port into the
00230 memory described by DMA operation */
00231       for (auto& e :  sp) {
00232         e = read();
00233       }
00234     });
00235   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, receiving_dma< AXIStreamSwitch > >::push_command(), and trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::read().

Here is the call graph for this function:



12.70.4.4 receive() [2/2]

```
template<typename AXIStreamSwitch >
void trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::receive (
            std::span< axi_packet::value_type > sp )  [inline]
```

Enqueue a DMA transfer to receive a span.

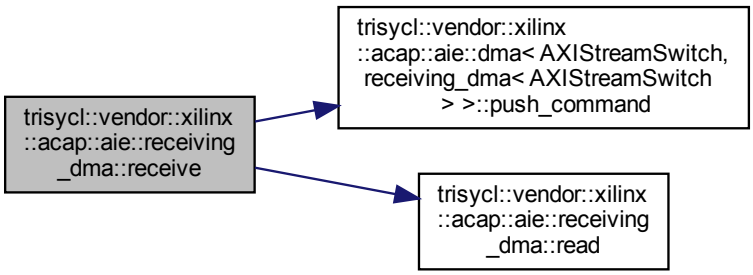
Use std::span for 1D contiguous transfers only for now

Definition at line 227 of file dma.hpp.

```
00227                                                {
00228     this->push_command([=, this] {
00229       /* Write each element received from input router port into the
00230 memory described by DMA operation */
00231       for (auto& e :  sp) {
00232         e = read();
00233       }
00234     });
00235   }
```

References trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, receiving_dma< AXIStreamSwitch > >::push_command(), and trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::read().

Here is the call graph for this function:



12.70.4.5 try_read() [1/2]

```
template<typename AXIStreamSwitch >
bool trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::try_read (
            axi_packet::value_type & v )  [inline], [override], [virtual]
```

Non-blocking read to a core input port.

Returns

true if the value was correctly read

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 259 of file dma.hpp.

```
00259                                                {
00260     axi_packet p;
00261
00262     if (fifo.try_pop(p) == boost::fibers::channel_op_status::success) {
00263       v = p.data;
00264       return true;
00265     }
00266     return false;
00267   }
```

References trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::fifo, and p.

12.70.4.6 try_read() [2/2]

```
template<typename AXIStreamSwitch >
bool trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::try_read (
            axi_packet::value_type & v )  [inline], [override], [virtual]
```

Non-blocking read to a core input port.

Returns

true if the value was correctly read

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 259 of file dma.hpp.

```
00259                                                {
00260     axi_packet p;
00261
00262     if (fifo.try_pop(p) == boost::fibers::channel_op_status::success) {
00263       v = p.data;
00264       return true;
00265     }
00266     return false;
00267   }
```

References trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::fifo, and p.

12.70.4.7 try_write() [1/2]

```
template<typename AXIStreamSwitch >
bool trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::try_write (
            const axi_packet & v )  [inline], [override], [virtual]
```

The network try to send some data to the DMA receiver input.

Returns

true if the packet is correctly enqueued or if the stream is closed, in order to to move on since we are in the tear-down phase of the infrastructure

Todo think about it again...

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 248 of file dma.hpp.

```
00248                                                {
00249     return this->fifo.try_push(v) != boost::fibers::channel_op_status::full;
00250   }
```

References trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::fifo.

12.70.4.8 try_write() [2/2]

```
template<typename AXIStreamSwitch >
bool trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::try_write (
            const axi_packet & v )  [inline], [override], [virtual]
```

The network try to send some data to the DMA receiver input.

Returns

true if the packet is correctly enqueued or if the stream is closed, in order to to move on since we are in the tear-down phase of the infrastructure

Todo think about it again...

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 248 of file dma.hpp.

```
00248                                           {
00249     return this->fifo.try_push(v) != boost::fibers::channel_op_status::full;
00250   }
```

References trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::fifo.

12.70.4.9 write() [1/2]

```
template<typename AXIStreamSwitch >
void trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::write (
            const axi_packet & v )  [inline], [override], [virtual]
```

The network sends some data to the DMA receiver input.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 238 of file dma.hpp.

```
00238 { fifo.push(v); }
```

References trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::fifo.

12.70.4.10 write() [2/2]

```
template<typename AXIStreamSwitch >
void trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::write (
            const axi_packet & v )  [inline], [override], [virtual]
```

The network sends some data to the DMA receiver input.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 238 of file dma.hpp.

```
00238 { fifo.push(v); }
```

References trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::fifo.

12.70.5 Member Data Documentation

12.70.5.1 axi_ss

```
template<typename AXIStreamSwitch >
AXIStreamSwitch & trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::axi_ss
[private]
```

Keep track of the AXI stream switch owning this port for debugging.

Definition at line 215 of file dma.hpp.

12.70.5.2 fifo

```
template<typename AXIStreamSwitch >
boost::fibers::buffered_channel< axi_packet > trisycl::vendor::xilinx::acap::aie::receiving_dma<
AXIStreamSwitch >::fifo { 8 }  [private]
```

Definition at line 212 of file dma.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::read(), trisycl::vendor::xilinx::acap::aie::receivin
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::try_write(), and trisycl::vendor::xilinx::acap::aie::receiving_dma

The documentation for this class was generated from the following files:

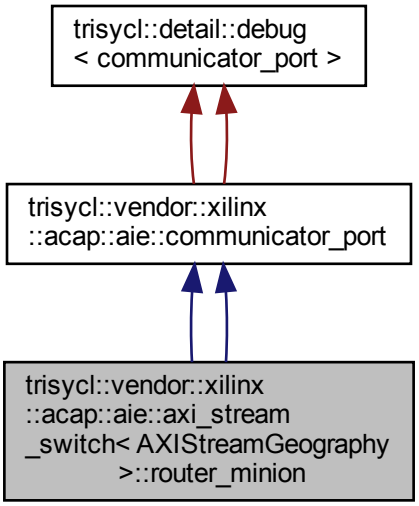
- include/sycl/vendor/Xilinx/acap/aie/dma.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/dma.hpp

12.71 trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion Class Reference

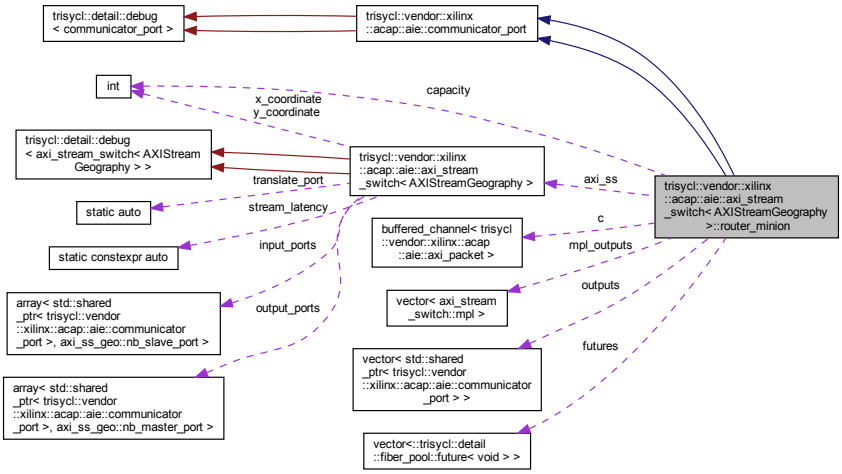
A router input port with routing skills.

```
#include <axi_stream_switch.hpp>
```

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion:



Public Types

- using value_type = axi_packet::value_type
- using value_type = axi_packet::value_type

Public Member Functions

- void join ()

  Wait for the fibers to complete.
- void write (const axi_packet &v) override

  Enqueue a packet on the router input.
- bool try_write (const axi_packet &v) override

  Try to enqueue a packet on the router input.
- value_type read () override

  Waiting read to a core input port.
- bool try_read (value_type &v) override

  Non-blocking read to a core input port.
- void connect_to (axi_stream_switch::mpl port_dest, std::shared_ptr< communicator_port > dest)

  Connect this routing input to a switch output.
- const auto & get_master_port_dests ()

  Return a vector of the master ports this router is forwarding.
- router_minion (axi_stream_switch &axi_ss, ::trisycl::detail::fiber_pool &fiber_executor, int capacity=axi_ss↩ _geo::latency)

  Create a router minion on this switch using a fiber executor.
- ∼router_minion () override

  Destructor handling the correct infrastructure shutdown.
- void join ()

  Wait for the fibers to complete.
- void write (const axi_packet &v) override

  Enqueue a packet on the router input.
- bool try_write (const axi_packet &v) override

  Try to enqueue a packet on the router input.
- value_type read () override

  Waiting read to a core input port.
- bool try_read (value_type &v) override

  Non-blocking read to a core input port.
- void connect_to (axi_stream_switch::mpl port_dest, std::shared_ptr< communicator_port > dest)

  Connect this routing input to a switch output.
- const auto & get_master_port_dests ()

  Return a vector of the master ports this router is forwarding.
- router_minion (axi_stream_switch &axi_ss, ::trisycl::detail::fiber_pool &fiber_executor, int capacity=axi_ss↩ _geo::latency)

  Create a router minion on this switch using a fiber executor.
- ∼router_minion () override

  Destructor handling the correct infrastructure shutdown.

Public Attributes

- const int capacity

  Router ingress capacity queue.
- boost::fibers::buffered_channel< axi_packet > c { static_cast<std::size_t>(capacity) }

  Ingress packet queue.
- std::vector< axi_stream_switch::mpl > mpl_outputs

  The ids of the outputs port the router minion has to forward to. Used by introspection to track current routing configuration.
- std::vector< std::shared_ptr< communicator_port > > outputs

The outputs the router minion has to forward to.

- std::vector<::trisycl::detail::fiber_pool::future< void > > futures

  To shepherd the routing fibers.

- axi_stream_switch & axi_ss

  Keep track of the AXI stream switch owning this port for debugging.

12.71.1 Detailed Description

template<typename AXIStreamGeography>
class trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion

A router input port with routing skills.

Definition at line 61 of file axi_stream_switch.hpp.

12.71.2 Member Typedef Documentation

12.71.2.1 value_type [1/2]

```
template<typename AXIStreamGeography >
using trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_↩
minion::value_type = axi_packet::value_type
```

Definition at line 95 of file axi_stream_switch.hpp.

12.71.2.2 value_type [2/2]

```
template<typename AXIStreamGeography >
using trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_↩
minion::value_type = axi_packet::value_type
```

Definition at line 95 of file axi_stream_switch.hpp.

12.71.3 Constructor & Destructor Documentation

12.71.3.1 router_minion() [1/2]

```
template<typename AXIStreamGeography >
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion↩
::router_minion (
            axi_stream_switch & axi_ss,
            ::trisycl::detail::fiber_pool & fiber_executor,
            int capacity = axi_ss_geo::latency )  [inline]
```

Create a router minion on this switch using a fiber executor.

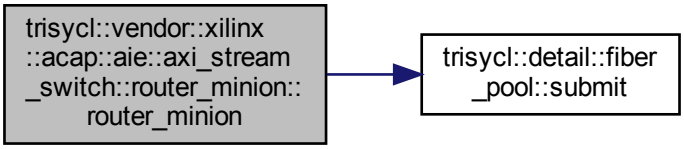
Use a fiber as a data mover from the input queue

Definition at line 166 of file axi_stream_switch.hpp.

```
00169       :  capacity { capacity }
00170       , axi_ss { axi_ss }
00171     {
00172 /// Use a fiber as a data mover from the input queue
00173       futures.push_back(fiber_executor.submit([&] {
00174         TRISYCL_DUMP_T("router_minion " « this « " on tile("
00175                        « router_minion::axi_ss.x_coordinate
00176                        « ',' « router_minion::axi_ss.y_coordinate
00177                        « ") on fiber " « boost::this_fiber::get_id()
00178                        « " starting with buffered_channel " « &c);
00179           for (;;) {
00180             TRISYCL_DUMP_T("router_minion " « this « " on tile("
00181                            « router_minion::axi_ss.x_coordinate
00182                            « ',' « router_minion::axi_ss.y_coordinate
00183                            « ") on fiber " « boost::this_fiber::get_id()
00184                            « " reading from buffered_channel "
00185                            « &c « " ...");
00186             auto v = c.value_pop();
00187             if (v.shutdown_request)
00188               // End the execution on shutdown packet reception
00189               break;
00190             TRISYCL_DUMP_T("router_minion " « this « " on tile("
00191                            « router_minion::axi_ss.x_coordinate
00192                            « ',' « router_minion::axi_ss.y_coordinate
00193                            « ") on fiber " « boost::this_fiber::get_id()
00194                            « " routing data value " « v.data
00195                            « " received from buffered_channel " « &c);
00196             // The routing itself is a blocking write on each output
00197             for (auto &o :  outputs) {
00198               TRISYCL_DUMP_T("router_minion " « this « " on tile("
00199                              « router_minion::axi_ss.x_coordinate
00200                              « ',' « router_minion::axi_ss.y_coordinate
00201                              « ") on fiber " « boost::this_fiber::get_id()
00202                              « " forwarding to router_minion " « o.get());
00203                o->write(v);
00204             }
00205           }
00206           TRISYCL_DUMP_T("router_minion " « this « " on tile("
00207                          « router_minion::axi_ss.x_coordinate
00208                          « ',' « router_minion::axi_ss.y_coordinate
00209                          « ") on fiber " « boost::this_fiber::get_id()
00210                          « " shutting down");
00211       }));
00212     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::futures, trisycl::detail::fiber_pool::subm TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::x_coordinate, and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::y_coordinate.

Here is the call graph for this function:



12.71.3.2 ~router_minion() [1/2]

```
template<typename AXIStreamGeography >
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion↩
::~router_minion ( )  [inline], [override]
```

Destructor handling the correct infrastructure shutdown.

Definition at line 216 of file axi_stream_switch.hpp.

```
00216                                {
00217       // Send a special packet to shutdown the routing process
00218       write(axi_packet::shutdown);
00219       // Wait for shutdown to avoid calling std::terminate on destruction...
00220       join();
00221     }
```

References trisycl::vendor::xilinx::acap::aie::axi_packet::shutdown.

12.71.3.3 router_minion() [2/2]

```
template<typename AXIStreamGeography >
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion↩
::router_minion (
              axi_stream_switch & axi_ss,
              ::trisycl::detail::fiber_pool & fiber_executor,
              int capacity = axi_ss_geo::latency )  [inline]
```

Create a router minion on this switch using a fiber executor.

Use a fiber as a data mover from the input queue

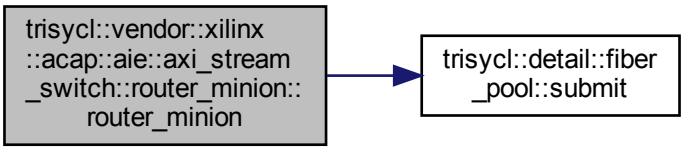
Definition at line 166 of file axi_stream_switch.hpp.

```
00169       :  capacity { capacity }
00170       , axi_ss { axi_ss }
00171     {
00172 /// Use a fiber as a data mover from the input queue
00173       futures.push_back(fiber_executor.submit([&] {
00174         TRISYCL_DUMP_T("router_minion " « this « " on tile("
00175                        « router_minion::axi_ss.x_coordinate
00176                        « ',' « router_minion::axi_ss.y_coordinate
```

```
00177                       « ") on fiber " « boost::this_fiber::get_id()
00178                       « " starting with buffered_channel " « &c);
00179           for (;;) {
00180             TRISYCL_DUMP_T("router_minion " « this « " on tile("
00181                            « router_minion::axi_ss.x_coordinate
00182                            « ',' « router_minion::axi_ss.y_coordinate
00183                            « ") on fiber " « boost::this_fiber::get_id()
00184                            « " reading from buffered_channel "
00185                            « &c « " ...");
00186             auto v = c.value_pop();
00187             if (v.shutdown_request)
00188               // End the execution on shutdown packet reception
00189               break;
00190             TRISYCL_DUMP_T("router_minion " « this « " on tile("
00191                            « router_minion::axi_ss.x_coordinate
00192                            « ',' « router_minion::axi_ss.y_coordinate
00193                            « ") on fiber " « boost::this_fiber::get_id()
00194                            « " routing data value " « v.data
00195                            « " received from buffered_channel " « &c);
00196             // The routing itself is a blocking write on each output
00197             for (auto &o :  outputs) {
00198               TRISYCL_DUMP_T("router_minion " « this « " on tile("
00199                              « router_minion::axi_ss.x_coordinate
00200                              « ',' « router_minion::axi_ss.y_coordinate
00201                              « ") on fiber " « boost::this_fiber::get_id()
00202                              « " forwarding to router_minion " « o.get());
00203                 o->write(v);
00204             }
00205           }
00206           TRISYCL_DUMP_T("router_minion " « this « " on tile("
00207                          « router_minion::axi_ss.x_coordinate
00208                          « ',' « router_minion::axi_ss.y_coordinate
00209                          « ") on fiber " « boost::this_fiber::get_id()
00210                          « " shutting down");
00211       }));
00212     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::futures, trisycl::detail::fiber_pool::subm TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::x_coordinate, and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::y_coordinate.

Here is the call graph for this function:



12.71.3.4 ~router_minion() [2/2]

```
template<typename AXIStreamGeography >
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion↩
::~router_minion ( )  [inline], [override]
```

Destructor handling the correct infrastructure shutdown.

Definition at line 216 of file axi_stream_switch.hpp.

```
00216                               {
00217       // Send a special packet to shutdown the routing process
00218       write(axi_packet::shutdown);
00219       // Wait for shutdown to avoid calling std::terminate on destruction...
00220       join();
00221     }
```

References trisycl::vendor::xilinx::acap::aie::axi_packet::shutdown.

12.71.4 Member Function Documentation

12.71.4.1 connect_to() [1/2]

```
template<typename AXIStreamGeography >
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_↩
minion::connect_to (
              axi_stream_switch::mpl port_dest,
              std::shared_ptr< communicator_port > dest )  [inline]
```

Connect this routing input to a switch output.

Definition at line 147 of file axi_stream_switch.hpp.

```
00148                                                     {
00149       TRISYCL_DUMP_T("connect_to router_minion " « this « " on tile("
00150                      « router_minion::axi_ss.x_coordinate
00151                      « ',' « router_minion::axi_ss.y_coordinate
00152                      « ") on fiber " « boost::this_fiber::get_id()
00153                      « " to dest " « &*dest);
00154       mpl_outputs.push_back(port_dest);
00155       outputs.push_back(dest);
00156     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::mpl_outputs, trisycl::vendor::xilinx::aca TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::x_coordinate, and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::y_coordinate.

12.71.4.2 connect_to() [2/2]

```
template<typename AXIStreamGeography >
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_↩
minion::connect_to (
              axi_stream_switch::mpl port_dest,
              std::shared_ptr< communicator_port > dest )  [inline]
```

Connect this routing input to a switch output.

Definition at line 147 of file axi_stream_switch.hpp.

```
00148                                                     {
00149       TRISYCL_DUMP_T("connect_to router_minion " « this « " on tile("
00150                      « router_minion::axi_ss.x_coordinate
00151                      « ',' « router_minion::axi_ss.y_coordinate
00152                      « ") on fiber " « boost::this_fiber::get_id()
00153                      « " to dest " « &*dest);
00154       mpl_outputs.push_back(port_dest);
00155       outputs.push_back(dest);
00156     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::mpl_outputs, trisycl::vendor::xilinx::aca TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::x_coordinate, and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::y_coordinate.

12.71.4.3 get_master_port_dests() [1/2]

```
template<typename AXIStreamGeography >
const auto & trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >↩
::router_minion::get_master_port_dests ( )  [inline]
```

Return a vector of the master ports this router is forwarding.

Definition at line 160 of file axi_stream_switch.hpp.

```
00160                                       {
00161       return mpl_outputs;
00162     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::mpl_outputs.

12.71.4.4 get_master_port_dests() [2/2]

```
template<typename AXIStreamGeography >
const auto & trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >↩
::router_minion::get_master_port_dests ( )  [inline]
```

Return a vector of the master ports this router is forwarding.

Definition at line 160 of file axi_stream_switch.hpp.

```
00160                                       {
00161       return mpl_outputs;
00162     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::mpl_outputs.

12.71.4.5 join() [1/2]

```
template<typename AXIStreamGeography >
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_↩
minion::join ( )  [inline]
```

Wait for the fibers to complete.

Definition at line 86 of file axi_stream_switch.hpp.

```
00086                 {
00087       // Wait on all the fibers to finish
00088       for (auto &f :  futures)
00089         // Get the value of the future, to get an exception if any
00090         f.get();
00091     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::futures.

12.71.4.6 join() [2/2]

```
template<typename AXIStreamGeography >
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_↩
minion::join ( )  [inline]
```

Wait for the fibers to complete.

Definition at line 86 of file axi_stream_switch.hpp.

```
00086                 {
00087       // Wait on all the fibers to finish
00088       for (auto &f :  futures)
00089         // Get the value of the future, to get an exception if any
00090         f.get();
00091     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::futures.

12.71.4.7 read() [1/2]

```
template<typename AXIStreamGeography >
value_type trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router↩
_minion::read ( )  [inline], [override], [virtual]
```

Waiting read to a core input port.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 121 of file axi_stream_switch.hpp.

```
00121                               {
00122       TRISYCL_DUMP_T("router_minion " « this « " on tile("
00123                      « router_minion::axi_ss.x_coordinate
00124                      « ',' « router_minion::axi_ss.y_coordinate
00125                      « ") on fiber " « boost::this_fiber::get_id()
00126                      « " reading from buffered_channel " « &c « " ...");
00127       return c.value_pop().data;
00128     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::c, TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::x_coordinate, and trisycl::vendor::xilinx::acap::aie::axi

12.71.4.8 read() [2/2]

```
template<typename AXIStreamGeography >
value_type trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router↩
_minion::read ( )  [inline], [override], [virtual]
```

Waiting read to a core input port.

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 121 of file axi_stream_switch.hpp.

```
00121                               {
00122       TRISYCL_DUMP_T("router_minion " « this « " on tile("
00123                      « router_minion::axi_ss.x_coordinate
00124                      « ',' « router_minion::axi_ss.y_coordinate
00125                      « ") on fiber " « boost::this_fiber::get_id()
00126                      « " reading from buffered_channel " « &c « " ...");
00127       return c.value_pop().data;
00128     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::c, TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::x_coordinate, and trisycl::vendor::xilinx::acap::aie::axi

12.71.4.9 try_read() [1/2]

```
template<typename AXIStreamGeography >
bool trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_↩
minion::try_read (
            value_type & v )  [inline], [override], [virtual]
```

Non-blocking read to a core input port.

Returns

true if the value was correctly read

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 135 of file axi_stream_switch.hpp.

```
00135                                       {
00136       axi_packet p;
00137
00138       if (c.try_pop(p) == boost::fibers::channel_op_status::success) {
00139         v = p.data;
00140         return true;
00141       }
00142       return false;
00143     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::c, and p.

12.71.4.10 try_read() [2/2]

```
template<typename AXIStreamGeography >
bool trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_↩
minion::try_read (
            value_type & v )  [inline], [override], [virtual]
```

Non-blocking read to a core input port.

Returns

true if the value was correctly read

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 135 of file axi_stream_switch.hpp.

```
00135                                       {
00136       axi_packet p;
00137
00138       if (c.try_pop(p) == boost::fibers::channel_op_status::success) {
00139         v = p.data;
00140         return true;
00141       }
00142       return false;
00143     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::c, and p.

12.71.4.11 try_write() [1/2]

```
template<typename AXIStreamGeography >
bool trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_↩
minion::try_write (
            const axi_packet & v )  [inline], [override], [virtual]
```

Try to enqueue a packet on the router input.

Returns

true if the packet is correctly enqueued

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 115 of file axi_stream_switch.hpp.

```
00115                                       {
00116       return c.try_push(v) != boost::fibers::channel_op_status::full;
00117     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::c.

12.71.4.12 try_write() [2/2]

```
template<typename AXIStreamGeography >
bool trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_↩
minion::try_write (
            const axi_packet & v )  [inline], [override], [virtual]
```

Try to enqueue a packet on the router input.

Returns

true if the packet is correctly enqueued

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 115 of file axi_stream_switch.hpp.

```
00115                                       {
00116       return c.try_push(v) != boost::fibers::channel_op_status::full;
00117     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::c.

12.71.4.13 write() [1/2]

```
template<typename AXIStreamGeography >
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_↩
minion::write (
            const axi_packet & v )  [inline], [override], [virtual]
```

Enqueue a packet on the router input.

Todo Clean up the API to separate data from signaling

Todo separate debug from shutdown case

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 99 of file axi_stream_switch.hpp.

```
00099                                           {
00100 /// \todo separate debug from shutdown case
00101       TRISYCL_DUMP_T("router_minion " « this
00102                      « " on tile(" « router_minion::axi_ss.x_coordinate « ','
00103                      « router_minion::axi_ss.y_coordinate
00104                      « ") on fiber " « boost::this_fiber::get_id()
00105                      « " write data value " « v.data
00106                      « " to buffered_channel " « &c);
00107       c.push(v);
00108     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::c, trisycl::vendor::xilinx::acap::aie::axi_
TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::x_coordinate, and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::y_coordinate.

12.71.4.14 write() [2/2]

```
template<typename AXIStreamGeography >
void trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_↩
minion::write (
            const axi_packet & v )  [inline], [override], [virtual]
```

Enqueue a packet on the router input.

Todo Clean up the API to separate data from signaling

Todo separate debug from shutdown case

Implements trisycl::vendor::xilinx::acap::aie::communicator_port.

Definition at line 99 of file axi_stream_switch.hpp.

```
00099                                           {
00100 /// \todo separate debug from shutdown case
00101       TRISYCL_DUMP_T("router_minion " « this
00102                      « " on tile(" « router_minion::axi_ss.x_coordinate « ','
00103                      « router_minion::axi_ss.y_coordinate
00104                      « ") on fiber " « boost::this_fiber::get_id()
00105                      « " write data value " « v.data
00106                      « " to buffered_channel " « &c);
00107       c.push(v);
00108     }
```

References trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::axi_ss, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::c, trisycl::vendor::xilinx::acap::aie::axi_
TRISYCL_DUMP_T, trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::x_coordinate, and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::y_coordinate.

12.71.5 Member Data Documentation

12.71.5.1 axi_ss

```
template<typename AXIStreamGeography >
axi_stream_switch & trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography
>::router_minion::axi_ss
```

Keep track of the AXI stream switch owning this port for debugging.

Definition at line 83 of file axi_stream_switch.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::connect_to(), trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::read(), trisycl::vendor::xilinx::acap::aie:: and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::write().

12.71.5.2 c

```
template<typename AXIStreamGeography >
boost::fibers::buffered_channel< axi_packet > trisycl::vendor::xilinx::acap::aie::axi_stream_switch<
AXIStreamGeography >::router_minion::c { static_cast<std::size_t>(capacity) }
```

Ingress packet queue.

Definition at line 67 of file axi_stream_switch.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::read(), trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::try_read(), trisycl::vendor::xilinx::acap:: and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::write().

12.71.5.3 capacity

```
template<typename AXIStreamGeography >
const int trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router↩
_minion::capacity
```

Router ingress capacity queue.

Definition at line 64 of file axi_stream_switch.hpp.

12.71.5.4 futures

```
template<typename AXIStreamGeography >
std::vector<::trisycl::detail::fiber_pool::future< void > > trisycl::vendor::xilinx::acap::aie::axi_stream_s
AXIStreamGeography >::router_minion::futures
```

To shepherd the routing fibers.

Definition at line 80 of file axi_stream_switch.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::join(), and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::router_minion().

12.71.5.5 mpl_outputs

```
template<typename AXIStreamGeography >
std::vector< axi_stream_switch::mpl > trisycl::vendor::xilinx::acap::aie::axi_stream_switch<
AXIStreamGeography >::router_minion::mpl_outputs
```

The ids of the outputs port the router minion has to forward to. Used by introspection to track current routing configuration.

Definition at line 72 of file axi_stream_switch.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::connect_to(), and trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::get_master_port_dests().

12.71.5.6 outputs

```
template<typename AXIStreamGeography >
std::vector< std::shared_ptr< communicator_port > > trisycl::vendor::xilinx::acap::aie::axi_stream_switch<
AXIStreamGeography >::router_minion::outputs
```

The outputs the router minion has to forward to.

This what is used to speed up the routing itself even if the id could be enough

Definition at line 77 of file axi_stream_switch.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion::connect_to().

The documentation for this class was generated from the following files:

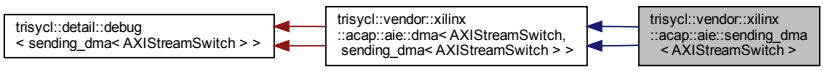
- include/sycl/vendor/Xilinx/acap/aie/axi_stream_switch.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/axi_stream_switch.hpp

12.72 trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch > Class Template Reference

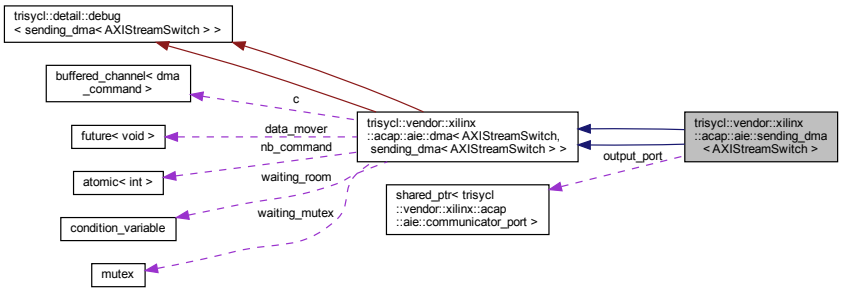
Sending DMA.

`#include <dma.hpp>`

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >:



Public Member Functions

- sending_dma (::trisycl::detail::fiber_pool &fe, std::shared_ptr< communicator_port > output)
  Start the DMA engine using an executor to push things on a port.
- void send (std::span< axi_packet::value_type > sp)
  Enqueue a DMA transfer to send a span.
- sending_dma (::trisycl::detail::fiber_pool &fe, std::shared_ptr< communicator_port > output)
  Start the DMA engine using an executor to push things on a port.
- void send (std::span< axi_packet::value_type > sp)
  Enqueue a DMA transfer to send a span.

Private Types

- using dma_base = dma< AXIStreamSwitch, sending_dma< AXIStreamSwitch > >
- using dma_base = dma< AXIStreamSwitch, sending_dma< AXIStreamSwitch > >

Private Attributes

- std::shared_ptr< communicator_port > output_port

Additional Inherited Members

12.72.1 Detailed Description

template<typename AXIStreamSwitch>
class trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >

Sending DMA.

Todo Factorize out the FIFO port

Todo Factorize out the FIFO port

Definition at line 275 of file dma.hpp.

12.72.2 Member Typedef Documentation

12.72.2.1 dma_base [1/2]

```
template<typename AXIStreamSwitch >
using trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >::dma_base = dma<AXIStream↩
Switch, sending_dma<AXIStreamSwitch> > [private]
```

Definition at line 276 of file dma.hpp.

12.72.2.2 dma_base [2/2]

```
template<typename AXIStreamSwitch >
using trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >::dma_base = dma<AXIStream↩
Switch, sending_dma<AXIStreamSwitch> > [private]
```

Definition at line 276 of file dma.hpp.

12.72.3 Constructor & Destructor Documentation

12.72.3.1 sending_dma() [1/2]

```
template<typename AXIStreamSwitch >
trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >::sending_dma (
            ::trisycl::detail::fiber_pool & fe,
            std::shared_ptr< communicator_port > output )  [inline]
```

Start the DMA engine using an executor to push things on a port.

Definition at line 282 of file dma.hpp.

```
00284       :  dma_base { fe }
00285       , output_port { output } {}
```

12.72.3.2 sending_dma() [2/2]

```
template<typename AXIStreamSwitch >
trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >::sending_dma (
            ::trisycl::detail::fiber_pool & fe,
            std::shared_ptr< communicator_port > output )  [inline]
```

Start the DMA engine using an executor to push things on a port.

Definition at line 282 of file dma.hpp.

```
00284       :  dma_base { fe }
00285       , output_port { output } {}
```

12.72.4 Member Function Documentation

12.72.4.1 send() [1/2]

```
template<typename AXIStreamSwitch >
void trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >::send (
            std::span< axi_packet::value_type > sp )  [inline]
```

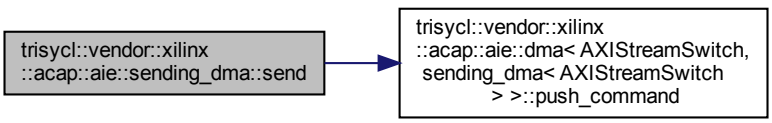
Enqueue a DMA transfer to send a span.

Definition at line 288 of file dma.hpp.

```
00288                                                {
00289     this->push_command([=, this] {
00290       // Send each element of this span to the output port
00291       for (const auto& e :  sp) {
00292         output_port->write(e);
00293       }
00294     });
00295   }
```

References trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >::output_port, and trisycl::vendor::xilinx::acap::aie::dr

Here is the call graph for this function:



12.72.4.2 send() [2/2]

```
template<typename AXIStreamSwitch >
void trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >::send (
            std::span< axi_packet::value_type > sp )  [inline]
```

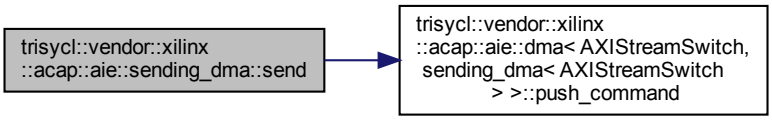
Enqueue a DMA transfer to send a span.

Definition at line 288 of file dma.hpp.

```
00288                                                {
00289     this->push_command([=, this] {
00290       // Send each element of this span to the output port
00291       for (const auto& e :  sp) {
00292         output_port->write(e);
00293       }
00294     });
00295   }
```

References trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >::output_port, and trisycl::vendor::xilinx::acap::aie::dr

Here is the call graph for this function:



12.72.5 Member Data Documentation

12.72.5.1 output_port

```
template<typename AXIStreamSwitch >
std::shared_ptr< communicator_port > trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStream↩
Switch >::output_port  [private]
```

Definition at line 278 of file dma.hpp.

Referenced by trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >::send().

The documentation for this class was generated from the following files:

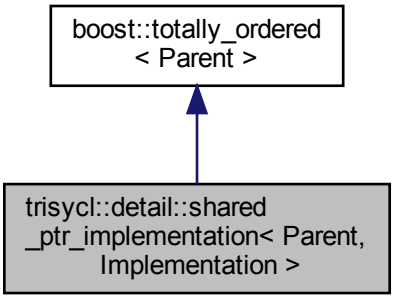
- include/sycl/vendor/Xilinx/acap/aie/dma.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/dma.hpp

12.73 trisycl::detail::shared_ptr_implementation< Parent, Implementation > Struct Template Reference

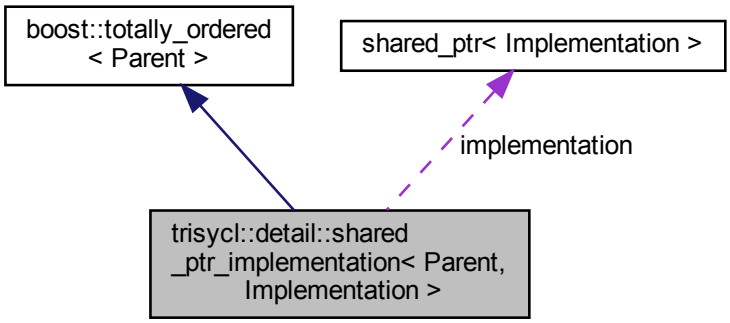
Provide an implementation as shared_ptr with total ordering and hashing to be used with algorithms and in (un)ordered containers.

`#include <shared_ptr_implementation.hpp>`

Inheritance diagram for trisycl::detail::shared_ptr_implementation< Parent, Implementation >:



Collaboration diagram for trisycl::detail::shared_ptr_implementation< Parent, Implementation >:



Public Member Functions

- shared_ptr_implementation (std::shared_ptr< Implementation > i)

  The implementation directly as a shared pointer.
- shared_ptr_implementation (Implementation *i)

  The implementation takes the ownership from a raw pointer.
- shared_ptr_implementation ()=default

  Keep all other constructors to have usual shared_ptr behaviour.
- auto hash () const

  Forward the hashing for unordered containers to the implementation.

Public Attributes

- std::shared_ptr< Implementation > implementation

  The implementation forward everything to this... implementation.

Friends

- bool operator== (const Parent &lhs, const Parent &rhs)

  Equality operator.
- bool operator< (const Parent &lhs, const Parent &rhs)

  Inferior operator.

12.73.1 Detailed Description

template<typename Parent, typename Implementation>
struct trisycl::detail::shared_ptr_implementation< Parent, Implementation >

Provide an implementation as shared_ptr with total ordering and hashing to be used with algorithms and in (un)ordered containers.

To be used, a Parent class wanting an Implementation needs to inherit from.

The implementation ends up in a member really named "implementation".
`public detail::shared_ptr_implementation<Parent, Implementation>`

and also inject in std namespace a specialization for
`hash<Parent>`

Definition at line 38 of file shared_ptr_implementation.hpp.

12.73.2 Constructor & Destructor Documentation

12.73.2.1 shared_ptr_implementation() [1/3]

```
template<typename Parent , typename Implementation >
trisycl::detail::shared_ptr_implementation< Parent, Implementation >::shared_ptr_implementation
(
            std::shared_ptr< Implementation > i )  [inline]
```

The implementation directly as a shared pointer.

Definition at line 44 of file shared_ptr_implementation.hpp.
```
00045     : implementation { i } {}
```

12.73.2.2 shared_ptr_implementation() [2/3]

```
template<typename Parent , typename Implementation >
trisycl::detail::shared_ptr_implementation< Parent, Implementation >::shared_ptr_implementation
(
            Implementation * i )  [inline]
```

The implementation takes the ownership from a raw pointer.

Definition at line 49 of file shared_ptr_implementation.hpp.

```
00049 :  implementation { i } {}
```

12.73.2.3 shared_ptr_implementation() [3/3]

```
template<typename Parent , typename Implementation >
trisycl::detail::shared_ptr_implementation< Parent, Implementation >::shared_ptr_implementation
( )  [default]
```

Keep all other constructors to have usual shared_ptr behaviour.

12.73.3 Member Function Documentation

12.73.3.1 hash()

```
template<typename Parent , typename Implementation >
auto trisycl::detail::shared_ptr_implementation< Parent, Implementation >::hash ( ) const
[inline]
```

Forward the hashing for unordered containers to the implementation.

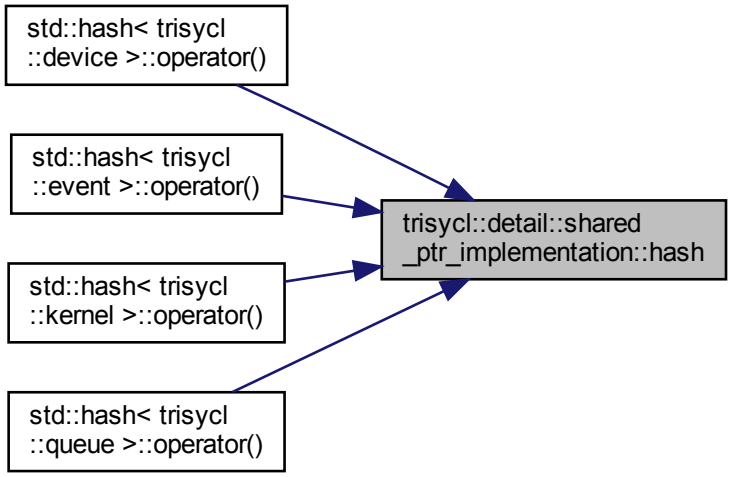
Definition at line 87 of file shared_ptr_implementation.hpp.

```
00087                     {
00088     return std::hash<decltype(implementation)>{}(implementation);
00089   }
```

References trisycl::detail::shared_ptr_implementation< Parent, Implementation >::implementation.

Referenced by std::hash< trisycl::device >::operator()(), std::hash< trisycl::event >::operator()(), std::hash< trisycl::kernel >::operat
and std::hash< trisycl::queue >::operator()().

Here is the caller graph for this function:



12.73.4 Friends And Related Function Documentation

12.73.4.1 operator<

```
template<typename Parent , typename Implementation >
bool operator< (
             const Parent & lhs,
             const Parent & rhs )  [friend]
```

Inferior operator.

This is generalized by boost::less_than_comparable from boost::totally_ordered to implement the equality comparable concept

Use recipe for hidden friends from P1601R0: "Recommendations for Specifying “Hidden Friends”"

Todo Add this to the spec

Definition at line 81 of file shared_ptr_implementation.hpp.

```
00081                                                                    {
00082     return lhs.implementation < rhs.implementation;
00083   }
```

12.73.4.2 operator==

```
template<typename Parent , typename Implementation >
bool operator== (
             const Parent & lhs,
             const Parent & rhs )  [friend]
```

Equality operator.

This is generalized by boost::equality_comparable from boost::totally_ordered to implement the equality comparable concept

Use recipe for hidden friends from P1601R0: "Recommendations for Specifying “Hidden Friends""

Definition at line 65 of file shared_ptr_implementation.hpp.

```
00065                                                                    {
00066     return lhs.implementation == rhs.implementation;
00067   }
```

12.73.5 Member Data Documentation

12.73.5.1 implementation

```
template<typename Parent , typename Implementation >
std::shared_ptr<Implementation> trisycl::detail::shared_ptr_implementation< Parent, Implementation
>::implementation
```

The implementation forward everything to this... implementation.

Definition at line 41 of file shared_ptr_implementation.hpp.

Referenced by trisycl::detail::shared_ptr_implementation< Parent, Implementation >::hash(), trisycl::handler::parallel_for(), and trisycl::handler::single_task().

The documentation for this struct was generated from the following file:

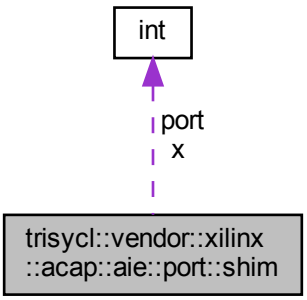
- include/triSYCL/detail/shared_ptr_implementation.hpp

12.74 trisycl::vendor::xilinx::acap::aie::port::shim Struct Reference

Abstraction of a communication port inside a shim tile.

`#include <connection.hpp>`

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::port::shim:



Public Attributes

- const int x
- const int port

12.74.1 Detailed Description

Abstraction of a communication port inside a shim tile.

Definition at line 247 of file connection.hpp.

12.74.2 Member Data Documentation

12.74.2.1 port

`const int trisycl::vendor::xilinx::acap::aie::port::shim::port`

Definition at line 249 of file connection.hpp.

12.74.2.2 x

```
const int trisycl::vendor::xilinx::acap::aie::port::shim::x
```

Definition at line 248 of file connection.hpp.

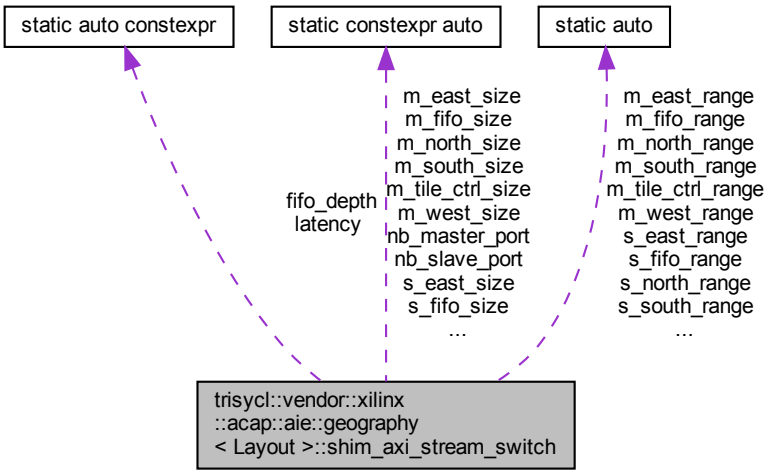
The documentation for this struct was generated from the following files:

- include/sycl/vendor/Xilinx/acap/aie/connection.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/connection.hpp

12.75 trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch Struct Reference

```
#include <geography.hpp>
```

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch:



Public Types

- enum class master_port_layout : std::int8_t {
  tile_ctrl , tile_ctrl_last = tile_ctrl , fifo_0 , fifo_1 ,
  fifo_last = fifo_1 , south_0 , south_1 , south_2 ,
  south_3 , south_4 , south_5 , south_last = south_5 ,
  west_0 , west_1 , west_2 , west_3 ,
  west_last = west_3 , north_0 , north_1 , north_2 ,
  north_3 , north_4 , north_5 , north_last = north_5 ,
  east_0 , east_1 , east_2 , east_3 ,
  east_last = east_3 , size , tile_ctrl , tile_ctrl_last = tile_ctrl ,
  fifo_0 , fifo_1 , fifo_last = fifo_1 , south_0 ,
  south_1 , south_2 , south_3 , south_4 ,

south_5 , south_last = south_5 , west_0 , west_1 ,
west_2 , west_3 , west_last = west_3 , north_0 ,
north_1 , north_2 , north_3 , north_4 ,
north_5 , north_last = north_5 , east_0 , east_1 ,
east_2 , east_3 , east_last = east_3 , size }

Layout of the AXI stream master ports in the switch.

- enum class slave_port_layout : std::int8_t {
tile_ctrl , tile_ctrl_last = tile_ctrl , fifo_0 , fifo_1 ,
fifo_last = fifo_1 , south_0 , south_1 , south_2 ,
south_3 , south_4 , south_5 , south_6 ,
south_7 , south_last = south_7 , west_0 , west_1 ,
west_2 , west_3 , west_last = west_3 , north_0 ,
north_1 , north_2 , north_3 , north_last = north_3 ,
east_0 , east_1 , east_2 , east_3 ,
east_last = east_3 , shim_trace , size , tile_ctrl ,
tile_ctrl_last = tile_ctrl , fifo_0 , fifo_1 , fifo_last = fifo_1 ,
south_0 , south_1 , south_2 , south_3 ,
south_4 , south_5 , south_6 , south_7 ,
south_last = south_7 , west_0 , west_1 , west_2 ,
west_3 , west_last = west_3 , north_0 , north_1 ,
north_2 , north_3 , north_last = north_3 , east_0 ,
east_1 , east_2 , east_3 , east_last = east_3 ,
shim_trace , size }

Layout of the AXI stream slave ports in the switch.

- enum class master_port_layout : std::int8_t {
tile_ctrl , tile_ctrl_last = tile_ctrl , fifo_0 , fifo_1 ,
fifo_last = fifo_1 , south_0 , south_1 , south_2 ,
south_3 , south_4 , south_5 , south_last = south_5 ,
west_0 , west_1 , west_2 , west_3 ,
west_last = west_3 , north_0 , north_1 , north_2 ,
north_3 , north_4 , north_5 , north_last = north_5 ,
east_0 , east_1 , east_2 , east_3 ,
east_last = east_3 , size , tile_ctrl , tile_ctrl_last = tile_ctrl ,
fifo_0 , fifo_1 , fifo_last = fifo_1 , south_0 ,
south_1 , south_2 , south_3 , south_4 ,
south_5 , south_last = south_5 , west_0 , west_1 ,
west_2 , west_3 , west_last = west_3 , north_0 ,
north_1 , north_2 , north_3 , north_4 ,
north_5 , north_last = north_5 , east_0 , east_1 ,
east_2 , east_3 , east_last = east_3 , size }

Layout of the AXI stream master ports in the switch.

- enum class slave_port_layout : std::int8_t {
tile_ctrl , tile_ctrl_last = tile_ctrl , fifo_0 , fifo_1 ,
fifo_last = fifo_1 , south_0 , south_1 , south_2 ,
south_3 , south_4 , south_5 , south_6 ,
south_7 , south_last = south_7 , west_0 , west_1 ,
west_2 , west_3 , west_last = west_3 , north_0 ,
north_1 , north_2 , north_3 , north_last = north_3 ,
east_0 , east_1 , east_2 , east_3 ,
east_last = east_3 , shim_trace , size , tile_ctrl ,
tile_ctrl_last = tile_ctrl , fifo_0 , fifo_1 , fifo_last = fifo_1 ,
south_0 , south_1 , south_2 , south_3 ,
south_4 , south_5 , south_6 , south_7 ,
south_last = south_7 , west_0 , west_1 , west_2 ,
west_3 , west_last = west_3 , north_0 , north_1 ,
north_2 , north_3 , north_last = north_3 , east_0 ,
east_1 , east_2 , east_3 , east_last = east_3 ,
shim_trace , size }

Layout of the AXI stream slave ports in the switch.

Static Public Attributes

- static auto constexpr latency = 4
- static auto constexpr fifo_depth = 16

  Depth of additional FIFO on specific FIFO paths.
- static constexpr auto nb_master_port

  Number of AXI stream master ports in the switch.
- static auto m_tile_ctrl_range

  A range of the master ports connected to the tile control inputs.
- static constexpr auto m_tile_ctrl_size

  Number of master ports connected to the tile control inputs.
- static auto m_fifo_range

  A range of the master ports connected to the FIFO inputs.
- static constexpr auto m_fifo_size

  Number of master ports connected to the FIFO inputs.
- static auto m_south_range

  A range of the South master ports.
- static constexpr auto m_south_size

  Number of South master ports.
- static auto m_west_range

  A range of the West master ports.
- static constexpr auto m_west_size

  Number of West master ports.
- static auto m_north_range

  A range of the North master ports.
- static constexpr auto m_north_size

  Number of North master ports.
- static auto m_east_range

  A range of the East master ports.
- static constexpr auto m_east_size

  Number of East master ports.
- static constexpr auto nb_slave_port

  Number of master AXI stream slave ports in the switch.
- static auto s_tile_ctrl_range

  A range of the slave ports connected to the tile control outputs.
- static constexpr auto s_tile_ctrl_size

  Number of slave ports connected to the tile control outputs.
- static auto s_fifo_range

  A range of the slave ports connected to the FIFO outputs.
- static constexpr auto s_fifo_size

  Number of slave ports connected to the FIFO outputs.
- static auto s_south_range

  A range of the South slave ports.
- static constexpr auto s_south_size

  Number of South slave ports.
- static auto s_west_range

  A range of the West slave ports.
- static constexpr auto s_west_size

Number of West slave ports.

- static auto s_north_range

  A range of the North slave ports.

- static constexpr auto s_north_size

  Number of North slave ports.

- static auto s_east_range

  A range of the East slave ports.

- static constexpr auto s_east_size

  Number of East slave ports.

Friends

- bool constexpr is_axi_master (master_port_layout)
- bool constexpr is_axi_master (slave_port_layout)

  Use a hidden friend to have introspection for slave_port_layout.

- bool constexpr is_axi_master (master_port_layout)
- bool constexpr is_axi_master (slave_port_layout)

  Use a hidden friend to have introspection for slave_port_layout.

12.75.1 Detailed Description

template<typename Layout>
struct trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch

Definition at line 636 of file geography.hpp.

12.75.2 Member Enumeration Documentation

12.75.2.1 master_port_layout [1/2]

```
template<typename Layout >
enum class trisycl::vendor::xilinx::acap::aie::geography::shim_axi_stream_switch::master_port_layout
:  std::int8_t  [strong]
```

Layout of the AXI stream master ports in the switch.

Revision v2.02, 6.5.3 Shim AXI-Stream Interconnect, Table 6-2, p. 291

Enumerator

tile_ctrl	
tile_ctrl_last	Used for validation,.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	

Enumerator

south_1	
south_2	
south_3	
south_4	
south_5	
south_last	Used for shim BLI port validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_4	
north_5	
north_last	Used for north validation.
east_0	
east_1	
east_2	
east_3	
east_last	Used for east validation.
size	
tile_ctrl	
tile_ctrl_last	Used for validation,.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	
south_4	
south_5	
south_last	Used for shim BLI port validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_4	
north_5	
north_last	Used for north validation.
east_0	
east_1	

Enumerator

east_2	
east_3	
east_last	Used for east validation.
size	

Definition at line 647 of file geography.hpp.

```
00647                                    :  std::int8_t {
00648       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation,
00649       fifo_0,
00650       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00651       south_0,
00652       south_1,
00653       south_2,
00654       south_3,
00655       south_4,
00656       south_5, south_last = south_5, ///< Used for shim BLI port validation
00657       west_0,
00658       west_1,
00659       west_2,
00660       west_3, west_last = west_3, ///< Used for west validation
00661       north_0,
00662       north_1,
00663       north_2,
00664       north_3,
00665       north_4,
00666       north_5, north_last = north_5, ///< Used for north validation
00667       east_0,
00668       east_1,
00669       east_2,
00670       east_3, east_last = east_3, ///< Used for east validation
00671       // To measure the enum
00672       size
00673     };
```

12.75.2.2 master_port_layout [2/2]

```
template<typename Layout >
enum class trisycl::vendor::xilinx::acap::aie::geography::shim_axi_stream_switch::master_port_layout
:  std::int8_t  [strong]
```

Layout of the AXI stream master ports in the switch.

Revision v2.02, 6.5.3 Shim AXI-Stream Interconnect, Table 6-2, p. 291

Enumerator

tile_ctrl	
tile_ctrl_last	Used for validation,.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	
south_4	
south_5	
south_last	Used for shim BLI port validation.
west_0	

Enumerator

west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_4	
north_5	
north_last	Used for north validation.
east_0	
east_1	
east_2	
east_3	
east_last	Used for east validation.
size	
tile_ctrl	
tile_ctrl_last	Used for validation,.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	
south_4	
south_5	
south_last	Used for shim BLI port validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_4	
north_5	
north_last	Used for north validation.
east_0	
east_1	
east_2	
east_3	
east_last	Used for east validation.
size	

Definition at line 647 of file geography.hpp.

```
00647                                    :  std::int8_t {
00648       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation,
00649       fifo_0,
00650       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00651       south_0,
00652       south_1,
00653       south_2,
00654       south_3,
00655       south_4,
00656       south_5, south_last = south_5, ///< Used for shim BLI port validation
00657       west_0,
00658       west_1,
00659       west_2,
00660       west_3, west_last = west_3, ///< Used for west validation
00661       north_0,
00662       north_1,
00663       north_2,
00664       north_3,
00665       north_4,
00666       north_5, north_last = north_5, ///< Used for north validation
00667       east_0,
00668       east_1,
00669       east_2,
00670       east_3, east_last = east_3, ///< Used for east validation
00671       // To measure the enum
00672       size
00673     };
```

12.75.2.3 slave_port_layout [1/2]

```
template<typename Layout >
enum class trisycl::vendor::xilinx::acap::aie::geography::shim_axi_stream_switch::slave_port_layout
:  std::int8_t  [strong]
```

Layout of the AXI stream slave ports in the switch.

Revision v2.02, 6.5.3 Shim AXI-Stream Interconnect, Table 6-3, p. 291

Enumerator

tile_ctrl	
tile_ctrl_last	Used for validation,.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	
south_4	
south_5	
south_6	
south_7	
south_last	Used for shim BLI port validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	

Enumerator

north_2	
north_3	
north_last	Used for north validation.
east_0	
east_1	
east_2	
east_3	
east_last	Used for east validation.
shim_trace	
size	
tile_ctrl	
tile_ctrl_last	Used for validation,.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	
south_4	
south_5	
south_6	
south_7	
south_last	Used for shim BLI port validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_last	Used for north validation.
east_0	
east_1	
east_2	
east_3	
east_last	Used for east validation.
shim_trace	
size	

Definition at line 757 of file geography.hpp.

```
00757                                   :  std::int8_t {
00758       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation,
00759       fifo_0,
00760       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00761       south_0,
00762       south_1,
00763       south_2,
00764       south_3,
00765       south_4,
00766       south_5,
```

```
00767       south_6,
00768       south_7, south_last = south_7, ///< Used for shim BLI port validation
00769       west_0,
00770       west_1,
00771       west_2,
00772       west_3, west_last = west_3, ///< Used for west validation
00773       north_0,
00774       north_1,
00775       north_2,
00776       north_3, north_last = north_3, ///< Used for north validation
00777       east_0,
00778       east_1,
00779       east_2,
00780       east_3, east_last = east_3, ///< Used for east validation
00781       shim_trace,
00782       // To measure the enum
00783       size
00784     };
```

12.75.2.4 slave_port_layout [2/2]

```
template<typename Layout >
enum class trisycl::vendor::xilinx::acap::aie::geography::shim_axi_stream_switch::slave_port_layout
:  std::int8_t  [strong]
```

Layout of the AXI stream slave ports in the switch.

Revision v2.02, 6.5.3 Shim AXI-Stream Interconnect, Table 6-3, p. 291

Enumerator

tile_ctrl	
tile_ctrl_last	Used for validation,.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	
south_4	
south_5	
south_6	
south_7	
south_last	Used for shim BLI port validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_last	Used for north validation.
east_0	
east_1	

Enumerator

east_2	
east_3	
east_last	Used for east validation.
shim_trace	
size	
tile_ctrl	
tile_ctrl_last	Used for validation,.
fifo_0	
fifo_1	
fifo_last	Used for FIFO validation.
south_0	
south_1	
south_2	
south_3	
south_4	
south_5	
south_6	
south_7	
south_last	Used for shim BLI port validation.
west_0	
west_1	
west_2	
west_3	
west_last	Used for west validation.
north_0	
north_1	
north_2	
north_3	
north_last	Used for north validation.
east_0	
east_1	
east_2	
east_3	
east_last	Used for east validation.
shim_trace	
size	

Definition at line 757 of file geography.hpp.

```
00757                                  :  std::int8_t {
00758       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation,
00759       fifo_0,
00760       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00761       south_0,
00762       south_1,
00763       south_2,
00764       south_3,
00765       south_4,
00766       south_5,
00767       south_6,
00768       south_7, south_last = south_7, ///< Used for shim BLI port validation
00769       west_0,
00770       west_1,
00771       west_2,
00772       west_3, west_last = west_3, ///< Used for west validation
00773       north_0,
00774       north_1,
00775       north_2,
```

```
00776       north_3, north_last = north_3, ///< Used for north validation
00777       east_0,
00778       east_1,
00779       east_2,
00780       east_3, east_last = east_3, ///< Used for east validation
00781       shim_trace,
00782       // To measure the enum
00783       size
00784     };
```

12.75.3 Friends And Related Function Documentation

12.75.3.1 is_axi_master [1/4]

```
template<typename Layout >
bool constexpr is_axi_master (
             master_port_layout  )  [friend]
```

Todo Factorize out Use a hidden friend to have introspection for master_port_layout

Definition at line 679 of file geography.hpp.

```
00679                                                    {
00680       // Yes, master_port_layout is definitely an AXI master
00681       return true;
00682     }
```

12.75.3.2 is_axi_master [2/4]

```
template<typename Layout >
bool constexpr is_axi_master (
             master_port_layout  )  [friend]
```

Todo Factorize out Use a hidden friend to have introspection for master_port_layout

Definition at line 679 of file geography.hpp.

```
00679                                                    {
00680       // Yes, master_port_layout is definitely an AXI master
00681       return true;
00682     }
```

12.75.3.3 is_axi_master [3/4]

```
template<typename Layout >
bool constexpr is_axi_master (
             slave_port_layout  )  [friend]
```

Use a hidden friend to have introspection for slave_port_layout.

Definition at line 789 of file geography.hpp.

```
00789                                                    {
00790       // No, slave_port_layout is definitely not an AXI master
00791       return false;
00792     }
```

12.75.3.4 is_axi_master [4/4]

```
template<typename Layout >
bool constexpr is_axi_master (
             slave_port_layout  )  [friend]
```

Use a hidden friend to have introspection for slave_port_layout.

Definition at line 789 of file geography.hpp.

```
00789                                                          {
00790       // No, slave_port_layout is definitely not an AXI master
00791       return false;
00792     }
```

12.75.4 Member Data Documentation

12.75.4.1 fifo_depth

```
template<typename Layout >
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::fifo_depth = 16  [static], [constexpr]
```

Depth of additional FIFO on specific FIFO paths.

Definition at line 642 of file geography.hpp.

12.75.4.2 latency

```
template<typename Layout >
static auto constexpr trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::latency = 4  [static], [constexpr]
```

Todo factorize the 2 following values with core tiles? Default number of registers on default router paths

Todo factorize the 2 following values with core tiles? Default number of registers on default router paths

Definition at line 639 of file geography.hpp.

12.75.4.3 m_east_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch↩
::m_east_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::east_0,
                       master_port_layout::east_last)
```

A range of the East master ports.

Definition at line 745 of file geography.hpp.

12.75.4.4 m_east_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::m_east_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::east_0,
                         master_port_layout::east_last)
```

Number of East master ports.

Definition at line 750 of file geography.hpp.

12.75.4.5 m_fifo_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch↩
::m_fifo_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::fifo_0,
                       master_port_layout::fifo_last)
```

A range of the master ports connected to the FIFO inputs.

Definition at line 705 of file geography.hpp.

12.75.4.6 m_fifo_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::m_fifo_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::fifo_0,
                         master_port_layout::fifo_last)
```

Number of master ports connected to the FIFO inputs.

Definition at line 710 of file geography.hpp.

12.75.4.7 m_north_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch↩
::m_north_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::north_0,
                       master_port_layout::north_last)
```

A range of the North master ports.

Definition at line 735 of file geography.hpp.

12.75.4.8 m_north_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::m_north_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::north_0,
                         master_port_layout::north_last)
```

Number of North master ports.

Definition at line 740 of file geography.hpp.

12.75.4.9 m_south_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch↩
::m_south_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::south_0,
                       master_port_layout::south_last)
```

A range of the South master ports.

Definition at line 715 of file geography.hpp.

12.75.4.10 m_south_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::m_south_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::south_0,
                         master_port_layout::south_last)
```

Number of South master ports.

Definition at line 720 of file geography.hpp.

12.75.4.11 m_tile_ctrl_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch↩
::m_tile_ctrl_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::tile_ctrl,
                       master_port_layout::tile_ctrl_last)
```

A range of the master ports connected to the tile control inputs.

Todo Factorize out this range code with core tile

Todo Factorize out this range code with core tile

Definition at line 695 of file geography.hpp.

12.75.4.12 m_tile_ctrl_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::m_tile_ctrl_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::tile_ctrl,
                         master_port_layout::tile_ctrl_last)
```

Number of master ports connected to the tile control inputs.

Definition at line 700 of file geography.hpp.

12.75.4.13 m_west_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch↩
::m_west_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(master_port_layout::west_0,
                       master_port_layout::west_last)
```

A range of the West master ports.

Definition at line 725 of file geography.hpp.

12.75.4.14 m_west_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::m_west_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(master_port_layout::west_0,
                         master_port_layout::west_last)
```

Number of West master ports.

Definition at line 730 of file geography.hpp.

12.75.4.15 nb_master_port

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::nb_master_port  [static], [constexpr]
```

Initial value:

```
=
      detail::underlying_value(master_port_layout::size)
```

Number of AXI stream master ports in the switch.

Todo factorize with a mix-in the following between master and slave

Todo factorize with a mix-in the following between master and slave

Definition at line 689 of file geography.hpp.

12.75.4.16 nb_slave_port

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::nb_slave_port  [static], [constexpr]
```

Initial value:

```
=
      static_cast<int>(slave_port_layout::size)
```

Number of master AXI stream slave ports in the switch.

Definition at line 797 of file geography.hpp.

12.75.4.17 s_east_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch↩
::s_east_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::east_0, slave_port_layout::east_last)
```

A range of the East slave ports.

Definition at line 849 of file geography.hpp.

12.75.4.18 s_east_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::s_east_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::east_0,
                         slave_port_layout::east_last)
```

Number of East slave ports.

Definition at line 853 of file geography.hpp.

12.75.4.19 s_fifo_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch↩
::s_fifo_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::fifo_0, slave_port_layout::fifo_last)
```

A range of the slave ports connected to the FIFO outputs.

Definition at line 811 of file geography.hpp.

12.75.4.20 s_fifo_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::s_fifo_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::fifo_0,
                         slave_port_layout::fifo_last)
```

Number of slave ports connected to the FIFO outputs.

Definition at line 815 of file geography.hpp.

12.75.4.21 s_north_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch↩
::s_north_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::north_0,
                       slave_port_layout::north_last)
```

A range of the North slave ports.

Definition at line 839 of file geography.hpp.

12.75.4.22 s_north_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::s_north_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::north_0,
                         slave_port_layout::north_last)
```

Number of North slave ports.

Definition at line 844 of file geography.hpp.

12.75.4.23 s_south_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch↩
::s_south_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::south_0,
                       slave_port_layout::south_last)
```

A range of the South slave ports.

Definition at line 820 of file geography.hpp.

12.75.4.24 s_south_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::s_south_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::south_0,
                         slave_port_layout::south_last)
```

Number of South slave ports.

Definition at line 825 of file geography.hpp.

12.75.4.25 s_tile_ctrl_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch↩
::s_tile_ctrl_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::tile_ctrl,
                       slave_port_layout::tile_ctrl_last)
```

A range of the slave ports connected to the tile control outputs.

Definition at line 801 of file geography.hpp.

12.75.4.26 s_tile_ctrl_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::s_tile_ctrl_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::tile_ctrl,
                         slave_port_layout::tile_ctrl_last)
```

Number of slave ports connected to the tile control outputs.

Definition at line 806 of file geography.hpp.

12.75.4.27 s_west_range

```
template<typename Layout >
static auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch↩
::s_west_range  [inline], [static]
```

Initial value:

```
=
      views::enum_type(slave_port_layout::west_0, slave_port_layout::west_last)
```

A range of the West slave ports.

Definition at line 830 of file geography.hpp.

12.75.4.28 s_west_size

```
template<typename Layout >
static constexpr auto trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_↩
stream_switch::s_west_size  [static], [constexpr]
```

Initial value:

```
=
      detail::enum_count(slave_port_layout::west_0,
                         slave_port_layout::west_last)
```

Number of West slave ports.

Definition at line 834 of file geography.hpp.

The documentation for this struct was generated from the following files:

- include/sycl/vendor/Xilinx/acap/aie/geography.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/geography.hpp

12.76 trisycl::vendor::xilinx::acap::aie::axi_packet::shutdown_t Struct Reference

Marker type used to signal network shutdown.

12.76.1 Detailed Description

Marker type used to signal network shutdown.

Definition at line 33 of file connection.hpp.

The documentation for this struct was generated from the following file:

- include/sycl/vendor/Xilinx/acap/aie/connection.hpp

12.77 trisycl::detail::singleton< T > Struct Template Reference

Provide a singleton factory.

`#include <singleton.hpp>`

Static Public Member Functions

- static std::shared_ptr< T > instance ()

  Get a singleton instance of T.

12.77.1 Detailed Description

template<typename T>
struct trisycl::detail::singleton< T >

Provide a singleton factory.

Definition at line 23 of file singleton.hpp.

12.77.2 Member Function Documentation

12.77.2.1 instance()

```
template<typename T >
static std::shared_ptr< T > trisycl::detail::singleton< T >::instance ( )  [inline], [static]
```

Get a singleton instance of T.

Use a null_deleter since the singleton should not be deleted, as allocated in the static area

Definition at line 26 of file singleton.hpp.

```
00026                                    {
00027     // C++11 guaranties the static construction is thread-safe
00028     static T single;
00029 /** Use a null_deleter since the singleton should not be deleted,
00030 as allocated in the static area */
00031     static std::shared_ptr<T> sps { &single,
00032                                     boost::null_deleter {} };
00033
00034     return sps;
00035   }
```

The documentation for this struct was generated from the following file:

- include/triSYCL/detail/singleton.hpp

12.78 trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, StorageType > Struct Template Reference

`#include <util.hpp>`

Public Types

- using type = StorageType
- using type = StorageType

12.78.1 Detailed Description

template<typename ScopedType, typename StorageType = void>
struct trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, StorageType >

Definition at line 17 of file util.hpp.

12.78.2 Member Typedef Documentation

12.78.2.1 type [1/2]

```
template<typename ScopedType , typename StorageType = void>
using trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, StorageType >↩
::type = StorageType
```

Definition at line 18 of file util.hpp.

12.78.2.2 type [2/2]

```
template<typename ScopedType , typename StorageType = void>
using trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, StorageType >↩
::type = StorageType
```

Definition at line 18 of file util.hpp.

The documentation for this struct was generated from the following files:

- include/sycl/vendor/triSYCL/scope/detail/util.hpp
- include/triSYCL/vendor/triSYCL/scope/detail/util.hpp

12.79 trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, typename ScopedType::storage_type > Struct Template Reference

`#include <util.hpp>`

Public Types

- using type = typename ScopedType::storage_type
- using type = typename ScopedType::storage_type

12.79.1 Detailed Description

template<typename ScopedType>
struct trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, typename ScopedType::storage_type >

Definition at line 23 of file util.hpp.

12.79.2 Member Typedef Documentation

12.79.2.1 type [1/2]

```
template<typename ScopedType >
using trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, typename Scoped↩
Type::storage_type >::type = typename ScopedType::storage_type
```

Definition at line 24 of file util.hpp.

12.79.2.2 type [2/2]

```
template<typename ScopedType >
using trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, typename Scoped↩
Type::storage_type >::type = typename ScopedType::storage_type
```

Definition at line 24 of file util.hpp.

The documentation for this struct was generated from the following files:

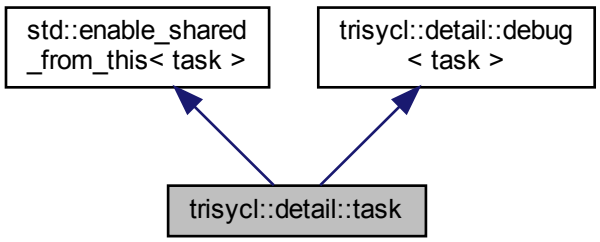
- include/sycl/vendor/triSYCL/scope/detail/util.hpp
- include/triSYCL/vendor/triSYCL/scope/detail/util.hpp

12.80 trisycl::detail::task Struct Reference

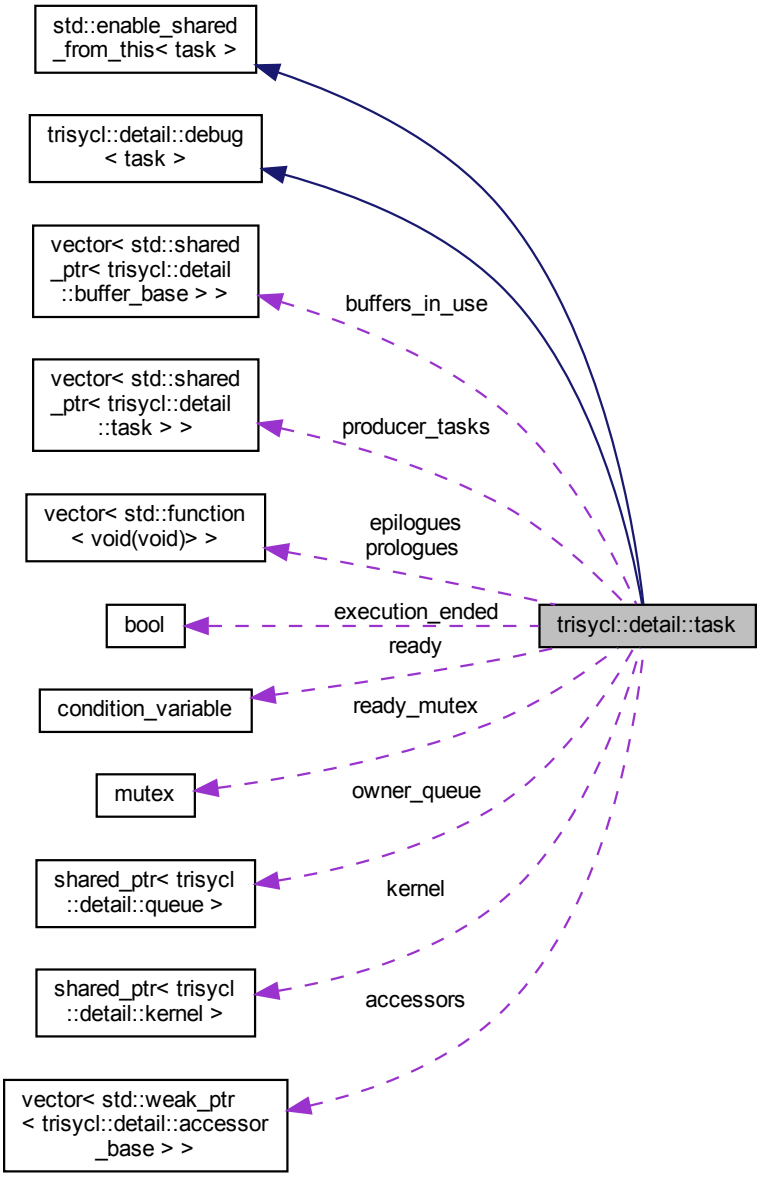
The abstraction to represent SYCL tasks executing inside command_group.

`#include <task.hpp>`

Inheritance diagram for trisycl::detail::task:



Collaboration diagram for trisycl::detail::task:



Public Member Functions

- task (const std::shared_ptr< detail::queue > &q)

  Create a task from a submitting queue.
- void schedule (std::function< void(void)> f)

  Add a new task to the task graph and schedule for execution.
- void wait_for_producers ()

  Wait for the required producer tasks to be ready.
- void release_buffers ()

  Release the buffers that have been used by this task.

- void notify_consumers ()

  Notify the waiting tasks that we are done.
- void wait ()

  Wait for this task to be ready.
- void add_buffer (std::shared_ptr< detail::buffer_base > &buf, bool is_write_mode)

  Register a buffer to this task.
- void prelude ()

  Execute the prologues.
- void postlude ()

  Execute the epilogues.
- void add_prelude (const std::function< void(void)> &f)

  Add a function to the prelude to run before kernel execution.
- void add_postlude (const std::function< void(void)> &f)

  Add a function to the postlude to run after kernel execution.
- auto get_queue ()

  Get the queue behind the task to run a kernel on.
- void set_kernel (const std::shared_ptr< trisycl::detail::kernel > &k)

  Set the OpenCL-compatible kernel running this task if any.
- trisycl::detail::kernel & get_kernel ()

  Get the kernel running if any.
- void set_arg (std::size_t arg_index, std::size_t arg_size, const void *scalar_value)

  Set a kernel argument by address.
- template<typename T >
  void set_arg (std::size_t arg_index, const T &scalar_value)

  Set a kernel argument by value.
- auto get_compute_buffer (std::size_t order)

  Get the Boost.Compute buffer for an accessor of the task.
- std::size_t register_accessor (std::weak_ptr< detail::accessor_base > a)

  Register an accessor and return its registration order.

Public Attributes

- std::vector< std::shared_ptr< detail::buffer_base > > buffers_in_use

  List of the buffers used by this task.
- std::vector< std::shared_ptr< detail::task > > producer_tasks

  The tasks producing the buffers used by this task.
- std::vector< std::function< void(void)> > prologues

  Keep track of any prologue to be executed before the kernel.
- std::vector< std::function< void(void)> > epilogues

  Keep track of any epilogue to be executed after the kernel.
- bool execution_ended = false

  Store if the execution ended, to be notified by task_ready.
- std::condition_variable ready

  To signal when this task is ready.
- std::mutex ready_mutex

  To protect the access to the condition variable.
- std::shared_ptr< detail::queue > owner_queue

  Keep track of the queue used to submission to notify kernel completion or to run OpenCL kernels on.
- std::shared_ptr< detail::kernel > kernel

  The OpenCL-compatible kernel run by this task, if any.
- std::vector< std::weak_ptr< detail::accessor_base > > accessors

  The accessors indexed by their creation order.

Additional Inherited Members

12.80.1 Detailed Description

The abstraction to represent SYCL tasks executing inside command_group.

"enable_shared_from_this" allows to access the shared_ptr behind the scene.

Definition at line 35 of file task.hpp.

12.80.2 Constructor & Destructor Documentation

12.80.2.1 task()

```
trisycl::detail::task::task (
            const std::shared_ptr< detail::queue > & q )  [inline]
```

Create a task from a submitting queue.

Definition at line 78 of file task.hpp.

```
00079     :  owner_queue { q } {}
```

12.80.3 Member Function Documentation

12.80.3.1 add_buffer()

```
void trisycl::detail::task::add_buffer (
            std::shared_ptr< detail::buffer_base > & buf,
            bool is_write_mode )  [inline]
```

Register a buffer to this task.

This is how the dependency graph is incrementally built.

Definition at line 176 of file task.hpp.

```
00177                                     {
00178     TRISYCL_DUMP_T("Add buffer " « buf « " in task " « this);
00179     /* Keep track of the use of the buffer to notify its release at
00180 the end of the execution */
00181     buffers_in_use.push_back(buf);
00182     // To be sure the buffer does not disappear before the kernel can run
00183     buf->use();
00184
00185     std::shared_ptr<detail::task> latest_producer;
00186     if (is_write_mode) {
00187       /* Set this task as the latest producer of the buffer so that
00188 another kernel may wait on this task */
00189       latest_producer = buf->set_latest_producer(shared_from_this());
00190     }
00191     else
00192       latest_producer = buf->get_latest_producer();
00193
00194     /* If the buffer is to be produced by a task, add the task in the
00195 producer list to wait on it before running the task core
00196
00197 If a buffer is accessed first in write mode and then in read mode,
00198 the task will add itself as a producer and will wait for itself
00199 when calling \c wait_for_producers, we avoid this by checking that
00200 \c latest_producer is not \c this
00201 */
00202     if (latest_producer && latest_producer != shared_from_this())
00203       producer_tasks.push_back(latest_producer);
00204   }
```

References buffers_in_use, producer_tasks, and TRISYCL_DUMP_T.

12.80.3.2 add_postlude()

```
void trisycl::detail::task::add_postlude (
            const std::function< void(void)> & f )  [inline]
```

Add a function to the postlude to run after kernel execution.

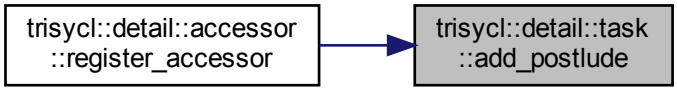
Definition at line 238 of file task.hpp.

```
00238                                                {
00239     epilogues.push_back(f);
00240   }
```

References epilogues.

Referenced by trisycl::detail::accessor< T, Dimensions, Mode, Target >::register_accessor().

Here is the caller graph for this function:



12.80.3.3 add_prelude()

```
void trisycl::detail::task::add_prelude (
            const std::function< void(void)> & f )  [inline]
```

Add a function to the prelude to run before kernel execution.

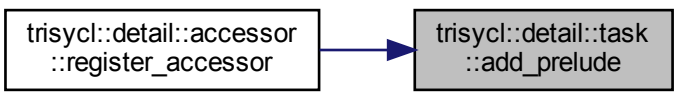
Definition at line 230 of file task.hpp.

```
00230                                               {
00231     TRISYCL_DUMP_T("task::add_prelude");
00232
00233     prologues.push_back(f);
00234   }
```

References prologues, and TRISYCL_DUMP_T.

Referenced by trisycl::detail::accessor< T, Dimensions, Mode, Target >::register_accessor().

Here is the caller graph for this function:



12.80.3.4 get_compute_buffer()

```
auto trisycl::detail::task::get_compute_buffer (
             std::size_t order )  [inline]
```

Get the Boost.Compute buffer for an accessor of the task.

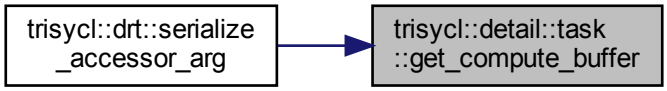
Definition at line 293 of file task.hpp.

```
00293                                          {
00294     return accessors[order].lock()->get_cl_buffer();
00295   }
```

References accessors.

Referenced by trisycl::drt::serialize_accessor_arg().

Here is the caller graph for this function:



12.80.3.5 get_kernel()

```
trisycl::detail::kernel & trisycl::detail::task::get_kernel ( )  [inline]
```

Get the kernel running if any.

Todo Specify this error in the spec

Definition at line 259 of file task.hpp.

```
00259                                       {
00260     if (!kernel)
00261       throw non_cl_error("Cannot use an OpenCL kernel in this context");
00262     return *kernel;
00263   }
```

References kernel.

Referenced by set_arg().

Here is the caller graph for this function:



12.80.3.6 get_queue()

```
auto trisycl::detail::task::get_queue ( )  [inline]
```

Get the queue behind the task to run a kernel on.

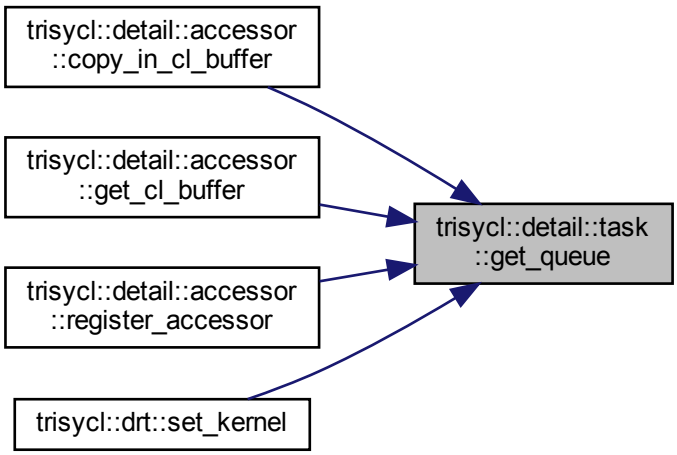
Definition at line 244 of file task.hpp.

```
00244                     {
00245     return owner_queue;
00246   }
```

References owner_queue.

Referenced by trisycl::detail::accessor< T, Dimensions, Mode, Target >::copy_in_cl_buffer(), trisycl::detail::accessor< T, Dimensions
trisycl::detail::accessor< T, Dimensions, Mode, Target >::register_accessor(), and trisycl::drt::set_kernel().

Here is the caller graph for this function:



12.80.3.7 notify_consumers()

```
void trisycl::detail::task::notify_consumers ( )  [inline]
```

Notify the waiting tasks that we are done.

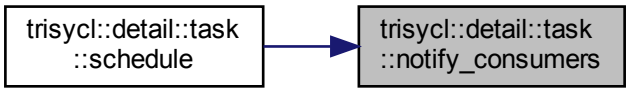
Definition at line 149 of file task.hpp.

```
00149                           {
00150     TRISYCL_DUMP_T("Notify all the task waiting for this task " « this);
00151     {
00152       std::unique_lock<std::mutex> ul { ready_mutex };
00153       execution_ended = true;
00154     }
00155     /* \todo Verify that the memory model with the notify does not
00156 require some fence or atomic */
00157     ready.notify_all();
00158   }
```

References execution_ended, ready, ready_mutex, and TRISYCL_DUMP_T.

Referenced by schedule().

Here is the caller graph for this function:



12.80.3.8 postlude()

```
void trisycl::detail::task::postlude ( )  [inline]
```

Execute the epilogues.

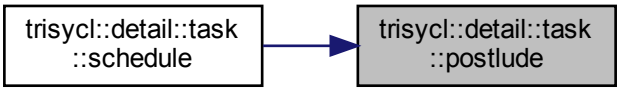
Definition at line 220 of file task.hpp.

```
00220                 {
00221     for (const auto &p :  epilogues)
00222       p();
00223     /* Free the functors that may own an accessor owning a buffer
00224 preventing the command group to complete */
00225     epilogues.clear();
00226   }
```

References epilogues, and p.

Referenced by schedule().

Here is the caller graph for this function:



12.80.3.9 prelude()

```
void trisycl::detail::task::prelude ( )  [inline]
```

Execute the prologues.

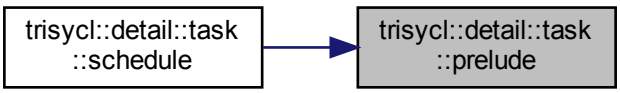
Definition at line 208 of file task.hpp.

```
00208                {
00209     TRISYCL_DUMP_T("task::prelude");
00210
00211     for (const auto &p :  prologues)
00212       p();
00213     /* Free the functors that may own an accessor owning a buffer
00214 preventing the command group to complete */
00215     prologues.clear();
00216   }
```

References p, prologues, and TRISYCL_DUMP_T.

Referenced by schedule().

Here is the caller graph for this function:



12.80.3.10 register_accessor()

```
std::size_t trisycl::detail::task::register_accessor (
             std::weak_ptr< detail::accessor_base > a )  [inline]
```

Register an accessor and return its registration order.

I would prefer to name this method `register` but this is a C++ keyword...

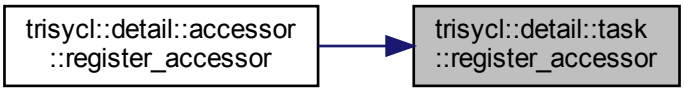
Definition at line 304 of file task.hpp.

```
00304                                                    {
00305   accessors.push_back(a);
00306   return accessors.size() - 1;
00307 }
```

References a, and accessors.

Referenced by trisycl::detail::accessor< T, Dimensions, Mode, Target >::register_accessor().

Here is the caller graph for this function:



12.80.3.11 release_buffers()

```
void trisycl::detail::task::release_buffers ( )  [inline]
```

Release the buffers that have been used by this task.

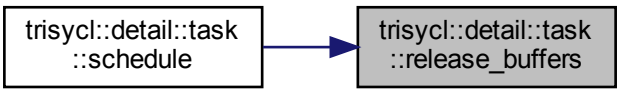
Definition at line 140 of file task.hpp.

```
00140                           {
00141     TRISYCL_DUMP_T("Task " « this « " releases the written buffers");
00142     for (auto b:  buffers_in_use)
00143       b->release();
00144     buffers_in_use.clear();
00145   }
```

References b, buffers_in_use, and TRISYCL_DUMP_T.

Referenced by schedule().

Here is the caller graph for this function:



12.80.3.12 schedule()

```
void trisycl::detail::task::schedule (
            std::function< void(void)> f )  [inline]
```

Add a new task to the task graph and schedule for execution.

Detach the thread since it will synchronize by its own means

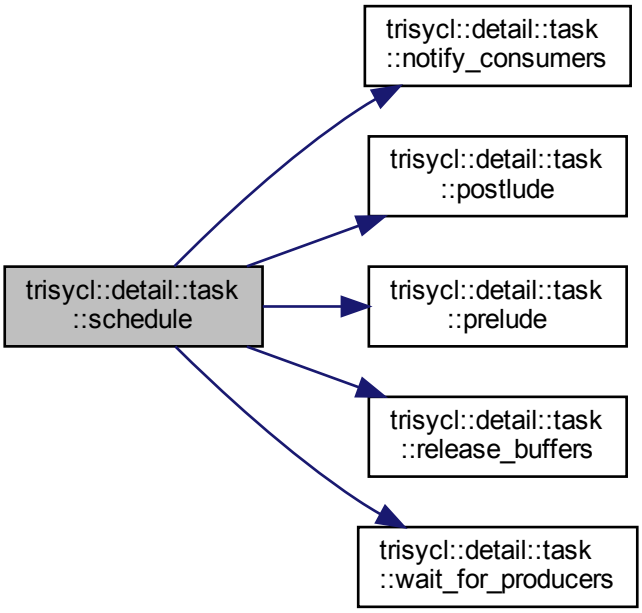
Todo This is an issue if there is an exception in the kernel

Definition at line 83 of file task.hpp.

```
00083                                                {
00084     /* To keep a copy of the task shared_ptr after the end of the
00085 command group, capture it by copy in the following lambda.
00086 */
00087     auto execution = [f = std::move(f), task = shared_from_this()] {
00088       // Wait for the required tasks to be ready
00089       task->wait_for_producers();
00090       task->prelude();
00091       TRISYCL_DUMP_T("Execute the kernel");
00092       // Execute the kernel
00093       f();
00094       task->postlude();
00095       // Release the buffers that have been written by this task
00096       task->release_buffers();
00097       // Notify the waiting tasks that we are done
00098       task->notify_consumers();
00099       // Notify the queue we are done
00100       task->owner_queue->kernel_end();
00101       TRISYCL_DUMP_T("Task thread exit");
00102     };
00103     /* Notify the queue that there is a kernel submitted to the
00104 queue.  Do not do it in the task contructor so that we can deal
00105 with command group without kernel and if we put it inside the
00106 thread, the queue may have finished before the thread is
00107 scheduled */
00108     owner_queue->kernel_start();
00109     /* \todo it may be implementable with packaged_task that would
00110 deal with exceptions in kernels
00111 */
00112 #ifndef TRISYCL_NO_ASYNC
00113     /* If in asynchronous execution mode, execute the functor in a new
00114 thread */
00115     std::thread thread(execution);
00116     TRISYCL_DUMP_T("Task thread started");
00117 /** Detach the thread since it will synchronize by its own means
00118
00119 \todo This is an issue if there is an exception in the kernel
00120 */
00121     thread.detach();
00122 #else
00123     // Just a synchronous execution otherwise
00124     execution();
00125 #endif
00126   }
```

References notify_consumers(), owner_queue, postlude(), prelude(), release_buffers(), TRISYCL_DUMP_T, and wait_for_producers().

Here is the call graph for this function:



12.80.3.13 set_arg() [1/2]

```
template<typename T >
void trisycl::detail::task::set_arg (
              std::size_t arg_index,
              const T & scalar_value )  [inline]
```

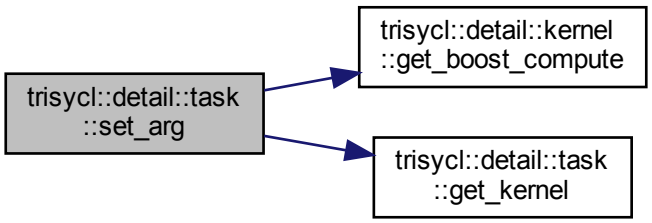
Set a kernel argument by value.

Definition at line 281 of file task.hpp.

```
00281                                                 {
00282 #ifdef TRISYCL_OPENCL
00283     // Forward to the OpenCL kernel
00284     get_kernel().get_boost_compute().set_arg(arg_index, scalar_value);
00285 #else
00286     throw non_cl_error("Not compiled with OpenCL support");
00287 #endif
00288   }
```

References trisycl::detail::kernel::get_boost_compute(), and get_kernel().

Here is the call graph for this function:



12.80.3.14 set_arg() [2/2]

```
void trisycl::detail::task::set_arg (
              std::size_t arg_index,
              std::size_t arg_size,
              const void * scalar_value )  [inline]
```

Set a kernel argument by address.

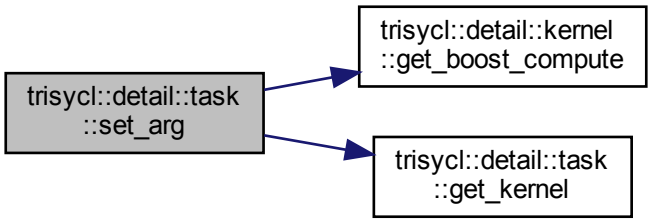
Definition at line 267 of file task.hpp.

```
00267                                                                                {
00268 #ifdef TRISYCL_OPENCL
00269     // Forward to the OpenCL kernel
00270     get_kernel().get_boost_compute().set_arg(arg_index,
00271                                              arg_size,
00272                                              scalar_value);
00273 #else
00274     throw non_cl_error("Not compiled with OpenCL support");
00275 #endif
00276   }
```

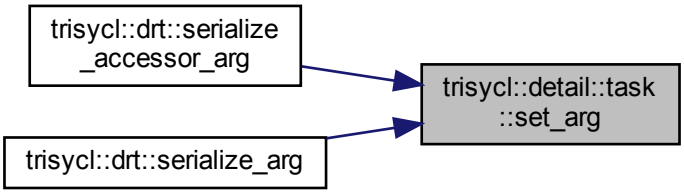
References trisycl::detail::kernel::get_boost_compute(), and get_kernel().

Referenced by trisycl::drt::serialize_accessor_arg(), and trisycl::drt::serialize_arg().

Here is the call graph for this function:



Here is the caller graph for this function:



12.80.3.15 set_kernel()

```
void trisycl::detail::task::set_kernel (
            const std::shared_ptr< trisycl::detail::kernel > & k )  [inline]
```

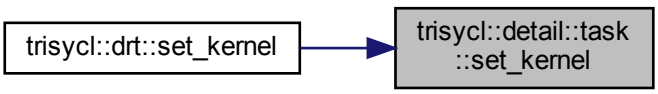
Set the OpenCL-compatible kernel running this task if any.

Definition at line 250 of file task.hpp.

```
00250                                                          {
00251     kernel = k;
00252   }
```

Referenced by trisycl::drt::set_kernel().

Here is the caller graph for this function:



12.80.3.16 wait()

```
void trisycl::detail::task::wait ( )  [inline]
```

Wait for this task to be ready.

This is to be called from another thread

Definition at line 165 of file task.hpp.

```
00165           {
00166     TRISYCL_DUMP_T("The task wait for task " << this << " to end");
00167     std::unique_lock<std::mutex> ul { ready_mutex };
00168     ready.wait(ul, [&] { return execution_ended; });
00169   }
```

References execution_ended, ready, ready_mutex, and TRISYCL_DUMP_T.

12.80.3.17 wait_for_producers()

```
void trisycl::detail::task::wait_for_producers ( )  [inline]
```

Wait for the required producer tasks to be ready.

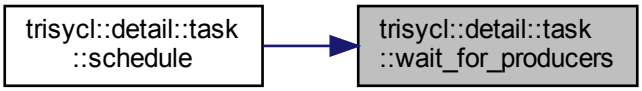
Definition at line 130 of file task.hpp.

```
00130                                {
00131     TRISYCL_DUMP_T("Task " « this « " waits for the producer tasks");
00132     for (auto &t :  producer_tasks)
00133       t->wait();
00134     // We can let the producers rest in peace
00135     producer_tasks.clear();
00136   }
```

References producer_tasks, and TRISYCL_DUMP_T.

Referenced by schedule().

Here is the caller graph for this function:



12.80.4 Member Data Documentation

12.80.4.1 accessors

```
std::vector<std::weak_ptr<detail::accessor_base> > trisycl::detail::task::accessors
```

The accessors indexed by their creation order.

This is used to relate a kernel parameter of a kernel generated by the device compiler to its accessor.

Definition at line 74 of file task.hpp.

Referenced by get_compute_buffer(), and register_accessor().

12.80.4.2 buffers_in_use

```
std::vector<std::shared_ptr<detail::buffer_base> > trisycl::detail::task::buffers_in_use
```

List of the buffers used by this task.

Todo Use a set to check that some buffers are not used many times at least on writing

Definition at line 43 of file task.hpp.

Referenced by add_buffer(), and release_buffers().

12.80.4.3 epilogues

```
std::vector<std::function<void(void)> > trisycl::detail::task::epilogues
```

Keep track of any epilogue to be executed after the kernel.

Definition at line 52 of file task.hpp.

Referenced by add_postlude(), and postlude().

12.80.4.4 execution_ended

```
bool trisycl::detail::task::execution_ended = false
```

Store if the execution ended, to be notified by task_ready.

Definition at line 55 of file task.hpp.

Referenced by notify_consumers(), and wait().

12.80.4.5 kernel

```
std::shared_ptr<detail::kernel> trisycl::detail::task::kernel
```

The OpenCL-compatible kernel run by this task, if any.

Definition at line 68 of file task.hpp.

Referenced by get_kernel().

12.80.4.6 owner_queue

```
std::shared_ptr<detail::queue> trisycl::detail::task::owner_queue
```

Keep track of the queue used to submission to notify kernel completion or to run OpenCL kernels on.

Definition at line 65 of file task.hpp.

Referenced by get_queue(), and schedule().

12.80.4.7 producer_tasks

```
std::vector<std::shared_ptr<detail::task> > trisycl::detail::task::producer_tasks
```

The tasks producing the buffers used by this task.

Definition at line 46 of file task.hpp.

Referenced by add_buffer(), and wait_for_producers().

12.80.4.8 prologues

```
std::vector<std::function<void(void)> > trisycl::detail::task::prologues
```

Keep track of any prologue to be executed before the kernel.

Definition at line 49 of file task.hpp.

Referenced by add_prelude(), and prelude().

12.80.4.9 ready

```
std::condition_variable trisycl::detail::task::ready
```

To signal when this task is ready.

Definition at line 58 of file task.hpp.

Referenced by notify_consumers(), and wait().

12.80.4.10 ready_mutex

```
std::mutex trisycl::detail::task::ready_mutex
```

To protect the access to the condition variable.

Definition at line 61 of file task.hpp.

Referenced by notify_consumers(), and wait().

The documentation for this struct was generated from the following file:

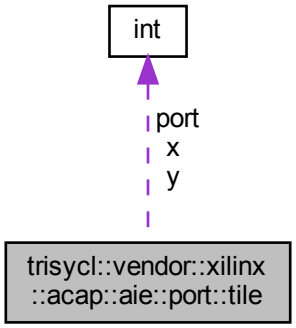
- include/triSYCL/command_group/detail/task.hpp

12.81 trisycl::vendor::xilinx::acap::aie::port::tile Struct Reference

Abstraction of a communication port inside a tile.

```
#include <connection.hpp>
```

Collaboration diagram for trisycl::vendor::xilinx::acap::aie::port::tile:



Public Attributes

- const int x
- const int y
- const int port

12.81.1 Detailed Description

Abstraction of a communication port inside a tile.

Definition at line 252 of file connection.hpp.

12.81.2 Member Data Documentation

12.81.2.1 port

```
const int trisycl::vendor::xilinx::acap::aie::port::tile::port
```

Definition at line 255 of file connection.hpp.

12.81.2.2 x

```
const int trisycl::vendor::xilinx::acap::aie::port::tile::x
```

Definition at line 253 of file connection.hpp.

12.81.2.3 y

```
const int trisycl::vendor::xilinx::acap::aie::port::tile::y
```

Definition at line 254 of file connection.hpp.

The documentation for this struct was generated from the following files:

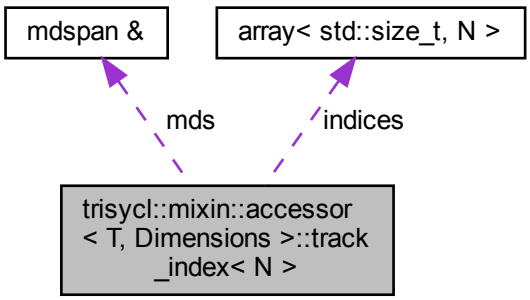
- include/sycl/vendor/Xilinx/acap/aie/connection.hpp
- include/triSYCL/vendor/Xilinx/acap/aie/connection.hpp

12.82 trisycl::mixin::accessor< T, Dimensions >::track_index< N > Struct Template Reference

Proxy object to transform an expression like accessor[i1][i2][i3] into the implementation mdpsan(i1,i2,i3) one index at a time.

```
#include <accessor.hpp>
```

Collaboration diagram for trisycl::mixin::accessor< T, Dimensions >::track_index< N >:



Public Member Functions

- track_index (accessor &a)

  Construct the initial tracking object from the accessor.
- template<typename... Index>
  track_index (mdspan &m, Index &&... inds)

  Create a tracking object from an mdspan and a list of indices.
- decltype(auto) operator[ ] (std::size_t index)

  The individual indexing operator.

Public Attributes

- mdspan & mds

  Keep a reference to the mdspan to eventually resolve the indexing.
- std::array< std::size_t, N > indices

  The list of indices in the order of [i1][i2][i3]...

12.82.1 Detailed Description

template<typename T, int Dimensions>
template<std::size_t N>
struct trisycl::mixin::accessor< T, Dimensions >::track_index< N >

Proxy object to transform an expression like accessor[i1][i2][i3] into the implementation mdpsan(i1,i2,i3) one index at a time.

It gathers intermediate [index] to finally call the mdspan indexing operator once they are all available

\parameter[in] N is the number of indices which can be stored in this proxy

Definition at line 174 of file accessor.hpp.

12.82.2 Constructor & Destructor Documentation

12.82.2.1 track_index() [1/2]

```
template<typename T , int Dimensions>
template<std::size_t N>
trisycl::mixin::accessor< T, Dimensions >::track_index< N >::track_index (
            accessor & a )  [inline]
```

Construct the initial tracking object from the accessor.

Definition at line 183 of file accessor.hpp.

```
00184         :  mds { a.access } {}
```

12.82.2.2 track_index() [2/2]

```
template<typename T , int Dimensions>
template<std::size_t N>
template<typename...  Index>
trisycl::mixin::accessor< T, Dimensions >::track_index< N >::track_index (
             mdspan & m,
             Index &&...  inds )  [inline]
```

Create a tracking object from an mdspan and a list of indices.

Definition at line 188 of file accessor.hpp.

```
00189         :  mds { m }
00190         // Typically there will be N - 1 indices in the array of size N
00191         , indices { std::forward<Index>(inds)...  } {}
```

12.82.3 Member Function Documentation

12.82.3.1 operator[]()

```
template<typename T , int Dimensions>
template<std::size_t N>
decltype(auto) trisycl::mixin::accessor< T, Dimensions >::track_index< N >::operator[] (
             std::size_t index )  [inline]
```

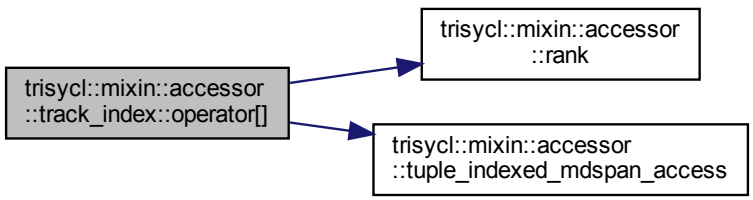
The individual indexing operator.

Definition at line 194 of file accessor.hpp.

```
00194                                                {
00195       // Keep track of the new index in the last element
00196       indices[N - 1] = index;
00197       if constexpr (N == accessor::rank())
00198         // If we have accumulated all the indices, call the mdspan
00199         // indexing function
00200         return tuple_indexed_mdspan_access(mds, indices);
00201       else
00202         // Otherwise return a tracker from the current indices and
00203         // with 1 more room for the next index
00204         return std::make_from_tuple<track_index<N + 1>>(
00205             std::tuple_cat(std::forward_as_tuple(mds), indices));
00206     }
```

References trisycl::mixin::accessor< T, Dimensions >::track_index< N >::indices, trisycl::mixin::accessor< T, Dimensions >::track_
trisycl::mixin::accessor< T, Dimensions >::rank(), and trisycl::mixin::accessor< T, Dimensions >::tuple_indexed_mdspan_access().

Here is the call graph for this function:



12.82.4 Member Data Documentation

12.82.4.1 indices

```
template<typename T , int Dimensions>
template<std::size_t N>
std::array<std::size_t, N> trisycl::mixin::accessor< T, Dimensions >::track_index< N >↩
::indices
```

The list of indices in the order of [i1][i2][i3]...

Definition at line 180 of file accessor.hpp.

Referenced by trisycl::mixin::accessor< T, Dimensions >::track_index< N >::operator[ ]().

12.82.4.2 mds

```
template<typename T , int Dimensions>
template<std::size_t N>
mdspan& trisycl::mixin::accessor< T, Dimensions >::track_index< N >::mds
```

Keep a reference to the mdspan to eventually resolve the indexing.

Definition at line 177 of file accessor.hpp.

Referenced by trisycl::mixin::accessor< T, Dimensions >::track_index< N >::operator[ ]().

The documentation for this struct was generated from the following file:

- include/triSYCL/accessor/mixin/accessor.hpp

12.83 std::tuple_element< I, T > Struct Template Reference

Each tuple element type is the same, the one of any small_array heir element.

```
#include <small_array.hpp>
```

Public Types

- using type = typename T::element_type

12.83.1 Detailed Description

template<std::size_t I, typename T>
requires trisycl::detail::is_small_array<T>
struct std::tuple_element< I, T >

Each tuple element type is the same, the one of any small_array heir element.

Definition at line 410 of file small_array.hpp.

12.83.2 Member Typedef Documentation

12.83.2.1 type

```
template<std::size_t I, typename T >
using std::tuple_element< I, T >::type = typename T::element_type
```

Definition at line 411 of file small_array.hpp.

The documentation for this struct was generated from the following file:

- include/triSYCL/detail/small_array.hpp

12.84 std::tuple_element< I, trisycl::vec< T, Size > > Struct Template Reference

The std::tuple_element specialization for vec.

```
#include <vec.hpp>
```

Public Types

- using type = T

12.84.1 Detailed Description

template<std::size_t I, typename T, int Size>
struct std::tuple_element< I, trisycl::vec< T, Size > >

The std::tuple_element specialization for vec.

Definition at line 575 of file vec.hpp.

12.84.2 Member Typedef Documentation

12.84.2.1 type

```
template<std::size_t I, typename T , int Size>
using std::tuple_element< I, trisycl::vec< T, Size > >::type = T
```

Definition at line 578 of file vec.hpp.

The documentation for this struct was generated from the following file:

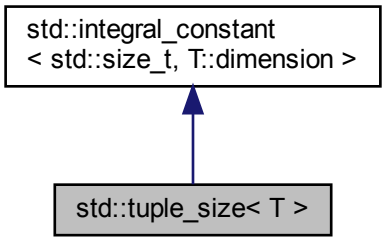
- include/triSYCL/vec.hpp

12.85 std::tuple_size< T > Struct Template Reference

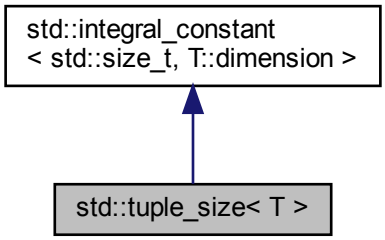
Export the dimension as its tuple size.

```
#include <small_array.hpp>
```

Inheritance diagram for std::tuple_size< T >:



Collaboration diagram for std::tuple_size< T >:



12.85.1 Detailed Description

template<typename T>
requires trisycl::detail::is_small_array<T>
struct std::tuple_size< T >

Export the dimension as its tuple size.

Definition at line 396 of file small_array.hpp.

The documentation for this struct was generated from the following file:

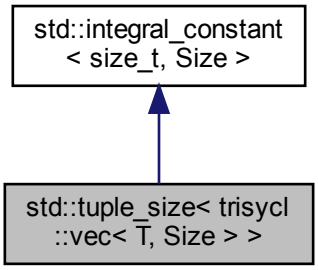
- include/triSYCL/detail/small_array.hpp

12.86 std::tuple_size< trisycl::vec< T, Size > > Struct Template Reference

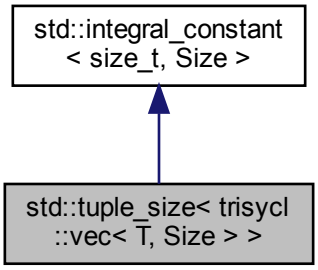
The std::tuple_size specialization for vec.

`#include <vec.hpp>`

Inheritance diagram for std::tuple_size< trisycl::vec< T, Size > >:



Collaboration diagram for std::tuple_size< trisycl::vec< T, Size > >:



12.86.1 Detailed Description

template<typename T, int Size>
struct std::tuple_size< trisycl::vec< T, Size > >

The std::tuple_size specialization for vec.

Definition at line 584 of file vec.hpp.

The documentation for this struct was generated from the following file:

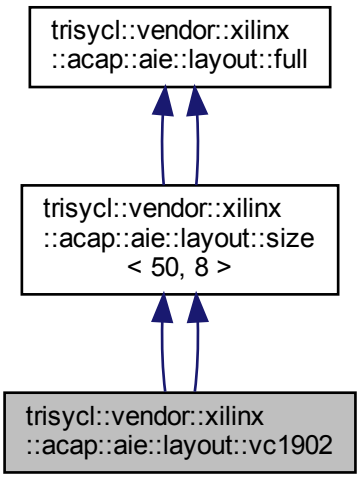
- include/triSYCL/vec.hpp

12.87 trisycl::vendor::xilinx::acap::aie::layout::vc1902 Struct Reference

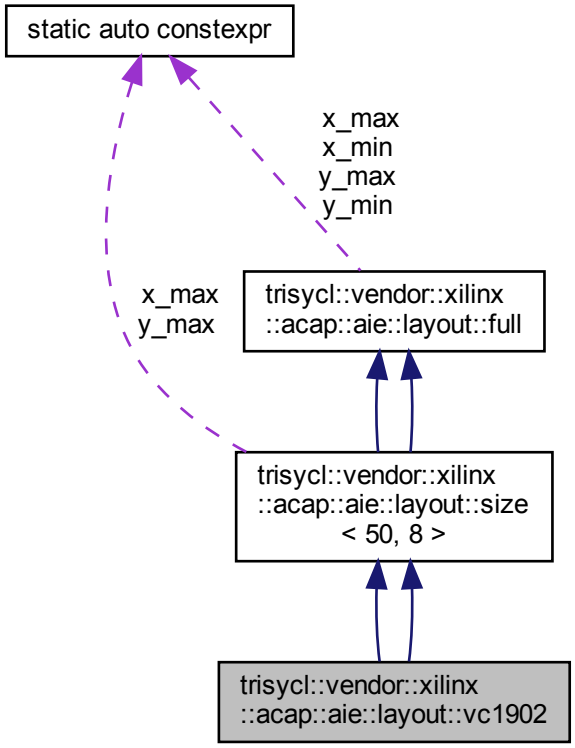
Describe the layout of a Versal VC1902 device.

`#include <versal.hpp>`

Inheritance diagram for trisycl::vendor::xilinx::acap::aie::layout::vc1902:



Collaboration diagram for trisycl::vendor::xilinx::acap::aie::layout::vc1902:



Additional Inherited Members

12.87.1 Detailed Description

Describe the layout of a Versal VC1902 device.

Definition at line 35 of file versal.hpp.

The documentation for this struct was generated from the following file:

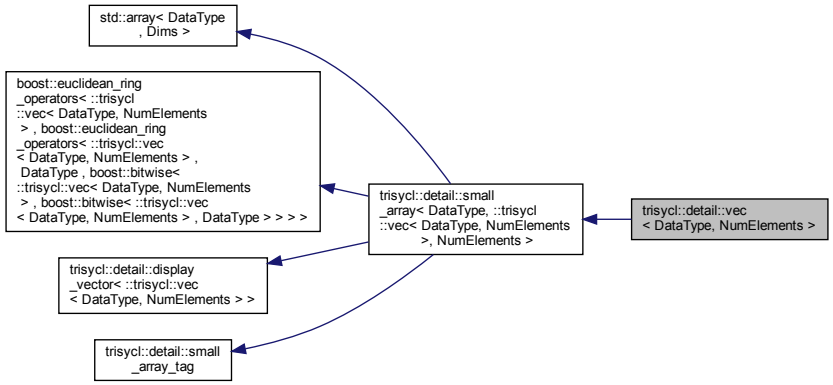
- include/sycl/vendor/Xilinx/acap/aie/layout/versal.hpp

12.88 trisycl::detail::vec< DataType, NumElements > Class Template Reference

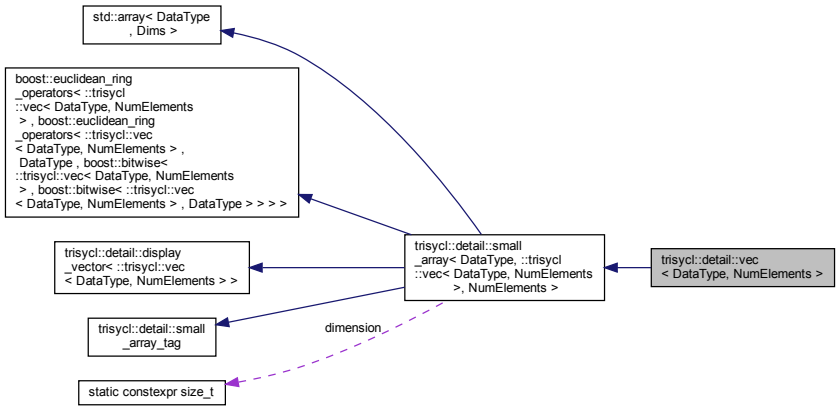
forward declaration to use in the detail class

`#include <vec.hpp>`

Inheritance diagram for trisycl::detail::vec< DataType, NumElements >:



Collaboration diagram for trisycl::detail::vec< DataType, NumElements >:



Public Member Functions

- template<typename... Types>
  requires (sizeof...(Types) != NumElements)
  vec (const Types... args)

  Construct a vec from anything from a scalar (to initialize all the elements with this value) up to an aggregate of scalar and vector types (in this case the total number of elements must match the size of the vector).
- vec ()=default

  Use classical constructors too.
- auto get_count () const

  Return the number of elements in the vector.
- auto get_size () const

  Return the number of bytes.
- template<typename convertT , rounding_mode roundingMode>
  vec< convertT, NumElements > convert () const

- template<typename asT >
  asT as () const
- template<int... swizzleIndexs>
  __swizzled_vec__< DataType, sizeof...(swizzleIndexs)> swizzle () const
- template<typename F >
  auto map (F f) const
- template<typename T , int size, typename F >
  auto zip (::trisycl::vec< T, size > x, F f) const
- template<typename T , int size, typename F >
  auto zip (::trisycl::vec< T, size > y, ::trisycl::vec< T, size > z, F f) const

Protected Member Functions

- template<typename... Ts>
  __swizzled_base_vec__< DataType, sizeof...(Ts)> swizzle (Ts... swizzleIndexes) const

  Apply a swizzle operation from swizzleIndexes parameters.

Private Types

- using basic_type = typename detail::small_array< DataType, ::trisycl::vec< DataType, NumElements >, NumElements >

Static Private Member Functions

- template<typename V , typename Element , int s>
  static auto flatten (const vec< Element, s > i)

  Flattening helper that does not change scalar values but flatten a vec<T, n> v into a tuple<T, T,..., T>{ v[0], v[1],..., v[n-1] }.
- template<typename V , typename Element , int s>
  static auto flatten (const ::trisycl::vec< Element, s > i)
- template<typename V , typename Type >
  static auto flatten (const Type i)

  If we do not have a vector, just forward it as a tuple up to the final initialization.
- template<typename V , typename... Types>
  static auto flatten_to_tuple (const Types... i)

  Take some initializer values and apply flattening on each value.
- template<typename T , typename F , std::size_t... Is>
  static auto apply_unary_functor_impl (T tuple, F f, std::index_sequence< Is... >)
- template<typename T , typename F >
  static auto apply_unary_functor (const T x, F f)
- template<typename T , typename T2 , typename F , std::size_t... Is>
  static auto apply_binary_functor_impl (T xTuple, T2 yTuple, F f, std::index_sequence< Is... >)
- template<typename T , typename T2 , typename T3 , typename F , std::size_t... Is>
  static auto apply_ternary_functor_impl (T xTuple, T2 yTuple, T3 zTuple, F f, std::index_sequence< Is... >)
- template<typename T , typename T2 , typename F >
  static auto apply_binary_functor (const T x, const T2 y, F f)

Additional Inherited Members

12.88.1 Detailed Description

template<typename DataType, int NumElements>
class trisycl::detail::vec< DataType, NumElements >

forward declaration to use in the detail class

Small SYCL vector class.

Definition at line 41 of file vec.hpp.

12.88.2 Member Typedef Documentation

12.88.2.1 basic_type

```
template<typename DataType , int NumElements>
using trisycl::detail::vec< DataType, NumElements >::basic_type = typename detail::small_array<Data↩
Type, ::trisycl::vec<DataType, NumElements>, NumElements>  [private]
```

Definition at line 45 of file vec.hpp.

12.88.3 Constructor & Destructor Documentation

12.88.3.1 vec() [1/2]

```
template<typename DataType , int NumElements>
template<typename...  Types>
requires (sizeof...(Types) != NumElements)
trisycl::detail::vec< DataType, NumElements >::vec (
            const Types...  args )  [inline]
```

Construct a vec from anything from a scalar (to initialize all the elements with this value) up to an aggregate of scalar and vector types (in this case the total number of elements must match the size of the vector).

Do not use this constructor with NumElements to avoid ambiguity with the inherited small_array constructor.

Definition at line 61 of file vec.hpp.

```
00062       :  basic_type { detail::expand<vec>(flatten_to_tuple<vec>(args...))  } {}
```

12.88.3.2 vec() [2/2]

```
template<typename DataType , int NumElements>
trisycl::detail::vec< DataType, NumElements >::vec ( )  [default]
```

Use classical constructors too.

12.88.4 Member Function Documentation

12.88.4.1 apply_binary_functor()

```
template<typename DataType , int NumElements>
template<typename T , typename T2 , typename F >
static auto trisycl::detail::vec< DataType, NumElements >::apply_binary_functor (
              const T x,
              const T2 y,
              F f )  [inline], [static], [private]
```

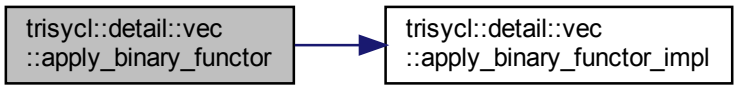
Definition at line 156 of file vec.hpp.

```
00156                                                                    {
00157     auto xTuple = flatten_to_tuple<T>(x);
00158     auto yTuple = flatten_to_tuple<T>(y);
00159     return apply_binary_functor_impl(xTuple, yTuple, f,
00160         std::make_index_sequence<std::tuple_size<decltype(xTuple)>::value>());
00161   }
```

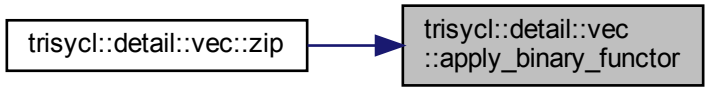
References trisycl::detail::vec< DataType, NumElements >::apply_binary_functor_impl(), x, and trisycl::y.

Referenced by trisycl::detail::vec< DataType, NumElements >::zip().

Here is the call graph for this function:



Here is the caller graph for this function:



12.88.4.2 apply_binary_functor_impl()

```
template<typename DataType , int NumElements>
template<typename T , typename T2 , typename F , std::size_t... Is>
static auto trisycl::detail::vec< DataType, NumElements >::apply_binary_functor_impl (
             T xTuple,
             T2 yTuple,
             F f,
             std::index_sequence< Is... > ) [inline], [static], [private]
```

Definition at line 140 of file vec.hpp.

```
00141                                {
00142     return ::trisycl::vec<DataType, NumElements>{f(std::get<Is>(xTuple),
00143        std::get<Is>(yTuple))...};
00144   }
```

Referenced by trisycl::detail::vec< DataType, NumElements >::apply_binary_functor().

Here is the caller graph for this function:



12.88.4.3 apply_ternary_functor_impl()

```
template<typename DataType , int NumElements>
template<typename T , typename T2 , typename T3 , typename F , std::size_t... Is>
static auto trisycl::detail::vec< DataType, NumElements >::apply_ternary_functor_impl (
             T xTuple,
             T2 yTuple,
             T3 zTuple,
             F f,
             std::index_sequence< Is... > ) [inline], [static], [private]
```

Definition at line 148 of file vec.hpp.

```
00149                                {
00150     return ::trisycl::vec<DataType, NumElements>{f(std::get<Is>(xTuple),
00151        std::get<Is>(yTuple), std::get<Is>(zTuple))...};
00152   }
```

12.88.4.4 apply_unary_functor()

```
template<typename DataType , int NumElements>
template<typename T , typename F >
static auto trisycl::detail::vec< DataType, NumElements >::apply_unary_functor (
             const T x,
             F f )  [inline], [static], [private]
```

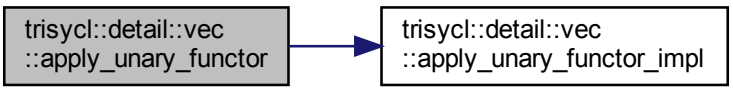
Definition at line 128 of file vec.hpp.

```
00128                                                {
00129     auto tuple = flatten_to_tuple<T>(x);
00130     return apply_unary_functor_impl(tuple, f,
00131         std::make_index_sequence<std::tuple_size<decltype(tuple)>::value>());
00132   }
```

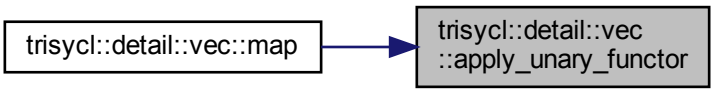
References trisycl::detail::vec< DataType, NumElements >::apply_unary_functor_impl(), and x.

Referenced by trisycl::detail::vec< DataType, NumElements >::map().

Here is the call graph for this function:



Here is the caller graph for this function:



12.88.4.5 apply_unary_functor_impl()

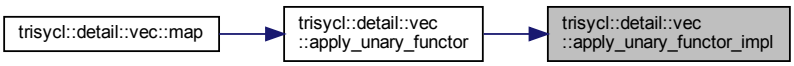
```
template<typename DataType , int NumElements>
template<typename T , typename F , std::size_t...  Is>
static auto trisycl::detail::vec< DataType, NumElements >::apply_unary_functor_impl (
             T tuple,
             F f,
             std::index_sequence< Is...  >  )  [inline], [static], [private]
```

Definition at line 122 of file vec.hpp.

```
00123                                                                    {
00124     return ::trisycl::vec<DataType, NumElements>{f(std::get<Is>(tuple))...};
00125   }
```

Referenced by trisycl::detail::vec< DataType, NumElements >::apply_unary_functor().

Here is the caller graph for this function:



12.88.4.6 as()

```
template<typename DataType , int NumElements>
template<typename asT >
asT trisycl::detail::vec< DataType, NumElements >::as ( ) const  [inline]
```

Definition at line 202 of file vec.hpp.

```
00202                                        {
00203     asT result;
00204     assert(result.get_size() == this->get_size());
00205     std::memcpy(result.data(), this->data(), this->get_size());
00206     return result;
00207   };
```

12.88.4.7 convert()

```
template<typename DataType , int NumElements>
template<typename convertT , rounding_mode roundingMode>
vec< convertT, NumElements > trisycl::detail::vec< DataType, NumElements >::convert ( ) const
[inline]
```

Definition at line 191 of file vec.hpp.

```
00191                                              {
00192     vec<convertT, NumElements> result;
00193
00194     assert(result.get_count() == this->get_count());
00195     for (int n = 0; n < NumElements; n++) {
00196       result[n] = (*this)[n];
00197     }
00198     return result;
00199   };
```

References trisycl::detail::vec< DataType, NumElements >::get_count(), and n.

Here is the call graph for this function:



12.88.4.8 flatten() [1/3]

```
template<typename DataType , int NumElements>
template<typename V , typename Element , int s>
static auto trisycl::detail::vec< DataType, NumElements >::flatten (
             const ::trisycl::vec< Element, s > i )  [inline], [static], [private]
```

Definition at line 88 of file vec.hpp.

```
00088                                                {
00089     static_assert(s <= V::dimension,
00090                   "The element i will not fit in the vector");
00091     return static_cast<std::array<Element, s>>(i);
00092   }
```

References i.

12.88.4.9 flatten() [2/3]

```
template<typename DataType , int NumElements>
template<typename V , typename Type >
static auto trisycl::detail::vec< DataType, NumElements >::flatten (
             const Type i )  [inline], [static], [private]
```

If we do not have a vector, just forward it as a tuple up to the final initialization.

Returns

typically tuple<double>{ 2.4 } from 2.4 input for example

Definition at line 101 of file vec.hpp.

```
00101                                {
00102     return std::make_tuple(i);
00103   }
```

References i.

12.88.4.10 flatten() [3/3]

```
template<typename DataType , int NumElements>
template<typename V , typename Element , int s>
static auto trisycl::detail::vec< DataType, NumElements >::flatten (
            const vec< Element, s > i )  [inline], [static], [private]
```

Flattening helper that does not change scalar values but flatten a vec<T, n> v into a tuple<T, T,..., T>{ v[0], v[1],..., v[n-1] }.

If we have a vector, just forward its array content since an array has also a tuple interface :-) (23.3.2.9 Tuple interface to class template array [array.tuple])

Definition at line 81 of file vec.hpp.

```
00081                                          {
00082     static_assert(s <= V::dimension,
00083                   "The element i will not fit in the vector");
00084     return static_cast<std::array<Element, s>>(i);
00085   }
```

References i.

12.88.4.11 flatten_to_tuple()

```
template<typename DataType , int NumElements>
template<typename V , typename...  Types>
static auto trisycl::detail::vec< DataType, NumElements >::flatten_to_tuple (
            const Types...  i )  [inline], [static], [private]
```

Take some initializer values and apply flattening on each value.

Returns

a tuple of scalar initializer values

Definition at line 111 of file vec.hpp.

```
00111                                                {
00112     // Concatenate the tuples returned by each flattening
00113     return std::tuple_cat(flatten<V>(i)...);
00114   }
```

References i.

12.88.4.12 get_count()

```
template<typename DataType , int NumElements>
auto trisycl::detail::vec< DataType, NumElements >::get_count ( ) const  [inline]
```

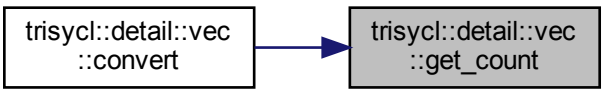
Return the number of elements in the vector.

Definition at line 177 of file vec.hpp.

```
00177                          {
00178     return NumElements;
00179   }
```

Referenced by trisycl::detail::vec< DataType, NumElements >::convert().

Here is the caller graph for this function:



12.88.4.13 get_size()

```
template<typename DataType , int NumElements>
auto trisycl::detail::vec< DataType, NumElements >::get_size ( ) const  [inline]
```

Return the number of bytes.

Definition at line 182 of file vec.hpp.

```
00182                          {
00183     // The alignment for the vec classes are equivalent to the vec's size e.g.
00184     // a vec of 4 floats would be aligned to 16 bytes which is also the vec's
00185     // size.
00186     return alignment_v<::trisycl::vec<DataType, NumElements»;
00187   }
```

12.88.4.14 map()

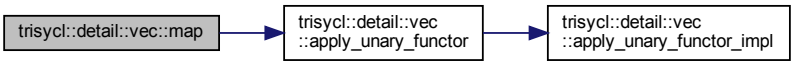
```
template<typename DataType , int NumElements>
template<typename F >
auto trisycl::detail::vec< DataType, NumElements >::map (
             F f ) const  [inline]
```

Definition at line 221 of file vec.hpp.

```
00221                        {
00222     return apply_unary_functor(*this, f);
00223   }
```

References trisycl::detail::vec< DataType, NumElements >::apply_unary_functor().

Here is the call graph for this function:



12.88.4.15 swizzle() [1/2]

```
template<typename DataType , int NumElements>
template<int...  swizzleIndexs>
__swizzled_vec__< DataType, sizeof...(swizzleIndexs)> trisycl::detail::vec< DataType, Num↩
Elements >::swizzle ( ) const  [inline]
```

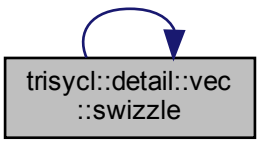
Definition at line 212 of file vec.hpp.

```
00212                                                                        {
00213     return static_cast<__swizzled_vec__<DataType, sizeof...(swizzleIndexs)>>(
00214         swizzle(swizzleIndexs...));
00215   }
```

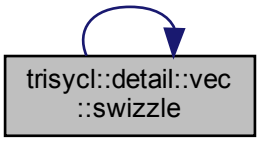
References trisycl::detail::vec< DataType, NumElements >::swizzle().

Referenced by trisycl::detail::vec< DataType, NumElements >::swizzle().

Here is the call graph for this function:



Here is the caller graph for this function:



12.88.4.16 swizzle() [2/2]

```
template<typename DataType , int NumElements>
template<typename... Ts>
__swizzled_base_vec__< DataType, sizeof...(Ts)> trisycl::detail::vec< DataType, NumElements
>::swizzle (
            Ts... swizzleIndexes ) const [inline], [protected]
```

Apply a swizzle operation from swizzleIndexes parameters.

Definition at line 168 of file vec.hpp.

```
00168                                       {
00169     // Construct a new vector from an elemental swizzle of each element
00170     return __swizzled_base_vec__<DataType, sizeof...(Ts)> {
00171       (*this)[swizzleIndexes]... };
00172   }
```

12.88.4.17 zip() [1/2]

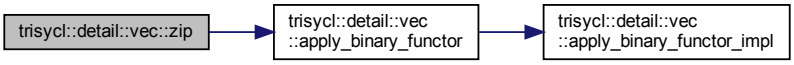
```
template<typename DataType , int NumElements>
template<typename T , int size, typename F >
auto trisycl::detail::vec< DataType, NumElements >::zip (
            ::trisycl::vec< T, size > x,
            F f ) const [inline]
```

Definition at line 235 of file vec.hpp.

```
00235                                            {
00236     static_assert(size == NumElements,
00237       "zip currently does not support vec's of differing element size");
00238     return apply_binary_functor(*this, x, f);
00239   }
```

References trisycl::detail::vec< DataType, NumElements >::apply_binary_functor(), and x.

Here is the call graph for this function:



12.88.4.18 zip() [2/2]

```
template<typename DataType , int NumElements>
template<typename T , int size, typename F >
auto trisycl::detail::vec< DataType, NumElements >::zip (
             ::trisycl::vec< T, size > y,
             ::trisycl::vec< T, size > z,
             F f ) const  [inline]
```

Todo this should be generalized to any arity

Definition at line 244 of file vec.hpp.

```
00244                                                                    {
00245       static_assert(size == NumElements,
00246         "zip currently does not support vec's of differing element size");
00247       return apply_ternary_functor(*this, y, z, f);
00248     }
```

References trisycl::y, and trisycl::z.

The documentation for this class was generated from the following file:

- include/triSYCL/vec/detail/vec.hpp

12.89 trisycl::vec< typename, int > Class Template Reference

forward declaration to use in the detail class

12.89.1 Detailed Description

template<typename, int>
class trisycl::vec< typename, int >

forward declaration to use in the detail class

Definition at line 19 of file alignment_helper.hpp.

The documentation for this class was generated from the following file:

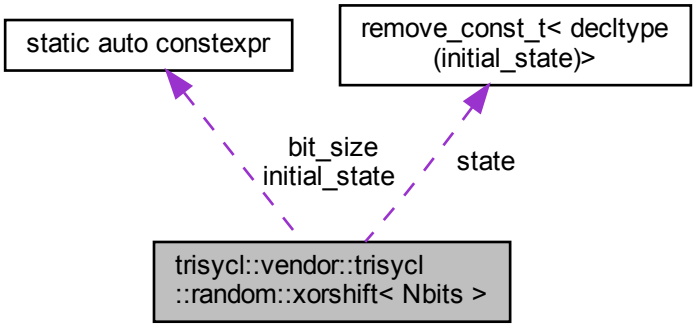
- include/triSYCL/detail/alignment_helper.hpp

12.90 trisycl::vendor::trisycl::random::xorshift< Nbits > Struct Template Reference

A possible implementation of the xorshift RNGs from.

`#include <xorshift.hpp>`

Collaboration diagram for trisycl::vendor::trisycl::random::xorshift< Nbits >:



Public Types

- using value_type = std::remove_const_t< decltype(initial_state)>

  The type of the internal state of the generator.
- using result_type = value_type

  The type of the result.
- using value_type = std::remove_const_t< decltype(initial_state)>

  The type of the internal state of the generator.
- using result_type = value_type

  The type of the result.

Public Member Functions

- xorshift (const value_type &s)

  Initialize the internal state from a user-given value.
- xorshift ()=default
- const result_type & operator() ()

  Compute a new pseudo random integer.
- xorshift (const value_type &s)

  Initialize the internal state from a user-given value.
- xorshift ()=default
- const result_type & operator() ()

  Compute a new pseudo random integer.

Static Public Member Functions

- static auto constexpr min ()

  The minimum returned value.
- static auto constexpr max ()

  The maximum returned value.
- static auto constexpr min ()

  The minimum returned value.
- static auto constexpr max ()

  The maximum returned value.

Public Attributes

- value_type state = initial_state

  The internal state of the pseudo random generator.

Static Public Attributes

- static auto constexpr bit_size = Nbits

  Bit size of the random generator data type.
- static auto constexpr initial_state

  The default initial internal state of the pseudo random generator.

12.90.1 Detailed Description

template<int Nbits = 32>
struct trisycl::vendor::trisycl::random::xorshift< Nbits >

A possible implementation of the xorshift RNGs from.

Marsaglia, George (July 2003). "Xorshift RNGs". Journal of Statistical Software. 8 (14). doi:10.18637/jss.v008.i14

Panneton, François; L'Écuyer, Pierre (October 2005). "On the xorshift random number generators" (PDF). ACM Transactions on Modeling and Computer Simulation. 15 (4):346–361. doi:10.1145/1113316.1113319

Definition at line 34 of file xorshift.hpp.

12.90.2 Member Typedef Documentation

12.90.2.1 result_type [1/2]

```
template<int Nbits = 32>
using trisycl::vendor::trisycl::random::xorshift< Nbits >::result_type = value_type
```

The type of the result.

Definition at line 60 of file xorshift.hpp.

12.90.2.2 result_type [2/2]

```
template<int Nbits = 32>
using trisycl::vendor::trisycl::random::xorshift< Nbits >::result_type = value_type
```

The type of the result.

Definition at line 60 of file xorshift.hpp.

12.90.2.3 value_type [1/2]

```
template<int Nbits = 32>
using trisycl::vendor::trisycl::random::xorshift< Nbits >::value_type = std::remove_const_↩
t<decltype(initial_state)>
```

The type of the internal state of the generator.

Definition at line 57 of file xorshift.hpp.

12.90.2.4 value_type [2/2]

```
template<int Nbits = 32>
using trisycl::vendor::trisycl::random::xorshift< Nbits >::value_type = std::remove_const_↩
t<decltype(initial_state)>
```

The type of the internal state of the generator.

Definition at line 57 of file xorshift.hpp.

12.90.3 Constructor & Destructor Documentation

12.90.3.1 xorshift() [1/4]

```
template<int Nbits = 32>
trisycl::vendor::trisycl::random::xorshift< Nbits >::xorshift (
            const value_type & s )  [inline]
```

Initialize the internal state from a user-given value.

Do not use the value 0 or the output is always 0

Definition at line 86 of file xorshift.hpp.

```
00087     : state { s }
00088   {}
```

12.90.3.2 xorshift() [2/4]

```
template<int Nbits = 32>
trisycl::vendor::trisycl::random::xorshift< Nbits >::xorshift ( )  [default]
```

12.90.3.3 xorshift() [3/4]

```
template<int Nbits = 32>
trisycl::vendor::trisycl::random::xorshift< Nbits >::xorshift (
            const value_type & s )  [inline]
```

Initialize the internal state from a user-given value.

Do not use the value 0 or the output is always 0

Definition at line 86 of file xorshift.hpp.

```
00087     : state { s }
00088   {}
```

12.90.3.4 xorshift() [4/4]

```
template<int Nbits = 32>
trisycl::vendor::trisycl::random::xorshift< Nbits >::xorshift ( )  [default]
```

12.90.4 Member Function Documentation

12.90.4.1 max() [1/2]

```
template<int Nbits = 32>
static auto constexpr trisycl::vendor::trisycl::random::xorshift< Nbits >::max ( )  [inline],
[static], [constexpr]
```

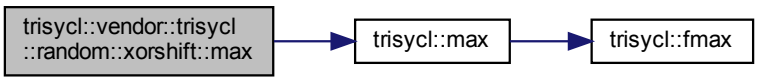
The maximum returned value.

Definition at line 77 of file xorshift.hpp.

```
00077                                {
00078     return std::numeric_limits<result_type>::max();
00079   };
```

References trisycl::max().

Here is the call graph for this function:



12.90.4.2 max() [2/2]

```
template<int Nbits = 32>
static auto constexpr trisycl::vendor::trisycl::random::xorshift< Nbits >::max ( )  [inline],
[static], [constexpr]
```

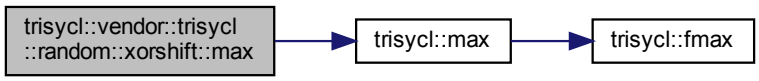
The maximum returned value.

Definition at line 77 of file xorshift.hpp.

```
00077                                {
00078     return std::numeric_limits<result_type>::max();
00079   };
```

References trisycl::max().

Here is the call graph for this function:



12.90.4.3 min() [1/2]

```
template<int Nbits = 32>
static auto constexpr trisycl::vendor::trisycl::random::xorshift< Nbits >::min ( )  [inline],
[static], [constexpr]
```

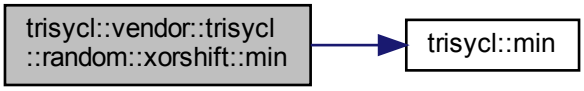
The minimum returned value.

Definition at line 70 of file xorshift.hpp.

```
00070                                {
00071     // It cannot return 0
00072     return std::numeric_limits<result_type>::min() + 1;
00073   };
```

References trisycl::min().

Here is the call graph for this function:



12.90.4.4 min() [2/2]

```
template<int Nbits = 32>
static auto constexpr trisycl::vendor::trisycl::random::xorshift< Nbits >::min ( )  [inline],
[static], [constexpr]
```

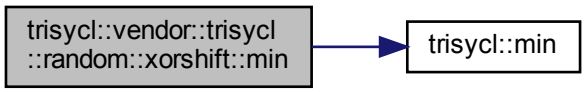
The minimum returned value.

Definition at line 70 of file xorshift.hpp.

```
00070                                {
00071     // It cannot return 0
00072     return std::numeric_limits<result_type>::min() + 1;
00073   };
```

References trisycl::min().

Here is the call graph for this function:



12.90.4.5 operator()() [1/2]

```
template<int Nbits = 32>
const result_type & trisycl::vendor::trisycl::random::xorshift< Nbits >::operator() ( )  [inline]
```

Compute a new pseudo random integer.

Definition at line 95 of file xorshift.hpp.

```
00095                                {
00096     if constexpr (bit_size == 32) {
00097       /* Pick the one of type "I" with best bit equidistribution from
00098 Panneton & L'Écuyer, Section "5.1 Equidistribution
00099 properties"
00100
00101 X4 = (I + Ra )(I + Lb )(I + Rc )
00102 */
00103       state ^= state » 7;
00104       state ^= state « 1;
00105       state ^= state » 9;
00106     }
00107     else if constexpr (bit_size == 64) {
00108       /* The xor64 from Marsaglia, p.  4 Section "3 Application to
00109 Xorshift RNGs" */
00110       state ^= state « 13;
00111       state ^= state » 7;
00112       state ^= state « 17;
00113     }
00114     else if constexpr (bit_size == 128) {
00115       // The xor128 from Marsaglia, p.  5 Section "4 Summary"
00116       auto t = state[0]^(state[0]«11);
00117       state[0] = state[1];
00118       state[1] = state[2];
00119       state[2] = state[3];
00120       state[3] = state[3]^(state[3]»19)^(t^(t»8));
```

```
00121     }
00122     else
00123       // Just to error in that case
00124       return nullptr;
00125
00126     return state;
00127   }
```

References trisycl::vendor::trisycl::random::xorshift< Nbits >::bit_size, and trisycl::vendor::trisycl::random::xorshift< Nbits >::state.

12.90.4.6 operator()() [2/2]

```
template<int Nbits = 32>
const result_type & trisycl::vendor::trisycl::random::xorshift< Nbits >::operator() ( )  [inline]
```

Compute a new pseudo random integer.

Definition at line 95 of file xorshift.hpp.

```
00095                                 {
00096     if constexpr (bit_size == 32) {
00097       /* Pick the one of type "I" with best bit equidistribution from
00098 Panneton & L'Écuyer, Section "5.1 Equidistribution
00099 properties"
00100
00101 X4 = (I + Ra )(I + Lb )(I + Rc )
00102 */
00103       state ^= state » 7;
00104       state ^= state « 1;
00105       state ^= state » 9;
00106     }
00107     else if constexpr (bit_size == 64) {
00108       /* The xor64 from Marsaglia, p.  4 Section "3 Application to
00109 Xorshift RNGs" */
00110       state ^= state « 13;
00111       state ^= state » 7;
00112       state ^= state « 17;
00113     }
00114     else if constexpr (bit_size == 128) {
00115       // The xor128 from Marsaglia, p.  5 Section "4 Summary"
00116       auto t = state[0]^(state[0]«11);
00117       state[0] = state[1];
00118       state[1] = state[2];
00119       state[2] = state[3];
00120       state[3] = state[3]^(state[3]»19)^(t^(t»8));
00121     }
00122     else
00123       // Just to error in that case
00124       return nullptr;
00125
00126     return state;
00127   }
```

References trisycl::vendor::trisycl::random::xorshift< Nbits >::bit_size, and trisycl::vendor::trisycl::random::xorshift< Nbits >::state.

12.90.5 Member Data Documentation

12.90.5.1 bit_size

```
template<int Nbits = 32>
static auto constexpr trisycl::vendor::trisycl::random::xorshift< Nbits >::bit_size = Nbits
[static], [constexpr]
```

Bit size of the random generator data type.

Definition at line 36 of file xorshift.hpp.

Referenced by trisycl::vendor::trisycl::random::xorshift< Nbits >::operator()().

12.90.5.2 initial_state

```
template<int Nbits = 32>
static auto constexpr trisycl::vendor::trisycl::random::xorshift< Nbits >::initial_state
[static], [constexpr]
```

Initial value:

```
= [] {
    if constexpr (bit_size == 32) {

      return std::uint32_t { 2463534242 };
    } else if constexpr (bit_size == 64) {

      return std::uint64_t { 88172645463325252 };
    } else if constexpr (bit_size == 128) {

      return std::array<std::uint32_t, 4>
        { 123456789, 362436069, 521288629, 88675123 };
    } else
      return nullptr;
  }()
```

The default initial internal state of the pseudo random generator.

Definition at line 39 of file xorshift.hpp.

12.90.5.3 state

```
template<int Nbits = 32>
value_type trisycl::vendor::trisycl::random::xorshift< Nbits >::state = initial_state
```

The internal state of the pseudo random generator.

Definition at line 66 of file xorshift.hpp.

Referenced by trisycl::vendor::trisycl::random::xorshift< Nbits >::operator()().

The documentation for this struct was generated from the following files:

- include/sycl/vendor/triSYCL/random/xorshift.hpp
- include/triSYCL/vendor/triSYCL/random/xorshift.hpp

Chapter 13

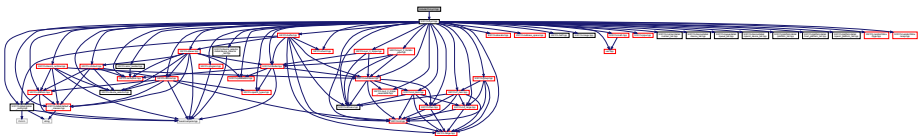
File Documentation

13.1 include/CL/sycl.hpp File Reference

This is the main SYCL 1.2.1 interoperability header to expose triSYCL into the `cl::sycl` namespace.

```
#include "triSYCL/sycl.hpp"
```

Include dependency graph for sycl.hpp:



Namespaces

- namespace cl
- namespace cl::sycl

  The official SYCL 1.2.1 specification exposes the API in the cl::sycl namespace.

Macros

- #define TRISYCL_SYCL_NAMESPACE cl::sycl

  Define what is the real SYCL namespace to use.

13.1.1 Detailed Description

This is the main SYCL 1.2.1 interoperability header to expose triSYCL into the `cl::sycl` namespace.

Definition in file sycl.hpp.

13.1.2 Macro Definition Documentation

13.1.2.1 TRISYCL_SYCL_NAMESPACE

```
#define TRISYCL_SYCL_NAMESPACE cl::sycl
```

Define what is the real SYCL namespace to use.

Definition at line 11 of file sycl.hpp.

13.2 sycl.hpp

Go to the documentation of this file.

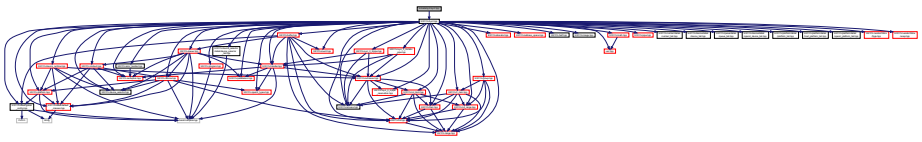
```
00001 #ifndef TRISYCL_CL_SYCL_HPP
00002 #define TRISYCL_CL_SYCL_HPP
00003
00004 /** \file
00005
00006 This is the main SYCL 1.2.1 interoperability header to expose
00007 triSYCL into the \c ::cl::sycl namespace
00008 */
00009
00010 /// Define what is the real SYCL namespace to use
00011 #define TRISYCL_SYCL_NAMESPACE cl::sycl
00012
00013 #include "triSYCL/sycl.hpp"
00014
00015 /// The official SYCL 1.2.1 specification exposes the API in the
00016 /// ::cl::sycl namespace
00017 namespace cl::sycl {
00018   // Make the triSYCL implementation appear here
00019   using namespace ::trisycl;
00020 }
00021
00022 /*
00023 # Some Emacs stuff:
00024 ### Local Variables:
00025 ### ispell-local-dictionary:  "american"
00026 ### eval:  (flyspell-prog-mode)
00027 ### End:
00028 */
00029
00030 #endif // TRISYCL_CL_SYCL_HPP
```

13.3 include/sycl/sycl.hpp File Reference

This is the SYCL extension header to expose triSYCL directly into the sycl namespace as expected by SYCL 2020.

```
#include "triSYCL/sycl.hpp"
```

Include dependency graph for sycl.hpp:



Namespaces

- namespace sycl

  Expose the SYCL API directly in the sycl namespace.

Macros

- #define TRISYCL_SYCL_NAMESPACE sycl

  Define what is the real SYCL namespace to use.

13.3.1 Detailed Description

This is the SYCL extension header to expose triSYCL directly into the sycl namespace as expected by SYCL 2020.

Definition in file sycl.hpp.

13.3.2 Macro Definition Documentation

13.3.2.1 TRISYCL_SYCL_NAMESPACE

```
#define TRISYCL_SYCL_NAMESPACE sycl
```

Define what is the real SYCL namespace to use.

Definition at line 12 of file sycl.hpp.

13.4 sycl.hpp

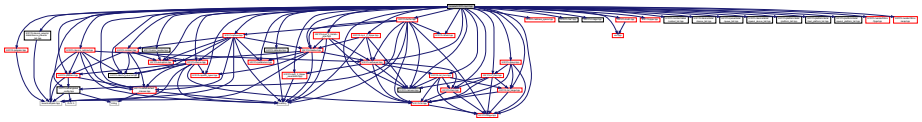
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_SYCL_HPP
00002 #define TRISYCL_SYCL_SYCL_HPP
00003
00004 /** \file
00005
00006 This is the SYCL extension header to expose triSYCL directly into
00007 the ::sycl namespace as expected by SYCL 2020
00008 */
00009
00010
00011 /// Define what is the real SYCL namespace to use
00012 #define TRISYCL_SYCL_NAMESPACE sycl
00013
00014 #include "triSYCL/sycl.hpp"
00015
00016 /// Expose the SYCL API directly in the ::sycl namespace
00017 namespace sycl {
00018   // Make the triSYCL implementation appear here
00019   using namespace ::trisycl;
00020 }
00021
00022 /*
00023 # Some Emacs stuff:
00024 ### Local Variables:
00025 ### ispell-local-dictionary:  "american"
00026 ### eval:  (flyspell-prog-mode)
00027 ### End:
00028 */
00029
00030 #endif // TRISYCL_SYCL_SYCL_HPP
```

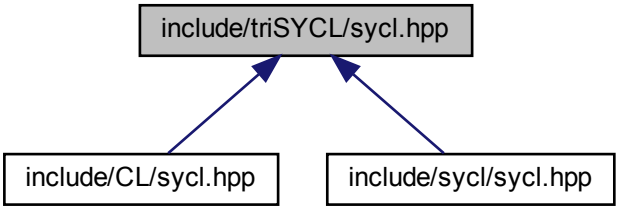
13.5 include/triSYCL/sycl.hpp File Reference

```
#include "triSYCL/detail/global_config.hpp"
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/access.hpp"
#include "triSYCL/accessor.hpp"
#include "triSYCL/allocator.hpp"
#include "triSYCL/address_space.hpp"
#include "triSYCL/buffer.hpp"
#include "triSYCL/context.hpp"
#include "triSYCL/device.hpp"
#include "triSYCL/device_runtime.hpp"
#include "triSYCL/device_selector.hpp"
#include "triSYCL/error_handler.hpp"
#include "triSYCL/event.hpp"
#include "triSYCL/exception.hpp"
#include "triSYCL/group.hpp"
#include "triSYCL/half.hpp"
#include "triSYCL/handler.hpp"
#include "triSYCL/h_item.hpp"
#include "triSYCL/id.hpp"
#include "triSYCL/image.hpp"
#include "triSYCL/item.hpp"
#include "triSYCL/math.hpp"
#include "triSYCL/nd_item.hpp"
#include "triSYCL/nd_range.hpp"
#include "triSYCL/opencl_types.hpp"
#include "triSYCL/parallelism.hpp"
#include "triSYCL/pipe.hpp"
#include "triSYCL/platform.hpp"
#include "triSYCL/program.hpp"
#include "triSYCL/queue.hpp"
#include "triSYCL/range.hpp"
#include "triSYCL/sycl_2_2/pipe.hpp"
#include "triSYCL/sycl_2_2/pipe_reservation.hpp"
#include "triSYCL/sycl_2_2/static_pipe.hpp"
#include "triSYCL/vec.hpp"
#include "triSYCL/device_selector/detail/device_selector_tail.hpp"
#include "triSYCL/context/detail/context_tail.hpp"
#include "triSYCL/device/detail/device_tail.hpp"
#include "triSYCL/queue/detail/queue_tail.hpp"
#include "triSYCL/device/detail/opencl_device_tail.hpp"
#include "triSYCL/platform/detail/platform_tail.hpp"
#include "triSYCL/platform/detail/host_platform_tail.hpp"
#include "triSYCL/platform/detail/opencl_platform_tail.hpp"
#include "triSYCL/vendor/Xilinx/fpga.hpp"
#include "triSYCL/vendor/Xilinx/acap.hpp"
```

Include dependency graph for sycl.hpp:



This graph shows which files directly or indirectly include this file:



13.6 sycl.hpp

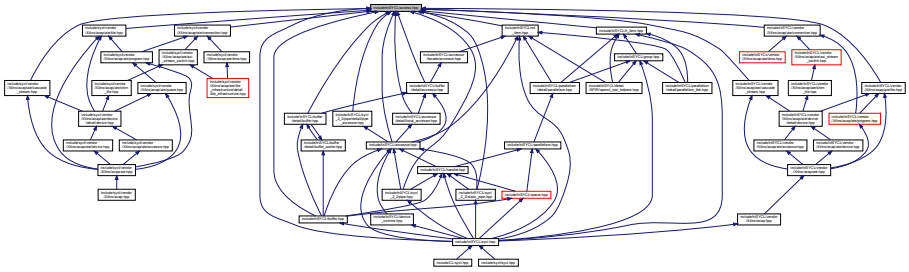
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_TRISYCL_HPP
00002 #define TRISYCL_TRISYCL_HPP
00003
00004 /** \file
00005
00006 \mainpage
00007
00008 This is the main triSYCL C++ header file to experiment with the
00009 SYCL specification.
00010
00011 It declares everything into the \c ::trisycl namespace instead of
00012 \c ::sycl for SYCL 2020 or \c ::cl::sycl for SYCL 1.2.1 to express
00013 triSYCL specific extensions and to be used in full or partially
00014 with other SYCL implementations without naming conflict.
00015
00016 You can use sycl/sycl.hpp or CL/sycl.hpp headers instead to use
00017 this implementation with the usual \c ::sycl or \c ::cl::sycl
00018 namespaces.
00019
00020 For more information about SYCL:
00021 http://www.khronos.org/sycl/
00022
00023 For more information on this project and to access to the source of
00024 this file, look at https://github.com/triSYCL/triSYCL
00025
00026 The Doxygen version of the implementation itself is in
00027 http://trisycl.github.io/triSYCL/Doxygen/triSYCL/html and
00028 http://trisycl.github.io/triSYCL/Doxygen/triSYCL/triSYCL-implementation-refman.pdf
00029
00030
00031 Ronan at keryell dot FR
00032
00033 Copyright 2014--2015 Advanced Micro Devices, Inc.
00034
00035 Copyright 2015--2022 Xilinx, Inc.
00036
00037 Copyright 2022--2023 Advanced Micro Devices, Inc.
00038
00039 This file is distributed under the University of Illinois Open Source
00040 License.  See LICENSE.TXT for details.
00041 */
00042
00043
00044 /** Some global triSYCL configuration */
00045 #include "triSYCL/detail/global_config.hpp"
00046 #include "triSYCL/detail/default_classes.hpp"
00047
00048
00049 /* All the SYCL components, one per file */
00050 #include "triSYCL/access.hpp"
00051 #include "triSYCL/accessor.hpp"
00052 #include "triSYCL/allocator.hpp"
00053 #include "triSYCL/address_space.hpp"
00054 #include "triSYCL/buffer.hpp"
```

```
00055 #include "triSYCL/context.hpp"
00056 #include "triSYCL/device.hpp"
00057 #include "triSYCL/device_runtime.hpp"
00058 #include "triSYCL/device_selector.hpp"
00059 #include "triSYCL/error_handler.hpp"
00060 #include "triSYCL/event.hpp"
00061 #include "triSYCL/exception.hpp"
00062 #include "triSYCL/group.hpp"
00063 #include "triSYCL/half.hpp"
00064 #include "triSYCL/handler.hpp"
00065 #include "triSYCL/h_item.hpp"
00066 #include "triSYCL/id.hpp"
00067 #include "triSYCL/image.hpp"
00068 #include "triSYCL/item.hpp"
00069 #include "triSYCL/math.hpp"
00070 #include "triSYCL/nd_item.hpp"
00071 #include "triSYCL/nd_range.hpp"
00072 #include "triSYCL/opencl_types.hpp"
00073 #include "triSYCL/parallelism.hpp"
00074 #include "triSYCL/pipe.hpp"
00075 #include "triSYCL/platform.hpp"
00076 #include "triSYCL/program.hpp"
00077 #include "triSYCL/queue.hpp"
00078 #include "triSYCL/range.hpp"
00079 #include "triSYCL/sycl_2_2/pipe.hpp"
00080 #include "triSYCL/sycl_2_2/pipe_reservation.hpp"
00081 #include "triSYCL/sycl_2_2/static_pipe.hpp"
00082 #include "triSYCL/vec.hpp"
00083
00084 // Some includes at the end to break some dependencies
00085 #include "triSYCL/device_selector/detail/device_selector_tail.hpp"
00086 #include "triSYCL/context/detail/context_tail.hpp"
00087 #include "triSYCL/device/detail/device_tail.hpp"
00088 #include "triSYCL/queue/detail/queue_tail.hpp"
00089 #ifdef TRISYCL_OPENCL
00090 #include "triSYCL/device/detail/opencl_device_tail.hpp"
00091 #endif
00092 #include "triSYCL/platform/detail/platform_tail.hpp"
00093 #include "triSYCL/platform/detail/host_platform_tail.hpp"
00094 #ifdef TRISYCL_OPENCL
00095 #include "triSYCL/platform/detail/opencl_platform_tail.hpp"
00096 #endif
00097
00098 // Some include files for Xilinx-specific features
00099
00100 /// For Xilinx FPGA
00101 #include "triSYCL/vendor/Xilinx/fpga.hpp"
00102
00103 /// For Xilinx ACAP
00104 #include "triSYCL/vendor/Xilinx/acap.hpp"
00105
00106 // An extension about constexpr host introspection API
00107 //#include "triSYCL/extension/ce/platform.hpp"
00108
00109 /*
00110 # Some Emacs stuff:
00111 ### Local Variables:
00112 ### ispell-local-dictionary:  "american"
00113 ### eval:  (flyspell-prog-mode)
00114 ### End:
00115 */
00116
00117 #endif // TRISYCL_TRISYCL_HPP
```

13.7 include/triSYCL/access.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::access

  Describe the type of access by kernels.

Enumerations

- enum class trisycl::access::mode {
  trisycl::access::read = 42 , trisycl::access::write , trisycl::access::read_write , trisycl::access::discard_write ,
  trisycl::access::discard_read_write , trisycl::access::atomic }

  This describes the type of the access mode to be used via accessor.
- enum class trisycl::access::target {
  trisycl::access::global_buffer = 2014 , trisycl::access::constant_buffer , trisycl::access::local , trisycl::access::image
  ,
  trisycl::access::host_buffer , trisycl::access::host_image , trisycl::access::image_array , trisycl::access::pipe ,
  trisycl::access::blocking_pipe }

  The target enumeration describes the type of object to be accessed via the accessor.
- enum class trisycl::access::fence_space : char { trisycl::access::local_space , trisycl::access::global_space ,
  trisycl::access::global_and_local }

  Precise the address space a barrier needs to act on.

13.8 access.hpp

Go to the documentation of this file.

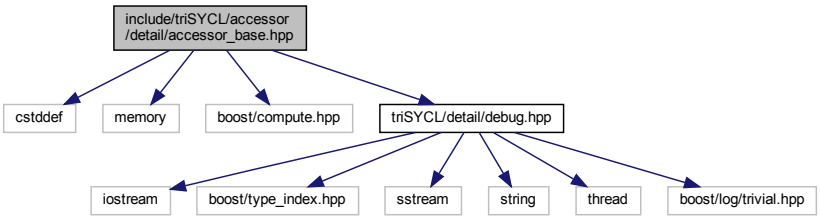
```
00001 #ifndef TRISYCL_SYCL_ACCESS_HPP
00002 #define TRISYCL_SYCL_ACCESS_HPP
00003
00004 /** \file The OpenCL SYCL access naming space
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 // SYCL dwells in the trisycl namespace
00013 namespace trisycl {
00014
```

```
00015 /** \addtogroup data Data access and storage in SYCL
00016
00017 @{
00018 */
00019
00020 /** Describe the type of access by kernels.
00021
00022 \todo This values should be normalized to allow separate compilation
00023 with different implementations?
00024 */
00025 namespace access {
00026   /* By using "enum mode" here instead of "enum struct mode", we have for
00027 example "write" appearing both as trisycl::access::mode::write and
00028 trisycl::access::write, instead of only the last one.  This seems
00029 more conform to the specification.  */
00030
00031 /// This describes the type of the access mode to be used via accessor
00032   enum class mode {
00033     read = 42, /**< Read-only access.  Insist on the fact that
00034 read_write != read + write */
00035     write, ///< Write-only access, but previous content *not* discarded
00036     read_write, ///< Read and write access
00037     discard_write, ///< Write-only access and previous content discarded
00038     discard_read_write, /**< Read and write access and previous
00039 content discarded*/
00040     atomic ///< Atomic access
00041   };
00042
00043
00044 /** The target enumeration describes the type of object to be accessed
00045 via the accessor
00046 */
00047   enum class target {
00048     global_buffer = 2014, //< Just pick a random number...
00049     constant_buffer,
00050     local,
00051     image,
00052     host_buffer,
00053     host_image,
00054     image_array,
00055     pipe,
00056     blocking_pipe
00057   };
00058
00059
00060 /** Precise the address space a barrier needs to act on
00061 */
00062   enum class fence_space :  char {
00063     local_space,
00064     global_space,
00065     global_and_local
00066   };
00067
00068 }
00069
00070 /// @} End the data Doxygen group
00071
00072 }
00073
00074 /*
00075 # Some Emacs stuff:
00076 ### Local Variables:
00077 ### ispell-local-dictionary:  "american"
00078 ### eval:  (flyspell-prog-mode)
00079 ### End:
00080 */
00081
00082 #endif // TRISYCL_SYCL_ACCESS_HPP
```

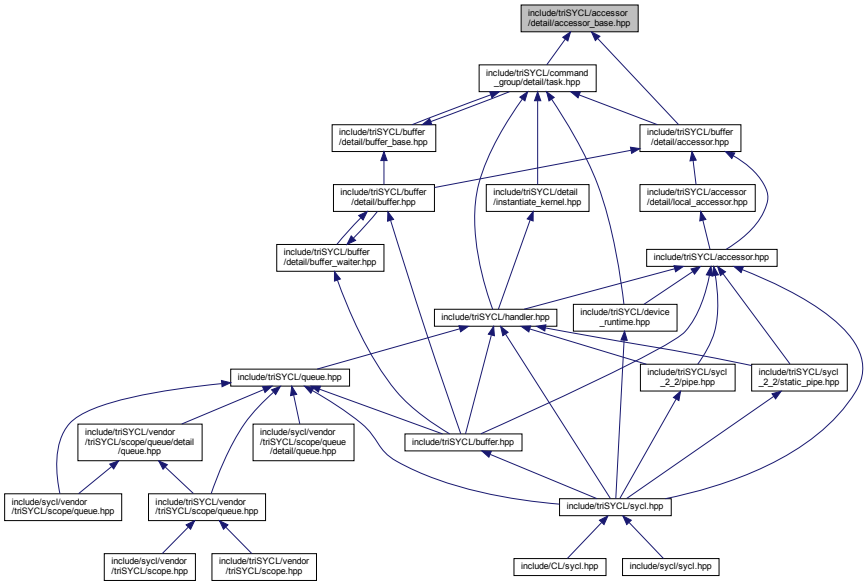
13.9 include/triSYCL/accessor/detail/accessor_base.hpp File Reference

```
#include <cstddef>
#include <memory>
#include <boost/compute.hpp>
#include "triSYCL/detail/debug.hpp"
```

Include dependency graph for accessor_base.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::accessor_base

  The buffer accessor base is here mainly to be accessed from the task without dependency on the buffer data types. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.10 accessor_base.hpp

Go to the documentation of this file.

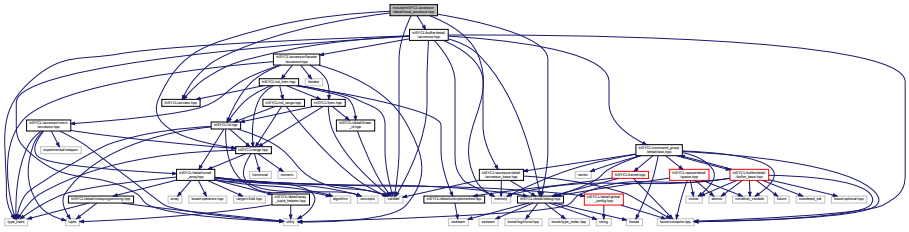
```
00001 #ifndef TRISYCL_SYCL_ACCESSOR_DETAIL_ACCESSOR_BASE_HPP
00002 #define TRISYCL_SYCL_ACCESSOR_DETAIL_ACCESSOR_BASE_HPP
00003
00004 /** \file The OpenCL SYCL buffer accessor_base<> detail behind the
00005 scene to store the data-independent of the accessor
00006
00007 Ronan at Keryell point FR
00008
00009 This file is distributed under the University of Illinois Open Source
00010 License.  See LICENSE.TXT for details.
00011 */
00012
00013 #include <cstddef>
00014 #include <memory>
00015
00016 #ifdef TRISYCL_OPENCL
00017 #include <boost/compute.hpp>
00018 #endif
00019
00020 #include "triSYCL/detail/debug.hpp"
00021
00022 namespace trisycl {
00023
00024 class handler;
00025
00026 namespace detail {
00027
00028 struct task;
00029
00030 /** \addtogroup data Data access and storage in SYCL
00031 @{
00032 */
00033
00034 /** The buffer accessor base is here mainly to be accessed from the
00035 task without dependency on the buffer data types
00036 */
00037 class accessor_base {
00038
00039 protected:
00040
00041 /// The task where the accessor is used in
00042   std::shared_ptr<detail::task> task;
00043
00044 /// The declaration order of the accessor
00045   std::size_t order;
00046
00047 public:
00048
00049 /// Set the order of the accessor
00050   void set_order(std::size_t o) {
00051     order = o;
00052   }
00053
00054
00055 /// Get the order of the accessor
00056   auto get_order() {
00057     return order;
00058   }
00059
00060
00061 #ifdef TRISYCL_OPENCL
00062 /// Get the boost::compute::buffer or throw if unset
00063   virtual boost::compute::buffer get_cl_buffer() const = 0;
00064
00065   // Virtual to call the real destructor
00066   virtual ~accessor_base() {}
00067 #endif
00068
00069 };
00070
00071 /// @} End the data Doxygen group
00072
00073 }
00074 }
00075
00076 /*
00077 # Some Emacs stuff:
00078 ### Local Variables:
00079 ### ispell-local-dictionary:  "american"
00080 ### eval:  (flyspell-prog-mode)
00081 ### End:
00082 */
```

```
00083
00084 #endif // TRISYCL_SYCL_ACCESSOR_DETAIL_ACCESSOR_BASE_HPP
```

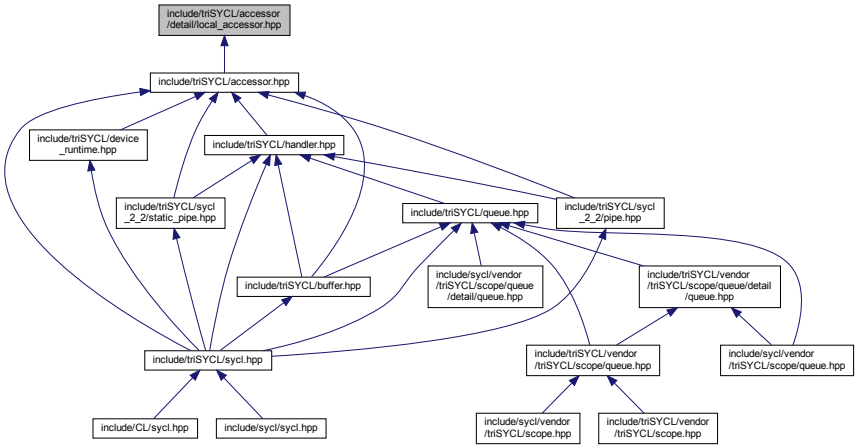
13.11 include/triSYCL/accessor/detail/local_accessor.hpp File Reference

```
#include <cstddef>
#include <memory>
#include "triSYCL/access.hpp"
#include "triSYCL/buffer/detail/accessor.hpp"
#include "triSYCL/detail/debug.hpp"
#include "triSYCL/range.hpp"
```

Include dependency graph for local_accessor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >

  The local accessor specialization abstracts the way local memory is allocated to a kernel to be shared between work-items of the same work-group. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.12 local_accessor.hpp

Go to the documentation of this file.

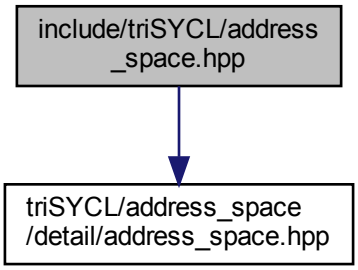
```
00001 #ifndef TRISYCL_SYCL_ACCESSOR_DETAIL_LOCAL_ACCESSOR_HPP
00002 #define TRISYCL_SYCL_ACCESSOR_DETAIL_LOCAL_ACCESSOR_HPP
00003
00004 /** \file The SYCL local accessor<> detail behind the scene for a CPU
00005 implementation
00006
00007 Ronan at Keryell point FR
00008
00009 This file is distributed under the University of Illinois Open Source
00010 License.  See LICENSE.TXT for details.
00011 */
00012
00013 #include <cstddef>
00014 #include <memory>
00015
00016 #include "triSYCL/access.hpp"
00017 #include "triSYCL/buffer/detail/accessor.hpp"
00018 #include "triSYCL/detail/debug.hpp"
00019 #include "triSYCL/range.hpp"
00020
00021 namespace trisycl {
00022
00023 class handler;
00024
00025 namespace detail {
00026
00027 // Forward declaration of detail::accessor to declare the specialization
00028 template <typename T, int Dimensions, access::mode Mode, access::target Target>
00029 class accessor;
00030
00031 /** \addtogroup data Data access and storage in SYCL
00032 @{
00033 */
00034
00035 /** The local accessor specialization abstracts the way local memory
00036 is allocated to a kernel to be shared between work-items of the
00037 same work-group.
00038
00039 Since this a pure library implementation, implement it as a
00040 host_accessor using its own local buffer.
00041
00042 \todo Implement a real local accessor handling several work-group
00043 running in parallel
00044 */
00045 template <typename T, int Dimensions, access::mode Mode>
00046 class accessor<T, Dimensions, Mode, access::target::local>
00047     :  public accessor<T, Dimensions, Mode, access::target::host_buffer>
00048     , public detail::debug<
00049           accessor<T, Dimensions, Mode, access::target::local>> {
00050   using hb = accessor<T, Dimensions, Mode, access::target::host_buffer>;
00051
00052   // Storage used by this local accessor instance
00053   std::shared_ptr<buffer<T, Dimensions>> buf;
00054
00055  public:
00056 /// Construct a local accessor of the right size
00057   accessor(const range<Dimensions>& allocation_size,
00058            handler& command_group_handler)
00059       :  buf { std::make_shared<buffer<T, Dimensions>>(allocation_size) } {
00060     this->set_buffer(buf);
00061     this->set_access(buf->access);
00062   }
00063 };
00064
00065 /// @} End the data Doxygen group
00066
00067 } // namespace detail
00068 } // namespace trisycl
00069
00070 /*
00071 # Some Emacs stuff:
00072 ### Local Variables:
00073 ### ispell-local-dictionary:  "american"
00074 ### eval:  (flyspell-prog-mode)
00075 ### End:
00076 */
00077
00078 #endif // TRISYCL_SYCL_ACCESSOR_DETAIL_LOCAL_ACCESSOR_HPP
```

13.13 include/triSYCL/address_space.hpp File Reference

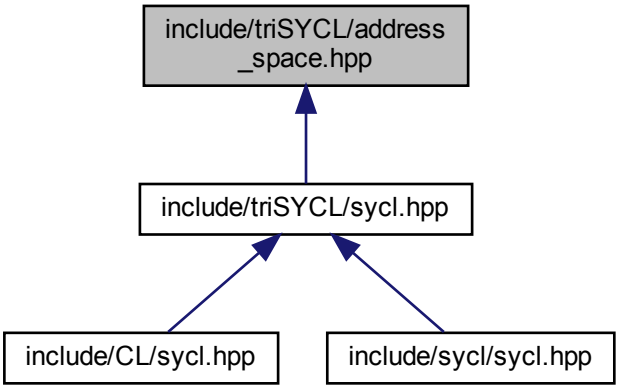
Implement OpenCL address spaces in SYCL with C++-style.

```
#include "triSYCL/address_space/detail/address_space.hpp"
```

Include dependency graph for address_space.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::access

  Describe the type of access by kernels.

Typedefs

- template<typename T >
  using trisycl::constant = detail::addr_space< T, access::address_space::constant_space >

  Declare a variable to be in the OpenCL constant address space.
- template<typename T >
  using trisycl::constant_ptr = constant< T * >

  Declare a variable to be in the OpenCL constant address space.
- template<typename T >
  using trisycl::generic = detail::addr_space< T, access::address_space::generic_space >

  Declare a variable to be in the OpenCL 2 generic address space.
- template<typename T >
  using trisycl::global = detail::addr_space< T, access::address_space::global_space >

  Declare a variable to be in the OpenCL global address space.
- template<typename T >
  using trisycl::global_ptr = global< T * >

  Declare a variable to be in the OpenCL global address space.
- template<typename T >
  using trisycl::local = detail::addr_space< T, access::address_space::local_space >

  Declare a variable to be in the OpenCL local address space.
- template<typename T >
  using trisycl::local_ptr = local< T * >

  Declare a variable to be in the OpenCL local address space.
- template<typename T >
  using trisycl::priv = detail::addr_space< T, access::address_space::private_space >

  Declare a variable to be in the OpenCL private address space.
- template<typename T >
  using trisycl::private_ptr = priv< T * >

  Declare a variable to be in the OpenCL private address space.
- template<typename Pointer , access::address_space AS>
  using trisycl::multi_ptr = detail::address_space_ptr< Pointer, AS >

  A pointer that can be statically associated to any address-space.

Enumerations

- enum class trisycl::access::address_space {
  trisycl::access::global_space , trisycl::access::local_space , trisycl::access::constant_space , trisycl::access::private_space
  ,
  trisycl::access::generic_space }

  Enumerate the different OpenCL 2 address spaces.

Functions

- template<typename T , access::address_space AS>
  multi_ptr< T, AS > trisycl::make_ptr (multi_ptr< T, AS > pointer)

  Construct a trisycl::multi_ptr<> with the right type.
- template<typename T , access::address_space AS>
  multi_ptr< T, AS > trisycl::make_ptr (T *pointer)

  Construct a trisycl::multi_ptr<> with the right type.

13.13.1 Detailed Description

Implement OpenCL address spaces in SYCL with C++-style.

Note that in SYCL 1.2, only pointer types should be specified but in this implementation we generalize the concept to any type.

Todo Add the alias ..._ptr<T> = ...<T *>

Ronan at Keryell point FR

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file address_space.hpp.

13.14 address_space.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_ADDRESS_SPACE_HPP
00002 #define TRISYCL_SYCL_ADDRESS_SPACE_HPP
00003
00004 /** \file
00005
00006 Implement OpenCL address spaces in SYCL with C++-style.
00007
00008 Note that in SYCL 1.2, only pointer types should be specified but
00009 in this implementation we generalize the concept to any type.
00010
00011 \todo Add the alias ..._ptr<T> = ...<T *>
00012
00013 Ronan at Keryell point FR
00014
00015 This file is distributed under the University of Illinois Open Source
00016 License.  See LICENSE.TXT for details.
00017 */
00018
00019 namespace trisycl {
00020
00021 /** \addtogroup address_spaces Dealing with OpenCL address spaces
00022 @{
00023 */
00024
00025 namespace access {
00026
00027 /** Enumerate the different OpenCL 2 address spaces */
00028 enum class address_space {
00029   global_space,
00030   local_space,
00031   constant_space,
00032   private_space,
00033   generic_space,
00034 };
00035
00036 }
00037 /// @} End the address_spaces Doxygen group
00038 }
00039
00040 #include "triSYCL/address_space/detail/address_space.hpp"
00041
00042
00043 namespace trisycl {
00044
00045 /** \addtogroup address_spaces
00046 @{
00047 */
00048
00049 /** Declare a variable to be in the OpenCL constant address space
00050
00051 \param T is the type of the object
00052 */
00053 template <typename T>
00054 using constant = detail::addr_space<T, access::address_space::constant_space>;
```

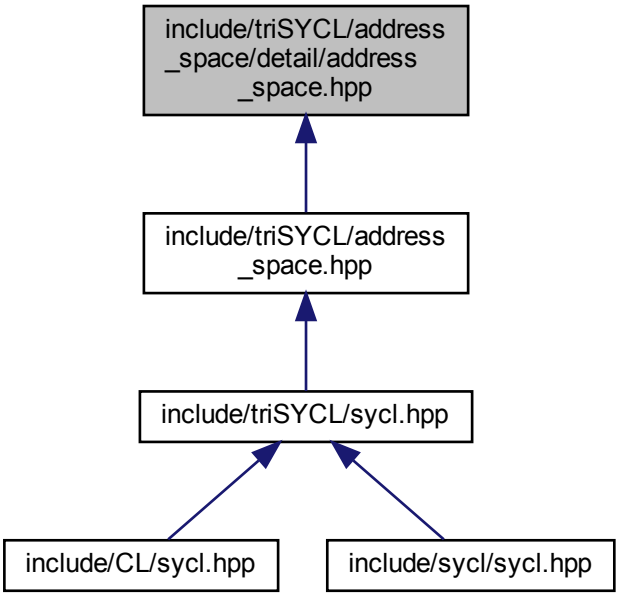
```
00055
00056
00057 /** Declare a variable to be in the OpenCL constant address space
00058
00059 \param T is the type of the object
00060 */
00061 template <typename T>
00062 using constant_ptr = constant<T*>;
00063
00064
00065 /** Declare a variable to be in the OpenCL 2 generic address space
00066
00067 \param T is the type of the object
00068 */
00069 template <typename T>
00070 using generic = detail::addr_space<T, access::address_space::generic_space>;
00071
00072
00073 /** Declare a variable to be in the OpenCL global address space
00074
00075 \param T is the type of the object
00076 */
00077 template <typename T>
00078 using global = detail::addr_space<T, access::address_space::global_space>;
00079
00080
00081 /** Declare a variable to be in the OpenCL global address space
00082
00083 \param T is the type of the object
00084 */
00085
00086 template <typename T>
00087 using global_ptr = global<T*>;
00088
00089
00090 /** Declare a variable to be in the OpenCL local address space
00091
00092 \param T is the type of the object
00093 */
00094 template <typename T>
00095 using local = detail::addr_space<T, access::address_space::local_space>;
00096
00097
00098 /** Declare a variable to be in the OpenCL local address space
00099
00100 \param T is the type of the object
00101 */
00102 template <typename T>
00103 using local_ptr = local<T*>;
00104
00105
00106 /** Declare a variable to be in the OpenCL private address space
00107
00108 \param T is the type of the object
00109 */
00110 template <typename T>
00111 using priv = detail::addr_space<T, access::address_space::private_space>;
00112
00113
00114 /** Declare a variable to be in the OpenCL private address space
00115
00116 \param T is the type of the object
00117 */
00118 template <typename T>
00119 using private_ptr = priv<T*>;
00120
00121
00122 /** A pointer that can be statically associated to any address-space
00123
00124 \param Pointer is the pointer type
00125
00126 \param AS is the address space to point to
00127
00128 Note that if \a Pointer is not a pointer type, it is an error.
00129 */
00130 template <typename Pointer, access::address_space AS>
00131 using multi_ptr = detail::address_space_ptr<Pointer, AS>;
00132
00133
00134 /** Construct a trisycl::multi_ptr<> with the right type
00135
00136 \param pointer is the address with its address space to point to
00137
00138 */
00139 template <typename T, access::address_space AS>
00140 multi_ptr<T, AS> make_ptr(multi_ptr<T, AS> pointer) {
00141   return pointer;
```

```
00142 }
00143
00144 /** Construct a trisycl::multi_ptr<> with the right type
00145
00146 \todo Implement the case with a plain pointer
00147 */
00148 template <typename T, access::address_space AS>
00149 multi_ptr<T, AS> make_ptr(T *pointer) {
00150   TRISYCL_UNIMPL;
00151   return pointer;
00152 }
00153 }
00154 /// @} End the address_spaces Doxygen group
00155
00156 /*
00157 # Some Emacs stuff:
00158 ### Local Variables:
00159 ### ispell-local-dictionary:  "american"
00160 ### eval:  (flyspell-prog-mode)
00161 ### End:
00162 */
00163
00164 #endif // TRISYCL_SYCL_ADDRESS_SPACE_HPP
```

13.15 include/triSYCL/address_space/detail/address_space.hpp File Reference

Implement OpenCL address spaces in SYCL with C++-style.

This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::detail::ocl_type< T, AS >

Generate a type with some real OpenCL 2 attribute if we are on an OpenCL device. More...

- struct trisycl::detail::ocl_type< T, access::address_space::constant_space >

  Add an attribute for __constant address space. More...
- struct trisycl::detail::ocl_type< T, access::address_space::global_space >

  Add an attribute for __global address space. More...
- struct trisycl::detail::ocl_type< T, access::address_space::local_space >

  Add an attribute for __local address space. More...
- struct trisycl::detail::ocl_type< T, access::address_space::private_space >

  Add an attribute for __private address space. More...
- struct trisycl::detail::address_space_base< T, AS >

  Implementation of the base infrastructure to wrap something in an OpenCL address space. More...
- struct trisycl::detail::address_space_variable< T, AS >

  Implementation of a variable with an OpenCL address space. More...
- struct trisycl::detail::address_space_fundamental< T, AS >

  Implementation of a fundamental type with an OpenCL address space. More...
- struct trisycl::detail::address_space_ptr< T, AS >

  Implementation for an OpenCL address space pointer. More...
- struct trisycl::detail::address_space_array< T, AS >

  Implementation of an array variable with an OpenCL address space. More...
- struct trisycl::detail::address_space_object< T, AS >

  Implementation of an object type with an OpenCL address space. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Macros

- #define TRISYCL_GLOBAL_AS TRISYCL_DEVICE_ONLY(__attribute__((address_space(1))))

  Address space for OpenCL __global.
- #define TRISYCL_LOCAL_AS TRISYCL_DEVICE_ONLY(__attribute__((address_space(3))))

  Address space for OpenCL __local.
- #define TRISYCL_CONSTANT_AS TRISYCL_DEVICE_ONLY(__attribute__((address_space(2))))

  Address space for OpenCL __constant.
- #define TRISYCL_GENERIC_AS TRISYCL_DEVICE_ONLY(__attribute__((address_space(4))))

  Address space for OpenCL __generic.
- #define TRISYCL_PRIVATE_AS

  Address space for OpenCL __private.

Typedefs

- template<typename T , access::address_space AS>
  using trisycl::detail::addr_space = typename std::conditional< std::is_pointer< T >::value, address_space_ptr< T, AS >, typename std::conditional< std::is_class< T >::value, address_space_object< T, AS >, typename std::conditional< std::is_array< T >::value, address_space_array< T, AS >, address_space_fundamental< T, AS > >::type >::type >::type

  Dispatch the address space implementation according to the requested type.

13.15.1 Detailed Description

Implement OpenCL address spaces in SYCL with C++-style.

Ronan at Keryell point FR

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file address_space.hpp.

13.15.2 Macro Definition Documentation

13.15.2.1 TRISYCL_CONSTANT_AS

```
#define TRISYCL_CONSTANT_AS  TRISYCL_DEVICE_ONLY(__attribute__((address_space(2))))
```

Address space for OpenCL __constant.

Definition at line 41 of file address_space.hpp.

13.15.2.2 TRISYCL_GENERIC_AS

```
#define TRISYCL_GENERIC_AS  TRISYCL_DEVICE_ONLY(__attribute__((address_space(4))))
```

Address space for OpenCL __generic.

Definition at line 45 of file address_space.hpp.

13.15.2.3 TRISYCL_GLOBAL_AS

```
#define TRISYCL_GLOBAL_AS TRISYCL_DEVICE_ONLY(__attribute__((address_space(1))))
```

Address space for OpenCL __global.

Definition at line 35 of file address_space.hpp.

13.15.2.4 TRISYCL_LOCAL_AS

```
#define TRISYCL_LOCAL_AS TRISYCL_DEVICE_ONLY(__attribute__((address_space(3))))
```

Address space for OpenCL __local.

Definition at line 38 of file address_space.hpp.

13.15.2.5 TRISYCL_PRIVATE_AS

```
#define TRISYCL_PRIVATE_AS
```

Address space for OpenCL __private.

Definition at line 49 of file address_space.hpp.

13.16 address_space.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_ADDRESS_SPACES_DETAIL_ADDRESS_SPACES_HPP
00002 #define TRISYCL_SYCL_ADDRESS_SPACES_DETAIL_ADDRESS_SPACES_HPP
00003
00004 /** \file
00005
00006 Implement OpenCL address spaces in SYCL with C++-style.
00007
00008 Ronan at Keryell point FR
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014
00015 /* Use macros for address spaces to be able to retarget more easily to
00016 different runtimes & compilers.
00017
00018 For example can have OpenCL address spaces even if the address
00019 space parsing is off in Clang/LLVM. Look at SPIRAddrSpaceMap in
00020 tools/clang/lib/Basic/Targets.cpp:
00021 \code
00022 static const unsigned SPIRAddrSpaceMap[] = {
00023 1, // opencl_global
00024 3, // opencl_local
00025 2, // opencl_constant
00026 4, // opencl_generic
00027 0, // cuda_device
00028 0, // cuda_constant
00029 0  // cuda_shared
00030 };
00031 \endcode
00032 */
00033
00034 /// Address space for OpenCL __global
00035 #define TRISYCL_GLOBAL_AS TRISYCL_DEVICE_ONLY(__attribute__((address_space(1))))
00036
00037 /// Address space for OpenCL __local
00038 #define TRISYCL_LOCAL_AS TRISYCL_DEVICE_ONLY(__attribute__((address_space(3))))
00039
00040 /// Address space for OpenCL __constant
00041 #define TRISYCL_CONSTANT_AS                                    \
00042 TRISYCL_DEVICE_ONLY(__attribute__((address_space(2))))
00043
00044 /// Address space for OpenCL __generic
00045 #define TRISYCL_GENERIC_AS                                     \
00046 TRISYCL_DEVICE_ONLY(__attribute__((address_space(4))))
00047
00048 /// Address space for OpenCL __private
00049 #define TRISYCL_PRIVATE_AS
00050
00051
00052 namespace trisycl::detail {
00053
00054 /** \addtogroup address_spaces
00055 @{
00056 */
00057
00058 /** Generate a type with some real OpenCL 2 attribute if we are on an
00059 OpenCL device
00060
00061 In the general case, do not add any OpenCL address space qualifier */
00062 template <typename T, access::address_space AS>
00063 struct ocl_type { // NOTE: renamed from opencl_type because of MSVC bug
00064   using type = T;
00065 };
00066
```

```
00067 /// Add an attribute for __constant address space
00068 template <typename T>
00069 struct ocl_type<T, access::address_space::constant_space> {
00070   using type = TRISYCL_CONSTANT_AS T;
00071 };
00072
00073 #if TRISYCL_CL_LANGUAGE_VERSION >= 200
00074 /// Add an attribute for __generic address space
00075 template <typename T>
00076 struct ocl_type<T, generic_space> {
00077   using type = TRISYCL_GENERIC_AS T;
00078 };
00079 #endif
00080
00081 /// Add an attribute for __global address space
00082 template <typename T>
00083 struct ocl_type<T, access::address_space::global_space> {
00084   using type = TRISYCL_GLOBAL_AS T;
00085 };
00086
00087 /// Add an attribute for __local address space
00088 template <typename T>
00089 struct ocl_type<T, access::address_space::local_space> {
00090   using type = TRISYCL_LOCAL_AS T;
00091 };
00092
00093 /// Add an attribute for __private address space
00094 template <typename T>
00095 struct ocl_type<T, access::address_space::private_space> {
00096   using type = TRISYCL_PRIVATE_AS T;
00097 };
00098
00099
00100 /* Forward declare some classes to allow some recursion in conversion
00101 operators */
00102 template <typename SomeType, access::address_space SomeAS>
00103 struct address_space_array;
00104
00105 template <typename SomeType, access::address_space SomeAS>
00106 struct address_space_fundamental;
00107
00108 template <typename SomeType, access::address_space SomeAS>
00109 struct address_space_object;
00110
00111 template <typename SomeType, access::address_space SomeAS>
00112 struct address_space_ptr;
00113
00114 /** Dispatch the address space implementation according to the requested type
00115
00116 \param T is the type of the object to be created
00117
00118 \param AS is the address space to place the object into or to point to
00119 in the case of a pointer type
00120 */
00121 template <typename T, access::address_space AS>
00122 using addr_space =
00123   typename std::conditional<std::is_pointer<T>::value,
00124                             address_space_ptr<T, AS>,
00125   typename std::conditional<std::is_class<T>::value,
00126                             address_space_object<T, AS>,
00127   typename std::conditional<std::is_array<T>::value,
00128                             address_space_array<T, AS>,
00129                             address_space_fundamental<T, AS>
00130   >::type>::type>::type;
00131
00132
00133 /** Implementation of the base infrastructure to wrap something in an
00134 OpenCL address space
00135
00136 \param T is the type of the basic stuff to be created
00137
00138 \param AS is the address space to place the object into
00139
00140 \todo Verify/improve to deal with const/volatile?
00141 */
00142 template <typename T, access::address_space AS>
00143 struct address_space_base {
00144 /** Store the base type of the object
00145
00146 \todo Add to the specification
00147 */
00148   using type = T;
00149
00150 /** Store the base type of the object with OpenCL address space modifier
00151
00152 \todo Add to the specification
00153 */
```

```
00154   using opencl_type = typename ocl_type<T, AS>::type;
00155
00156 /** Set the address_space identifier that can be queried to know the
00157 pointer type */
00158   static auto constexpr address_space = AS;
00159
00160 };
00161
00162
00163 /** Implementation of a variable with an OpenCL address space
00164
00165 \param T is the type of the basic object to be created
00166
00167 \param AS is the address space to place the object into
00168 */
00169 template <typename T, access::address_space AS>
00170 struct address_space_variable :  public address_space_base<T, AS> {
00171 /** Store the base type of the object with OpenCL address space modifier
00172
00173 \todo Add to the specification
00174 */
00175   using opencl_type = typename ocl_type<T, AS>::type;
00176
00177 /// Keep track of the base class as a short-cut
00178   using super = address_space_base<T, AS>;
00179
00180 protected:
00181
00182   /* C++11 helps a lot to be able to have the same constructors as the
00183 parent class here
00184
00185 \todo Add this to the list of required C++11 features needed for SYCL
00186 */
00187   opencl_type variable;
00188
00189 public:
00190
00191 /** Allow to create an address space version of an object or to convert
00192 one to be used by the classes inheriting by this one because it is
00193 not possible to directly initialize a base class member in C++ */
00194   address_space_variable(const T & v) :  variable(v) { }
00195
00196
00197 /// Put back the default constructors canceled by the previous definition
00198   address_space_variable() = default;
00199
00200
00201 /** Conversion operator to allow a address_space_object<T> to be used
00202 as a T so that all the methods of a T and the built-in operators for
00203 T can be used on a address_space_object<T> too.
00204
00205 Use opencl_type so that if we take the address of it, the address
00206 space is kept.
00207 */
00208   operator opencl_type & () { return variable; }
00209
00210 /// Return the address of the value to implement pointers
00211   opencl_type * get_address() { return &variable; }
00212
00213 };
00214
00215
00216 /** Implementation of a fundamental type with an OpenCL address space
00217
00218 \param T is the type of the basic object to be created
00219
00220 \param AS is the address space to place the object into
00221
00222 \todo Verify/improve to deal with const/volatile?
00223 */
00224 template <typename T, access::address_space AS>
00225 struct address_space_fundamental :  public address_space_variable<T, AS> {
00226 /// Keep track of the base class as a short-cut
00227   using super = address_space_variable<T, AS>;
00228
00229 /// Inherit from base class constructors
00230   using super::address_space_variable;
00231
00232
00233 /** Also request for the default constructors that have been disabled by
00234 the declaration of another constructor
00235
00236 This ensures for example that we can write
00237 \code
00238 generic<float *> q;
00239 \endcode
00240 without initialization.
```

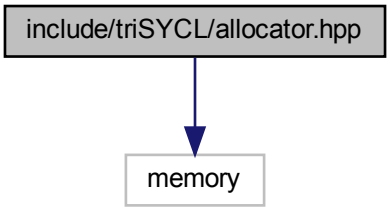
```
00241 */
00242   address_space_fundamental() = default;
00243
00244
00245 /** Allow for example assignment of a global<float> to a priv<double>
00246 for example
00247
00248 Since it needs 2 implicit conversions, it does not work with the
00249 conversion operators already define, so add 1 more explicit
00250 conversion here so that the remaining implicit conversion can be
00251 found by the compiler.
00252
00253 Strangely
00254 \code
00255 template <typename SomeType, address_space SomeAS>
00256 address_space_base(addr_space<SomeType, SomeAS>& v)
00257 :  variable(SomeType(v)) { }
00258 \endcode
00259 cannot be used here because SomeType cannot be inferred.  So use
00260 address_space_base<> instead
00261
00262 Need to think further about it...
00263 */
00264   template <typename SomeType, trisycl::access::address_space SomeAS>
00265   address_space_fundamental(address_space_fundamental<SomeType, SomeAS>& v)
00266   {
00267     /* Strangely I cannot have it working in the initializer instead, for
00268 some cases */
00269     super::variable = SomeType(v);
00270   }
00271
00272 };
00273
00274
00275 /** Implementation for an OpenCL address space pointer
00276
00277 \param T is the pointer type
00278
00279 Note that if \a T is not a pointer type, it is an error.
00280
00281 All the address space pointers inherit from it, which makes trivial
00282 the implementation of \c trisycl::multi_ptr<T, AS>
00283 */
00284 template <typename T, access::address_space AS>
00285 struct address_space_ptr :  public address_space_fundamental<T, AS> {
00286   // Verify that \a T is really a pointer
00287   static_assert(std::is_pointer<T>::value,
00288                 "T must be a pointer type");
00289
00290 /// Keep track of the base class as a short-cut
00291   using super = address_space_fundamental<T, AS>;
00292
00293 /// Inherit from base class constructors
00294   using super::address_space_fundamental;
00295
00296   using pointer_t = typename super::address_space_fundamental::type;
00297   using reference_t = typename std::remove_pointer_t<pointer_t>&;
00298
00299 /** Allow initialization of a pointer type from the address of an
00300 element with the same type and address space
00301 */
00302   address_space_ptr(address_space_fundamental<typename std::pointer_traits<T>::element_type, AS> *p)
00303     :  address_space_fundamental<T, AS> { p->get_address() } {}
00304
00305 /// Put back the default constructors canceled by the previous definition
00306   address_space_ptr() = default;
00307 };
00308
00309
00310 /** Implementation of an array variable with an OpenCL address space
00311
00312 \param T is the type of the basic object to be created
00313
00314 \param AS is the address space to place the object into
00315 */
00316 template <typename T, access::address_space AS>
00317 struct address_space_array :  public address_space_variable<T, AS> {
00318 /// Keep track of the base class as a short-cut
00319   using super = address_space_variable<T, AS>;
00320
00321 /// Inherit from base class constructors
00322   using super::address_space_variable;
00323
00324
00325 /** Allow to create an address space array from an array
00326 */
00327   address_space_array(const T &array) {
```

```
00328     std::copy(std::begin(array), std::end(array), std::begin(super::variable));
00329   };
00330
00331
00332 /** Allow to create an address space array from an initializer list
00333
00334 \todo Extend to more than 1 dimension
00335 */
00336   address_space_array(std::initializer_list<std::remove_extent_t<T» list) {
00337     std::copy(std::begin(list), std::end(list), std::begin(super::variable));
00338   };
00339
00340 };
00341
00342
00343 /** Implementation of an object type with an OpenCL address space
00344
00345 \param T is the type of the basic object to be created
00346
00347 \param AS is the address space to place the object into
00348
00349 The class implementation is just inheriting of T so that all methods
00350 and non-member operators on T work also on address_space_object<T>
00351
00352 \todo Verify/improve to deal with const/volatile?
00353
00354 \todo what about T having some final methods?
00355 */
00356 template <typename T, access::address_space AS>
00357 //struct address_space_object :  public opencl_type<T, AS>::type,
00358 struct address_space_object :  public ocl_type<T, AS>::type,
00359                                public address_space_base<T, AS> {
00360 /** Store the base type of the object with OpenCL address space modifier
00361
00362 \todo Add to the specification
00363 */
00364   using opencl_type = typename ocl_type<T, AS>::type;
00365
00366   /* C++11 helps a lot to be able to have the same constructors as the
00367 parent class here but with an OpenCL address space
00368
00369 \todo Add this to the list of required C++11 features needed for SYCL
00370 */
00371   using opencl_type::opencl_type;
00372
00373 /** Allow to create an address space version of an object or to
00374 convert one */
00375   address_space_object(T && v) :  opencl_type(v) { }
00376
00377 /** Conversion operator to allow a address_space_object<T> to be used
00378 as a T so that all the methods of a T and the built-in operators for
00379 T can be used on a address_space_object<T> too.
00380
00381 Use opencl_type so that if we take the address of it, the address
00382 space is kept.  */
00383   operator opencl_type & () { return *this; }
00384
00385 };
00386
00387 /// @} End the address_spaces Doxygen group
00388
00389 }
00390
00391 /*
00392 # Some Emacs stuff:
00393 ### Local Variables:
00394 ### ispell-local-dictionary:  "american"
00395 ### eval:  (flyspell-prog-mode)
00396 ### End:
00397 */
00398
00399 #endif // TRISYCL_SYCL_ADDRESS_SPACES_DETAIL_ADDRESS_SPACES_HPP
```

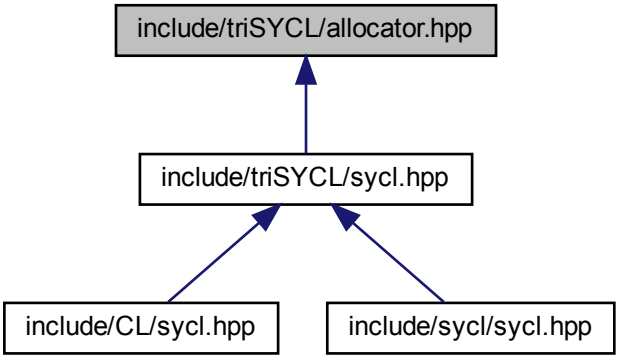
13.17 include/triSYCL/allocator.hpp File Reference

```
#include <memory>
```

Include dependency graph for allocator.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

Typedefs

- template<typename T >
  using trisycl::buffer_allocator = std::allocator< T >

  The allocator objects give the programmer some control on how the memory is allocated inside SYCL.

- template<typename T >
  using trisycl::image_allocator = std::allocator< T >

  The allocator used for the `image` inside SYCL.
- template<typename T >
  using trisycl::map_allocator = std::allocator< T >

  The allocator used to map the memory at the same place.

13.18 allocator.hpp

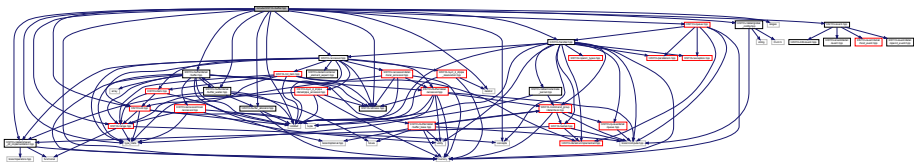
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_ALLOCATOR_HPP
00002 #define TRISYCL_SYCL_ALLOCATOR_HPP
00003
00004 /** \file The OpenCL SYCL allocator
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <memory>
00013
00014 namespace trisycl {
00015
00016 /** \addtogroup data Data access and storage in SYCL
00017 @{
00018 */
00019
00020 /** The allocator objects give the programmer some control on how the
00021 memory is allocated inside SYCL
00022 */
00023
00024 /** The allocator used for the \c buffer inside SYCL
00025
00026 Just use the default allocator for now.
00027 */
00028 template <typename T>
00029 using buffer_allocator = std::allocator<T>;
00030
00031
00032 /** The allocator used for the \c image inside SYCL
00033
00034 Just use the default allocator for now.
00035 */
00036 template <typename T>
00037 using image_allocator = std::allocator<T>;
00038
00039
00040 /** The allocator used to map the memory at the same place
00041
00042 Just use the default allocator for now.
00043
00044 \todo :  implement and clarify the specification.  It looks like it
00045 is not really an allocator according the current spec
00046 */
00047 template <typename T>
00048 using map_allocator = std::allocator<T>;
00049
00050
00051 /// @} End the data Doxygen group
00052
00053 }
00054
00055 /*
00056 # Some Emacs stuff:
00057 ### Local Variables:
00058 ### ispell-local-dictionary:  "american"
00059 ### eval:  (flyspell-prog-mode)
00060 ### End:
00061 */
00062
00063 #endif // TRISYCL_SYCL_ALLOCATOR_HPP
```

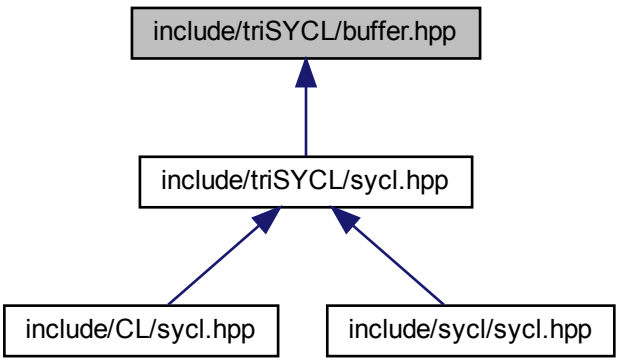
13.19 include/triSYCL/buffer.hpp File Reference

```
#include <concepts>
#include <cstddef>
#include <iterator>
#include <memory>
#include <ranges>
#include <type_traits>
#include "triSYCL/access.hpp"
#include "triSYCL/accessor.hpp"
#include "triSYCL/buffer/detail/buffer.hpp"
#include "triSYCL/buffer/detail/buffer_waiter.hpp"
#include "triSYCL/buffer_allocator.hpp"
#include "triSYCL/detail/global_config.hpp"
#include "triSYCL/detail/shared_ptr_implementation.hpp"
#include "triSYCL/event.hpp"
#include "triSYCL/handler.hpp"
#include "triSYCL/id.hpp"
#include "triSYCL/queue.hpp"
#include "triSYCL/range.hpp"
```

Include dependency graph for buffer.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::buffer< T, Dimensions, Allocator >

  A SYCL buffer is a multidimensional variable length array (à la C99 VLA or even Fortran before) that is used to store data to work on. More...
- struct std::hash< trisycl::buffer< T, Dimensions, Allocator > >

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace std

Functions

- template<std::forward_iterator FIB, std::forward_iterator FIE>
  trisycl::buffer (FIB begin, FIE end) -> buffer< typename std::iterator_traits< FIB >::value_type, 1 >

  A deduction guide to infer the buffer type from the read-write forward iterators into a copy-in on construction and copy-back on destruction.
- trisycl::buffer (auto &host_data) -> buffer< std::ranges::range_value_t< decltype(host_data)>, 1 >

  A deduction guide to infer the buffer type from the read-write range.

13.20 buffer.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_BUFFER_HPP
00002 #define TRISYCL_SYCL_BUFFER_HPP
00003
00004 /** \file The OpenCL SYCL buffer<>
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <concepts>
00013 #include <cstddef>
00014 #include <iterator>
00015 #include <memory>
00016 #include <ranges>
00017 #include <type_traits>
00018
00019 #include "triSYCL/access.hpp"
00020 #include "triSYCL/accessor.hpp"
00021 #include "triSYCL/buffer/detail/buffer.hpp"
00022 #include "triSYCL/buffer/detail/buffer_waiter.hpp"
00023 #include "triSYCL/buffer_allocator.hpp"
00024 #include "triSYCL/detail/global_config.hpp"
00025 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00026 #include "triSYCL/event.hpp"
00027 #include "triSYCL/handler.hpp"
00028 #include "triSYCL/id.hpp"
00029 #include "triSYCL/queue.hpp"
00030 #include "triSYCL/range.hpp"
00031
00032 namespace trisycl {
00033
00034 /** \addtogroup data Data access and storage in SYCL
00035 @{
00036 */
00037
00038 /** A SYCL buffer is a multidimensional variable length array (à la C99
00039 VLA or even Fortran before) that is used to store data to work on.
00040
00041 \todo There is a naming inconsistency in the specification between
00042 buffer and accessor on T versus datatype
00043
00044 \todo Finish allocator implementation
00045
00046 \todo Think about the need of an allocator when constructing a buffer
00047 from other buffers
00048
00049 \todo Update the specification to have a non-const allocator for
00050 const buffer?  Or do we rely on rebind_alloc<T>.  But does this work
00051 with astate-full allocator?
00052
00053 \todo Add constructors from arrays so that in C++17 the range and
00054 type can be infered from the constructor
```

```
00055
00056 \todo Add constructors from array_ref
00057 */
00058 template <typename T,
00059           int Dimensions = 1,
00060           /* Even a buffer of const T may need to allocate memory, so
00061 need an allocator of non const T */
00062           typename Allocator = buffer_allocator<std::remove_const_t<T>>
00063 class buffer
00064   /* Use the underlying buffer waiter implementation that can be
00065 shared in the SYCL model */
00066   :  public detail::shared_ptr_implementation<
00067                          buffer<T, Dimensions, Allocator>,
00068                          detail::buffer_waiter<T, Dimensions, Allocator>»,
00069     detail::debug<buffer<T, Dimensions, Allocator>» {
00070 public:
00071
00072 /// The STL-like types
00073   using value_type = T;
00074   using reference = value_type&;
00075   using const_reference = const value_type&;
00076   using allocator_type = Allocator;
00077
00078 /** Get the number of dimensions of the buffer
00079
00080 Name inspired from ISO C++ P0009 mdspan papers
00081 */
00082   static auto constexpr rank() { return Dimensions; }
00083
00084 private:
00085
00086   // The type encapsulating the implementation
00087   using implementation_t = typename buffer::shared_ptr_implementation;
00088
00089   // Allows the comparison operation to access the implementation
00090   friend implementation_t;
00091
00092 public:
00093
00094   // Make the implementation member directly accessible in this class
00095   using implementation_t::implementation;
00096
00097 /** Use default constructors so that we can create a new buffer copy
00098 from another one, with either a l-value or an r-value (for
00099 std::move() for example).
00100
00101 Since we just copy the shared_ptr<> from the
00102 shared_ptr_implementation above, this is where/how the sharing
00103 magic is happening with reference counting in this case.
00104 */
00105   buffer() = default;
00106
00107
00108 /** Create a new buffer of the given size with
00109 storage managed by the SYCL runtime
00110
00111 The default behavior is to use the default host buffer
00112 allocator, in order to allow for host accesses.  If the type of
00113 the buffer, has the const qualifier, then the default allocator
00114 will remove the qualifier to allow host access to the data.
00115
00116 \param[in] r defines the size
00117
00118 \param[in] allocator is to be used by the SYCL runtime
00119 */
00120   buffer(const range<Dimensions> &r, Allocator allocator = {})
00121     :  implementation_t { detail::waiter<T, Dimensions, Allocator>(
00122                          new detail::buffer<T, Dimensions> { r }) }
00123       {}
00124
00125
00126 /** Create a new buffer with associated host memory
00127
00128 \param[in] host_data points to the storage and values used by
00129 the buffer
00130
00131 \param[in] r defines the size
00132
00133 \param[in] allocator is to be used by the SYCL runtime, of type
00134 \c trisycl::buffer_allocator<T> by default
00135
00136 The host address is \code const T* \endcode, so the host memory
00137 is read-only.
00138
00139 However, the typename T is not const so the device accesses can
00140 be both read and write accesses.  Since, the host_data is const,
00141 this buffer is only initialized with this memory and there is
```

```
00142 no write after its destruction, unless there is another final
00143 data address given after construction of the buffer.
00144
00145 Only enable this constructor if it is not the same as the one
00146 with \code const T *host_data \endcode, which is when \c T is
00147 already a constant type.
00148
00149 \todo Actually this is redundant.
00150 */
00151   template <typename Dependent = T,
00152             typename = std::enable_if_t<!std::is_const<Dependent>::value»
00153   buffer(const T *host_data,
00154          const range<Dimensions> &r,
00155          Allocator allocator = {})
00156     :  implementation_t { detail::waiter(new detail::buffer<T, Dimensions>
00157                          { host_data, r }) }
00158   {}
00159
00160
00161 /** Create a new buffer with associated host memory
00162
00163 \param[inout] host_data points to the storage and values used by
00164 the buffer
00165
00166 \param[in] r defines the size
00167
00168 \param[in] allocator is to be used by the SYCL runtime, of type
00169 trisycl::buffer_allocator<T> by default
00170
00171 The memory is owned by the runtime during the lifetime of the
00172 object.    Data is copied back to the host unless the user
00173 overrides the behavior using the set_final_data method.  host_data
00174 points to the storage and values used by the buffer and
00175 range<Dimensions> defines the size.
00176 */
00177   buffer(T *host_data,
00178          const range<Dimensions> &r,
00179          Allocator allocator = {})
00180     :  implementation_t { detail::waiter(new detail::buffer<T, Dimensions>
00181                          { host_data, r }) }
00182   {}
00183
00184
00185 /** Create a new buffer with associated host memory from a range
00186
00187 \param[inout] host_data points to the storage and values used by
00188 the buffer
00189
00190 \param[in] allocator is to be used by the SYCL runtime, of type
00191 trisycl::buffer_allocator<T> by default
00192
00193 The memory is owned by the runtime during the lifetime of the
00194 object.    Data is copied back to the host unless the user
00195 overrides the behavior using the set_final_data method.
00196 */
00197   template <typename Range>
00198     requires requires {
00199                // \todo Use range concept when it lands
00200                std::begin(std::declval<Range>());
00201                std::end(std::declval<Range>());
00202              }
00203              /* But skip the case \c sycl::range<n> which is also a C++ range
00204 and we do not want to hijack the constructor from a \c
00205 sycl::range */
00206              && (!detail::is_range_v<Range>)
00207   buffer(Range /* auto std::continuous_range */& host_data,
00208          Allocator allocator = {})
00209       :  buffer { host_data.begin(),
00210                   range { std::ranges::distance(host_data) } } {}
00211
00212 /** Create a new buffer with associated memory, using the data in
00213 host_data
00214
00215 \param[inout] host_data points to the storage and values used by
00216 the buffer
00217
00218 \param[in] r defines the size
00219
00220 \param[in] allocator is to be used by the SYCL runtime, of type
00221 trisycl::buffer_allocator<T> by default
00222
00223 The ownership of the host_data is shared between the runtime and the
00224 user.  In order to enable both the user application and the SYCL
00225 runtime to use the same pointer, a trisycl::mutex_class is
00226 used.  The mutex m is locked by the runtime whenever the data is in
00227 use and unlocked otherwise.  Data is synchronized with host_data, when
00228 the mutex is unlocked by the runtime.
```

```
00229
00230 \todo update the specification to replace the pointer by a
00231 reference and provide the constructor with and without a mutex
00232 */
00233   buffer(shared_ptr_class<T> &host_data,
00234          const range<Dimensions> &buffer_range,
00235          trisycl::mutex_class &m,
00236          Allocator allocator = {}) {
00237     TRISYCL_UNIMPL;
00238   }
00239
00240
00241 /** Create a new buffer with associated memory, using the data in
00242 host_data
00243
00244 \param[inout] host_data points to the storage and values used by
00245 the buffer
00246
00247 \param[in] r defines the size
00248
00249 \param[in] allocator is to be used by the SYCL runtime, of type
00250 trisycl::buffer_allocator<T> by default
00251
00252 The ownership of the host_data is shared between the runtime and the
00253 user.  In order to enable both the user application and the SYCL
00254 runtime to use the same pointer, a trisycl::mutex_class is
00255 used.
00256
00257 \todo add this mutex-less constructor to the specification
00258 */
00259   buffer(shared_ptr_class<T> host_data,
00260          const range<Dimensions> &buffer_range,
00261          Allocator allocator = {})
00262     :  implementation_t { detail::waiter(new detail::buffer<T, Dimensions>
00263                          { host_data, buffer_range }) }
00264   {}
00265
00266
00267 /** Create a new allocated 1D buffer initialized from the given
00268 elements ranging from first up to one before last
00269
00270 The data is copied to an intermediate memory position by the
00271 runtime.  Data is written back to the same iterator set if the
00272 iterator is not a const iterator.
00273
00274 \param[inout] start_iterator points to the first element to copy
00275
00276 \param[in] end_iterator points to just after the last element to copy
00277
00278 \param[in] allocator is to be used by the SYCL runtime, of type
00279 trisycl::buffer_allocator<T> by default
00280
00281 \todo Implement the copy back at buffer destruction
00282
00283 \todo Generalize this for n-D and provide column-major and row-major
00284 initialization
00285
00286 \todo a reason to have this nD is that
00287 set_final_data(weak_ptr_class<T> & finalData) is actually
00288 doing this linearization anyway
00289
00290 \todo Allow read-only buffer construction too
00291
00292 \todo update the specification to deal with forward iterators
00293 instead and rewrite back only when it is non const and output
00294 iterator at least
00295
00296 \todo Allow initialization from ranges and collections à la STL
00297 */
00298   template <typename InputIterator,
00299             /* To force some iterator concept checking to avoid GCC 4.9
00300 diving into this when initializing from ({ int, int })
00301 which is a range<> and and not an iterator...  */
00302             typename ValueType =
00303             typename std::iterator_traits<InputIterator>::value_type>
00304   buffer(InputIterator start_iterator,
00305          InputIterator end_iterator,
00306          Allocator allocator = {}) :
00307     implementation_t { detail::waiter(new detail::buffer<T, Dimensions>
00308                        { start_iterator, end_iterator }) }
00309   {}
00310
00311
00312 /** Create a new sub-buffer without allocation to have separate
00313 accessors later
00314
00315 \param[inout] b is the buffer with the real data
```

```
00316
00317 \param[in] base_index specifies the origin of the sub-buffer inside the
00318 buffer b
00319
00320 \param[in] sub_range specifies the size of the sub-buffer
00321
00322 \todo To be implemented
00323
00324 \todo Update the specification to replace index by id
00325 */
00326   buffer(buffer<T, Dimensions, Allocator> &b,
00327          const id<Dimensions> &base_index,
00328          const range<Dimensions> &sub_range,
00329          Allocator allocator = {}) { TRISYCL_UNIMPL; }
00330
00331
00332 #ifdef TRISYCL_OPENCL
00333 /** Create a buffer from an existing OpenCL memory object associated
00334 with a context after waiting for an event signaling the
00335 availability of the OpenCL data
00336
00337 \param[inout] mem_object is the OpenCL memory object to use
00338
00339 \param[inout] from_queue is the queue associated to the memory
00340 object
00341
00342 \param[in] available_event specifies the event to wait for if
00343 non null
00344
00345 Note that a buffer created from a cl_mem object will only have
00346 one underlying cl_mem for the lifetime of the buffer and use on
00347 an incompatible queue constitues an error.
00348
00349 \todo To be implemented
00350
00351 \todo Improve the specification to allow CLHPP objects too
00352 */
00353   buffer(cl_mem mem_object,
00354          queue from_queue,
00355          event available_event = {},
00356          Allocator allocator = {}) { TRISYCL_UNIMPL;  }
00357 #endif
00358
00359
00360   // Use BOOST_DISABLE_ASSERTS at some time to disable range checking
00361
00362 /** Get an accessor to the buffer with the required mode
00363
00364 \param Mode is the requested access mode
00365
00366 \param Target is the type of object to be accessed
00367
00368 \param[in] command_group_handler is the command group handler in
00369 which the kernel is to be executed
00370
00371 \todo Do we need for an accessor to increase the reference count of
00372 a buffer object?  It does make more sense for a host-side accessor.
00373
00374 \todo Implement the modes and targets
00375 */
00376   template <access::mode Mode,
00377             access::target Target = access::target::global_buffer>
00378   accessor<T, Dimensions, Mode, Target>
00379   get_access(handler &command_group_handler) {
00380     static_assert(Target == access::target::global_buffer
00381                   || Target == access::target::constant_buffer,
00382                   "get_access(handler) can only deal with access::global_buffer"
00383                   " or access::constant_buffer (for host_buffer accessor"
00384                   " do not use a command group handler");
00385     implementation->implementation->template track_access_mode<Mode, Target>();
00386     return { *this, command_group_handler };
00387   }
00388
00389
00390 /** Force the buffer to behave like if we had created
00391 an accessor in write mode.
00392 */
00393   void mark_as_written() {
00394     return implementation->implementation->mark_as_written();
00395   }
00396
00397
00398 /** Get a host accessor to the buffer with the required mode
00399
00400 \param Mode is the requested access mode
00401
00402 \todo Implement the modes
```

```
00403
00404 \todo More elegant solution
00405 */
00406   template <access::mode Mode>
00407   accessor<T, Dimensions, Mode, access::target::host_buffer>
00408   get_access() {
00409     implementation->implementation->template track_access_mode<Mode, access::target::host_buffer>();
00410     return { *this };
00411   }
00412
00413
00414 /** Return a range object representing the size of the buffer in
00415 terms of number of elements in each dimension as passed to the
00416 constructor
00417
00418 \todo rename to the equivalent from array_ref proposals?  Such
00419 as size() in
00420 http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0009r2.html
00421 */
00422   auto get_range()const {
00423     /* Interpret the shape which is a pointer to the first element as an
00424 array of Dimensions elements so that the range<Dimensions>
00425 constructor is happy with this collection
00426 */
00427     return implementation->implementation->get_range();
00428   }
00429
00430
00431 /** Returns the total number of elements in the buffer
00432
00433 Equal to get_range()[0] * ...  * get_range()[Dimensions-1].
00434 */
00435   auto get_count()const {
00436     return implementation->implementation->get_count();
00437   }
00438
00439
00440 /** Returns the size of the buffer storage in bytes
00441
00442 Equal to get_count()*sizeof(T).
00443
00444 \todo rename to something else.  In
00445 http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/p0122r0.pdf
00446 it is named bytes() for example
00447 */
00448   size_t get_size()const {
00449     return implementation->implementation->get_size();
00450   }
00451
00452
00453 /** Returns the number of buffers that are shared/referenced
00454
00455 For example
00456 \code
00457 trisycl::buffer<int> b { 1000 };
00458 // Here b.use_count() should return 1
00459 trisycl::buffer<int> c { b };
00460 // Here b.use_count() and b.use_count() should return 2
00461 \endcode
00462
00463 \todo Add to the specification, useful for validation
00464 */
00465   auto use_count()const {
00466     // Rely on the shared_ptr<> use_count()
00467     return implementation.use_count();
00468   }
00469
00470
00471 /** Ask for read-only status of the buffer
00472
00473 \todo Add to specification
00474 */
00475   bool constexpr is_read_only()const {
00476     return std::is_const<T>::value;
00477   }
00478
00479
00480 /** Set destination of buffer data on destruction
00481
00482 The finalData points to the host memory to which, the outcome of all
00483 the buffer processing is going to be copied to.
00484
00485 This is the final pointer, which is going to be accessible after the
00486 destruction of the buffer and in the case where this is a valid
00487 pointer, the data are going to be copied to this host address.
00488
00489 finalData is different from the original host address, if the buffer
```

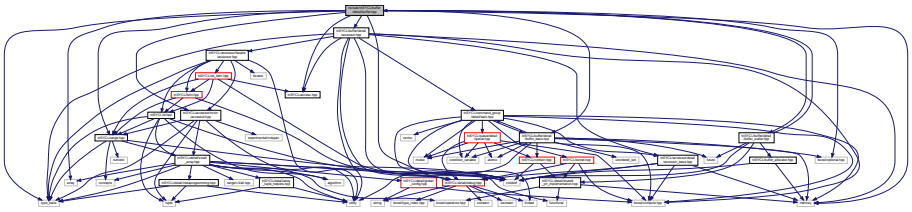
```
00490 was created associated with one.  This is mainly to be used when a
00491 shared_ptr is given in the constructor and the output data will
00492 reside in a different location from the initialization data.
00493
00494 It is defined as a weak_ptr referring to a shared_ptr that is not
00495 associated with the trisycl::buffer, and so the trisycl::buffer
00496 will have no ownership of finalData.
00497 */
00498   void set_final_data(weak_ptr_class<T> finalData) {
00499     implementation->implementation->set_final_data(std::move(finalData));
00500   }
00501
00502
00503 /** Disable write-back on buffer destruction.
00504 */
00505   void set_final_data(std::nullptr_t) {
00506     implementation->implementation->set_final_data(nullptr);
00507   }
00508
00509
00510 #ifdef TRISYCL_OPENCL
00511 /** Check if the buffer is already cached in a certain context
00512 */
00513   bool is_cached(trisycl::context& ctx) {
00514     return implementation->implementation->is_cached(ctx);
00515   }
00516
00517
00518 /** Check if the data stored in the buffer is up-to-date in a certain context
00519 */
00520   bool is_data_up_to_date(trisycl::context& ctx) {
00521     return implementation->implementation->is_data_up_to_date(ctx);
00522   }
00523 #endif
00524
00525 /** Set destination of buffer data on destruction.
00526
00527 WARNING: the user has to ensure that the object referred to by the
00528 iterator will be alive after buffer destruction, otherwise the behavior
00529 is undefined.
00530 */
00531   template <typename Iterator,
00532             typename ValueType =
00533             typename std::iterator_traits<
00534               std::remove_reference_t<Iterator»::value_type>
00535   void set_final_data(Iterator &&finalData) {
00536     implementation->implementation->set_final_data(
00537       std::forward<Iterator>(finalData));
00538   }
00539
00540 };
00541
00542 /** A deduction guide to infer the buffer type from the read-write
00543 forward iterators into a copy-in on construction and copy-back on
00544 destruction
00545
00546 \todo Use some standard concepts
00547
00548 \todo Add this to SYCL Next
00549 */
00550 template <std::forward_iterator FIB, std::forward_iterator FIE>
00551 buffer(FIB begin, FIE end)
00552     -> buffer<typename std::iterator_traits<FIB>::value_type, 1>;
00553
00554 /** A deduction guide to infer the buffer type from the read-write
00555 range
00556
00557 \todo Use some standard concepts
00558
00559 \todo Add this to SYCL Next
00560 */
00561   buffer(auto /* std::continuous_range */& host_data)
00562     -> buffer<std::ranges::range_value_t<decltype(host_data)>, 1>;
00563
00564
00565 /// @} End the data Doxygen group
00566
00567 }
00568
00569 /* Inject a custom specialization of std::hash to have the buffer
00570 usable into an unordered associative container
00571
00572 \todo Add this to the spec
00573 */
00574 namespace std {
00575
00576 template <typename T,
```

```
00577           int Dimensions,
00578           typename Allocator>
00579 struct hash<trisycl::buffer<T, Dimensions, Allocator>> {
00580
00581   auto operator()(const trisycl::buffer<T, Dimensions, Allocator> &b)const {
00582     // Forward the hashing to the implementation
00583     return b.hash();
00584   }
00585
00586 };
00587
00588 }
00589
00590 /*
00591 # Some Emacs stuff:
00592 ### Local Variables:
00593 ### ispell-local-dictionary:  "american"
00594 ### eval:  (flyspell-prog-mode)
00595 ### End:
00596 */
00597
00598 #endif // TRISYCL_SYCL_BUFFER_HPP
```

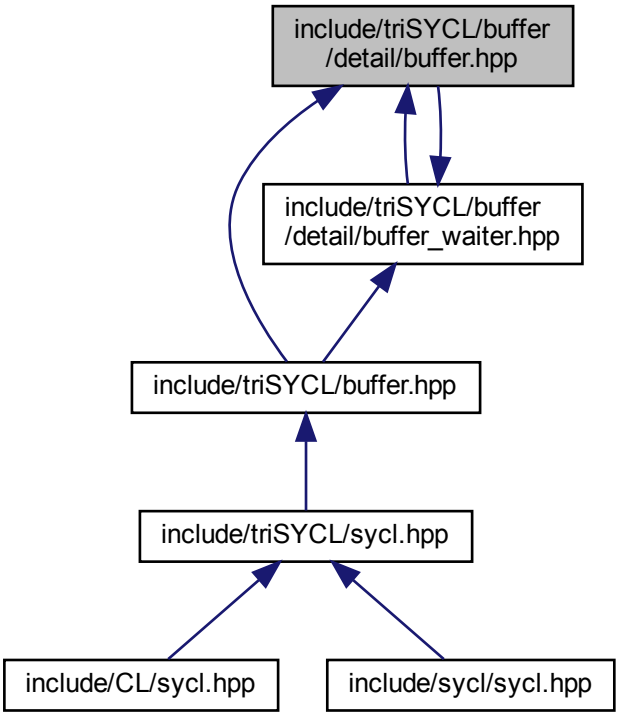
13.21 include/triSYCL/buffer/detail/buffer.hpp File Reference

```
#include <array>
#include <cstddef>
#include <memory>
#include <type_traits>
#include <boost/optional.hpp>
#include "triSYCL/access.hpp"
#include "triSYCL/accessor/mixin/accessor.hpp"
#include "triSYCL/buffer/detail/accessor.hpp"
#include "triSYCL/buffer/detail/buffer_base.hpp"
#include "triSYCL/buffer/detail/buffer_waiter.hpp"
#include "triSYCL/range.hpp"
```

Include dependency graph for buffer.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::buffer< T, Dimensions >

  A SYCL buffer is a multidimensional variable length array (à la C99 VLA or even Fortran before) that is used to store data to work on. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Functions

- template<typename BufferDetail >
  static std::shared_ptr< detail::task > trisycl::detail::buffer_add_to_task (BufferDetail buf, handler *command_group_handler, bool is_write_mode)

  Proxy function to avoid some circular type recursion.

13.22 buffer.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_BUFFER_DETAIL_BUFFER_HPP
00002 #define TRISYCL_SYCL_BUFFER_DETAIL_BUFFER_HPP
00003
00004 /** \file The OpenCL SYCL buffer<> detail implementation
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <array>
00013 #include <cstddef>
00014 #include <memory>
00015 #include <type_traits>
00016
00017 // \todo Use C++17 optional when it is mainstream
00018 #include <boost/optional.hpp>
00019
00020 #include "triSYCL/access.hpp"
00021 #include "triSYCL/accessor/mixin/accessor.hpp"
00022 #include "triSYCL/buffer/detail/accessor.hpp"
00023 #include "triSYCL/buffer/detail/buffer_base.hpp"
00024 #include "triSYCL/buffer/detail/buffer_waiter.hpp"
00025 #include "triSYCL/range.hpp"
00026
00027 namespace trisycl::detail {
00028
00029 /** \addtogroup data Data access and storage in SYCL
00030 @{
00031 */
00032
00033 /** A SYCL buffer is a multidimensional variable length array (à la C99
00034 VLA or even Fortran before) that is used to store data to work on.
00035 */
00036 template <typename T, int Dimensions = 1>
00037 class buffer
00038     :  public detail::buffer_base
00039     , public mixin::accessor<T, Dimensions>
00040     , public detail::debug<buffer<T, Dimensions>> {
00041  private:
00042   // To access directly some accessor aspects here
00043   using mixin = mixin::accessor<T, Dimensions>;
00044
00045   // \todo Replace U and D somehow by T and Dimensions
00046   // To allow allocation access
00047   template <typename U, int D, access::mode Mode,
00048             access::target Target /* = access::global_buffer */>
00049   friend class detail::accessor;
00050
00051 /** The allocator to be used when some memory is needed
00052
00053 \todo Implement user-provided allocator
00054 */
00055   std::allocator<typename mixin::value_type> alloc;
00056
00057 /** If some allocation is requested on the host for the buffer
00058 memory, this is where the memory is attached to.
00059
00060 Note that this is uninitialized memory, as stated in SYCL
00061 specification.
00062 */
00063   typename mixin::non_const_pointer allocation = nullptr;
00064
00065 /** How to copy back data on buffer destruction, can be modified with
00066 set_final_data( ...  )
00067 */
00068   boost::optional<std::function<void(void)>> final_write_back;
00069
00070 /// Keep the shared pointer used to create the buffer
00071   shared_ptr_class<T> input_shared_pointer;
00072
00073 /// Track if the buffer memory is backed by user-provided host memory
00074   bool data_host = false;
00075
00076 /** Track if data should be copied into buffer-owned memory if a
00077 modification occurs */
00078   bool copy_if_modified = false;
00079
00080   // Track if data have been modified
00081   bool modified = false;
00082
```

```
00083 /// Track the host context
00084   trisycl::context host_context { trisycl::device {} };
00085
00086  public:
00087 /// Create a new read-write buffer of size \param r
00088   buffer(const range<Dimensions>& r)
00089 /// \todo Lazily allocate memory since it might not be used on host
00090       :  mixin { allocate_buffer(r), r } {}
00091
00092 /** Create a new read-write buffer from \param host_data of size
00093 \param r without further allocation */
00094   buffer(T* host_data, const range<Dimensions>& r)
00095       :  mixin { host_data, r }
00096       , data_host { true } {}
00097
00098 /** Create a new read-only buffer from \param host_data of size \param r
00099 without further allocation
00100
00101 If the buffer is non const, use a copy-on-write mechanism with
00102 internal writable memory.
00103
00104 \todo Clarify the semantics in the spec.  What happens if the
00105 host change the host_data after buffer creation?
00106
00107 Only enable this constructor if the value type is not constant,
00108 because if it is constant, the buffer is constant too.
00109 */
00110   template <typename Dependent = T,
00111             typename = std::enable_if_t<!std::is_const<Dependent>::value»
00112   buffer(const T* host_data, const range<Dimensions>& r)
00113       :  /* The buffer is read-only, even if the internal multidimensional
00114 wrapper is not.  If a write accessor is requested, there should
00115 be a copy on write.  So this pointer should not be written and
00116 this const_cast should be acceptable.  */
00117       mixin { const_cast<T*>(host_data), r }
00118       , data_host { true }
00119       ,
00120       /* Set copy_if_modified to true, so that if an accessor with write
00121 access is created, data are copied before to be modified.  */
00122       copy_if_modified { true } {}
00123
00124 /** Create a new buffer with associated memory, using the data in
00125 host_data
00126
00127 The ownership of the host_data is shared between the runtime and the
00128 user.  In order to enable both the user application and the SYCL
00129 runtime to use the same pointer, a trisycl::mutex_class is
00130 used.
00131 */
00132   buffer(shared_ptr_class<T>& host_data, const range<Dimensions>& r)
00133       :  mixin { host_data.get(), r }
00134       , input_shared_pointer { host_data }
00135       , data_host { true } {}
00136
00137 /// Create a new allocated 1D buffer from the given elements
00138   template <typename Iterator>
00139   buffer(Iterator start_iterator, Iterator end_iterator)
00140       :  mixin { allocate_buffer(std::distance(start_iterator, end_iterator)),
00141                  range<1> { static_cast<std::size_t>(
00142                      std::distance(start_iterator, end_iterator)) } } {
00143     assign(start_iterator, end_iterator);
00144   }
00145
00146 /** Create a new sub-buffer without allocation to have separate
00147 accessors later
00148
00149 \todo To implement and deal with reference counting
00150 buffer(buffer<T, Dimensions> b,
00151 index<Dimensions> base_index,
00152 range<Dimensions> sub_range)
00153 */
00154
00155 /// \todo Allow CLHPP objects too?
00156 ///
00157   /*
00158 buffer(cl_mem mem_object,
00159 queue from_queue,
00160 event available_event)
00161 */
00162
00163 /** The buffer content may be copied back on destruction to some
00164 final location */
00165   ~buffer() {
00166 #ifdef TRISYCL_OPENCL
00167     /* We ensure that the host has the most up-to-date version of the data
00168 before the buffer is destroyed.  This is necessary because we do not
00169 systematically transfer the data back from a device with
```

```
00170 \c copy_back_cl_buffer any more.
00171 \todo Optimize for the case the buffer is not based on host memory
00172 */
00173     call_update_buffer_state(host_context, access::mode::read,
00174                              mixin::get_size(), mixin::data());
00175
00176 #endif
00177     if (modified && final_write_back)
00178       (*final_write_back)();
00179     // Allocate explicitly allocated memory if required
00180     deallocate_buffer();
00181   }
00182
00183 /** Enforce the buffer to be considered as being modified.
00184 Same as creating an accessor with write access.
00185 */
00186   void mark_as_written() { modified = true; }
00187
00188 /** This method is to be called whenever an accessor is created
00189
00190 Its current purpose is to track if an accessor with write access
00191 is created and acting accordingly.
00192 */
00193   template <access::mode Mode,
00194             access::target Target = access::target::host_buffer>
00195   void track_access_mode() {
00196     // test if write access is required
00197     if (Mode == access::mode::write || Mode == access::mode::read_write ||
00198         Mode == access::mode::discard_write ||
00199         Mode == access::mode::discard_read_write ||
00200         Mode == access::mode::atomic) {
00201       modified = true;
00202       if (copy_if_modified) {
00203         // Implement the allocate & copy-on-write optimization
00204         copy_if_modified = false;
00205         // Since the mixin store the geometry, keep a copy before updating
00206         auto current_access = mixin::access;
00207         /* The range is actually computed from \c access itself, so
00208 save it */
00209         auto current_range = mixin::get_range();
00210         allocate_buffer(current_range);
00211         /* Update the mixin accessor to point to the new allocated
00212 memory instead */
00213         mixin::update(allocation, current_range);
00214         // Then copy the read-only data to the new allocated place
00215         std::uninitialized_copy_n(current_access.data_handle(), mixin::get_count(),
00216                                   mixin::data());
00217         /* Now the data of the buffer is no longer backed-up by host
00218 user provided memory */
00219         data_host = false;
00220       }
00221     }
00222   }
00223
00224 /** Set the weak pointer as destination for write-back on buffer
00225 destruction
00226 */
00227   void set_final_data(std::weak_ptr<T> final_data) {
00228     // Capture this by reference is enough since the buffer will still exist
00229     final_write_back = [this, final_data = std::move(final_data)] {
00230       if (auto sptr = final_data.lock()) {
00231         std::copy_n(mixin::data(), mixin::get_count(), sptr.get());
00232       }
00233     };
00234   }
00235
00236 /** Disable write-back on buffer destruction as an iterator.
00237 */
00238   void set_final_data(std::nullptr_t) { final_write_back = boost::none; }
00239
00240 /** Provide destination for write-back on buffer destruction as an
00241 iterator
00242 */
00243   template <
00244       typename Iterator,
00245       typename ValueType = typename std::iterator_traits<Iterator>::value_type>
00246   void set_final_data(Iterator final_data) {
00247     /*   using type_ = typename iterator_value_type<Iterator>::value_type;
00248 static_assert(std::is_same<type_, T>::value, "buffer type mismatch");
00249 static_assert(!(std::is_const<type_>::value),
00250 "const iterator is not allowed");*/
00251     // Capture this by reference is enough since the buffer will still exist
00252     final_write_back = [this, final_data = std::move(final_data)] {
00253       std::copy_n(mixin::data(), mixin::get_count(), final_data);
00254     };
00255   }
00256
```

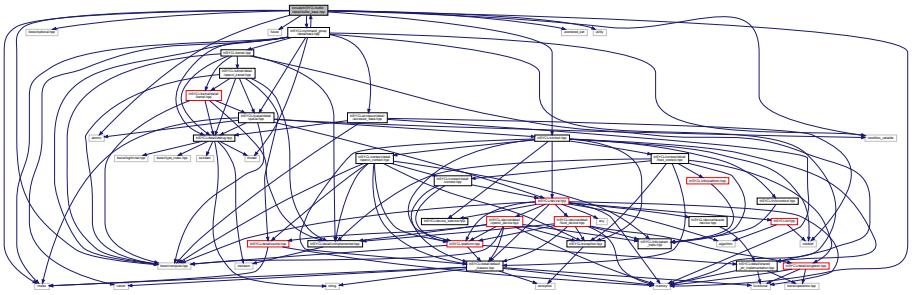
```
00257  private:
00258 /// Allocate uninitialized buffer memory
00259   auto allocate_buffer(const range<Dimensions>& r) {
00260     auto count = r.size();
00261     // Allocate uninitialized memory
00262     allocation = alloc.allocate(count);
00263     return allocation;
00264   }
00265
00266 /// Deallocate buffer memory if required
00267   void deallocate_buffer() {
00268     if (allocation)
00269       alloc.deallocate(allocation, mixin::get_count());
00270   }
00271
00272 /** Assign the 1-D storage behind the accessor
00273
00274 Use 2 different iterator types since in C++20 ranges it is now
00275 the case.
00276 */
00277   template <typename StartIter, typename EndIter>
00278   void assign(StartIter start_iterator, EndIter end_iterator) {
00279     std::copy(start_iterator, end_iterator, mixin::data());
00280   }
00281
00282 /** Function pair to work around the fact that T might be a \c const type.
00283 We call update_buffer_state only if T is not \c const, we have to
00284 use \c enable_if otherwise the compiler will try to cast \c const
00285 \c void* to \c void* if we create a buffer with a \c const type
00286
00287 \todo Use \c if \c constexpr when it is available with C++17
00288 */
00289   template <typename BaseType = T, typename DataType>
00290   void call_update_buffer_state(
00291       trisycl::context ctx, access::mode mode, size_t size, DataType* data,
00292       std::enable_if_t<!std::is_const<BaseType>::value>* = 0) {
00293     update_buffer_state(ctx, mode, size, data);
00294   }
00295
00296 /** Version of \c call_update_buffer_state that does nothing.  It is called if
00297 the type of the data in the buffer is \c const
00298 */
00299   template <typename BaseType = T, typename DataType>
00300   void call_update_buffer_state(
00301       trisycl::context ctx, access::mode mode, size_t size, DataType* data,
00302       std::enable_if_t<std::is_const<BaseType>::value>* = 0) {}
00303
00304  public:
00305 /** Get a \c future to wait from inside the \c trisycl::buffer in
00306 case there is something to copy back to the host
00307
00308 \return A \c future in the \c optional if there is something to
00309 wait for, otherwise an empty \c optional
00310 \todo Make the function private again
00311 */
00312   boost::optional<std::future<void>> get_destructor_future() {
00313     /* If there is only 1 shared_ptr user of the buffer, this is the
00314 caller of this function, the \c buffer_waiter, so there is no
00315 need to get a \ future otherwise there will be a dead-lock if
00316 there is only 1 thread waiting for itself.
00317
00318 Since \c use_count() is applied to a \c shared_ptr just created
00319 for this purpose, it actually increase locally the count by 1,
00320 so check for 1 + 1 use count instead...
00321 */
00322     // If the buffer's destruction triggers a write-back, wait
00323     if ((shared_from_this().use_count() > 2) && modified &&
00324         (final_write_back || data_host)) {
00325       // Create a promise to wait for
00326       notify_buffer_destructor = std::promise<void> {};
00327       // And return the future to wait for it
00328       return notify_buffer_destructor->get_future();
00329     }
00330     return boost::none;
00331   }
00332
00333  private:
00334   // Allow buffer_waiter destructor to access get_destructor_future()
00335   // friend detail::buffer_waiter<T, Dimensions>::~buffer_waiter();
00336   /* \todo Work around to Clang bug
00337 https://llvm.org/bugs/show_bug.cgi?id=28873 cannot use destructor
00338 here */
00339   /* \todo solve the fact that get_destructor_future is not accessible
00340 when private and buffer_waiter uses a custom allocator */
00341   friend detail::buffer_waiter<T, Dimensions>;
00342 };
00343
```

```
00344 /** Proxy function to avoid some circular type recursion
00345
00346 \return a shared_ptr<task>
00347
00348 \todo To remove with some refactoring
00349 */
00350 template <typename BufferDetail>
00351 static std::shared_ptr<detail::task>
00352 buffer_add_to_task(BufferDetail buf, handler* command_group_handler,
00353                    bool is_write_mode) {
00354   return buf->add_to_task(command_group_handler, is_write_mode);
00355 }
00356
00357 /// @} End the data Doxygen group
00358
00359 } // namespace trisycl::detail
00360
00361 /*
00362 # Some Emacs stuff:
00363 ### Local Variables:
00364 ### ispell-local-dictionary:  "american"
00365 ### eval:  (flyspell-prog-mode)
00366 ### End:
00367 */
00368
00369 #endif // TRISYCL_SYCL_BUFFER_DETAIL_BUFFER_HPP
```

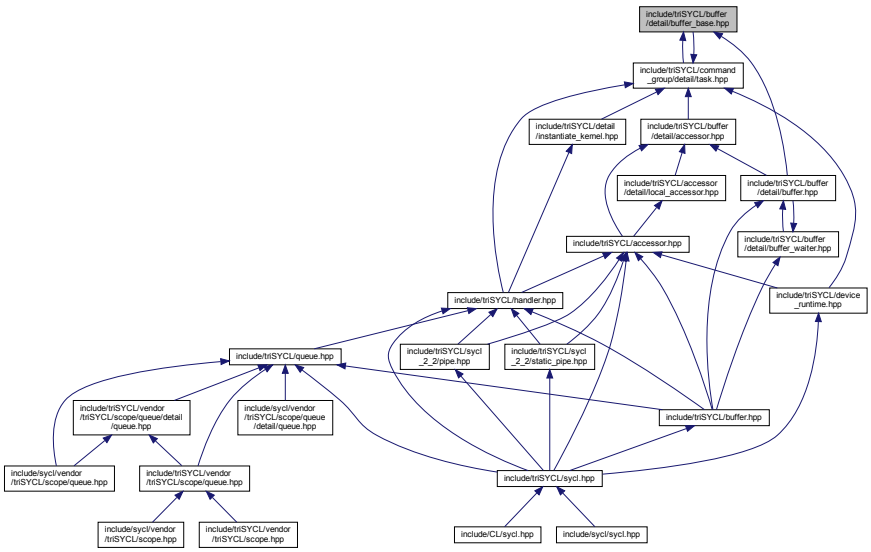
13.23 include/triSYCL/buffer/detail/buffer_base.hpp File Reference

```
#include <atomic>
#include <boost/compute.hpp>
#include <boost/optional.hpp>
#include <condition_variable>
#include <future>
#include <memory>
#include <mutex>
#include <unordered_set>
#include <utility>
#include "triSYCL/command_group/detail/task.hpp"
#include "triSYCL/context.hpp"
```

Include dependency graph for buffer_base.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::detail::buffer_base

  Factorize some template independent buffer aspects in a base class. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Functions

- static std::shared_ptr< detail::task > trisycl::detail::add_buffer_to_task (handler *command_group_handler, std::shared_ptr< detail::buffer_base > b, bool is_write_mode)

13.24 buffer_base.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_BUFFER_DETAIL_BUFFER_BASE_HPP
00002 #define TRISYCL_SYCL_BUFFER_DETAIL_BUFFER_BASE_HPP
00003
00004 /** \file The buffer_base behind the buffers, independent of the data
00005 type
00006
00007 Ronan at Keryell point FR
00008
00009 This file is distributed under the University of Illinois Open Source
00010 License.  See LICENSE.TXT for details.
00011 */
00012
```

```
00013 #include <atomic>
00014 #ifdef TRISYCL_OPENCL
00015 #include <boost/compute.hpp>
00016 #endif
00017 // \todo Use C++17 optional when it is mainstream
00018 #include <boost/optional.hpp>
00019 #include <condition_variable>
00020 #include <future>
00021 #include <memory>
00022 #include <mutex>
00023 #include <unordered_set>
00024 #include <utility>
00025
00026 #include "triSYCL/command_group/detail/task.hpp"
00027 #include "triSYCL/context.hpp"
00028
00029 namespace trisycl {
00030
00031 class handler;
00032
00033 namespace detail {
00034
00035 struct task;
00036 struct buffer_base;
00037
00038 /** \addtogroup data Data access and storage in SYCL
00039 @{
00040 */
00041
00042 inline static std::shared_ptr<detail::task>
00043 add_buffer_to_task(handler *command_group_handler,
00044                    std::shared_ptr<detail::buffer_base> b,
00045                    bool is_write_mode);
00046
00047 /** Factorize some template independent buffer aspects in a base class
00048 */
00049 struct buffer_base :  public std::enable_shared_from_this<buffer_base> {
00050
00051 //// Keep track of the number of kernel accessors using this buffer
00052   std::atomic<size_t> number_of_users;
00053
00054 /// Track the latest task to produce this buffer
00055   std::weak_ptr<detail::task> latest_producer;
00056 /// To protect the access to latest_producer
00057   std::mutex latest_producer_mutex;
00058
00059 /// To signal when this buffer ready
00060   std::condition_variable ready;
00061 /// To protect the access to the condition variable
00062   std::mutex ready_mutex;
00063
00064 /** If the SYCL user buffer destructor is blocking, use this to
00065 block until this buffer implementation is destroyed.
00066
00067 Use a void promise since there is no value to send, only
00068 waiting */
00069   boost::optional<std::promise<void>> notify_buffer_destructor;
00070
00071 /// To track contexts in which the data is up-to-date
00072   std::unordered_set<trisycl::context> fresh_ctx;
00073
00074 #ifdef TRISYCL_OPENCL
00075 /** Buffer-side cache that keeps the \c boost::compute::buffer (and the
00076 underlying \c cl_buffer ) so that if the buffer already exists inside
00077 the same context it is not recreated.
00078 */
00079   std::unordered_map<trisycl::context, boost::compute::buffer> buffer_cache;
00080 #endif
00081
00082 /** Create a buffer base and marks the host context as the context that
00083 holds the most recent version of the data
00084 \todo Use lazy allocation for the context tracking set
00085 */
00086   buffer_base() :  number_of_users { 0 },
00087                   fresh_ctx { trisycl::context {} } {}
00088
00089
00090 /// The destructor waits for not being used anymore
00091   ~buffer_base() {
00092     wait();
00093     // If there is the last SYCL user buffer waiting, notify it
00094     if (notify_buffer_destructor)
00095       notify_buffer_destructor->set_value();
00096   }
00097
00098
00099 /// Wait for this buffer to be ready, which is no longer in use
```

```
00100   void wait() {
00101     std::unique_lock<std::mutex> ul { ready_mutex };
00102     ready.wait(ul, [&] {
00103         // When there is no producer for this buffer, we are ready to use it
00104         return number_of_users == 0;
00105       });
00106   }
00107
00108
00109 /// Mark this buffer in use by a task
00110   void use() {
00111     // Increment the use count
00112     ++number_of_users;
00113   }
00114
00115
00116 /// A task has released the buffer
00117   void release() {
00118     std::unique_lock<std::mutex> lock { ready_mutex };
00119     if (--number_of_users == 0) {
00120       // Micro-optimization:  unlock before the notification
00121       // https://en.cppreference.com/w/cpp/thread/condition_variable/notify_all
00122       lock.unlock();
00123       // Notify the host consumers or the buffer destructor that it is ready
00124       ready.notify_all();
00125     }
00126   }
00127
00128
00129 /// Return the latest producer for the buffer
00130   std::shared_ptr<detail::task> get_latest_producer() {
00131     std::lock_guard<std::mutex> lg { latest_producer_mutex };
00132     // Return the valid shared_ptr to the task, if any
00133     return latest_producer.lock();
00134   }
00135
00136
00137 /** Return the latest producer for the buffer and set another
00138 future producer
00139 */
00140   std::shared_ptr<detail::task>
00141   set_latest_producer(std::weak_ptr<detail::task> newer_latest_producer) {
00142     std::lock_guard<std::mutex> lg { latest_producer_mutex };
00143     using std::swap;
00144
00145     swap(newer_latest_producer, latest_producer);
00146     // Return the valid shared_ptr to the previous producing task, if any
00147     return newer_latest_producer.lock();
00148   }
00149
00150
00151 /// Add a buffer to the task running the command group
00152   std::shared_ptr<detail::task>
00153   add_to_task(handler *command_group_handler, bool is_write_mode) {
00154     return add_buffer_to_task(command_group_handler,
00155                               shared_from_this(),
00156                               is_write_mode);
00157   }
00158
00159
00160 #ifdef TRISYCL_OPENCL
00161 /// Check if the data of this buffer is up-to-date in a certain context
00162   bool is_data_up_to_date(const trisycl::context& ctx) {
00163     return fresh_ctx.count(ctx);
00164   }
00165
00166
00167 /// Check if the buffer is already cached for a certain context
00168   bool is_cached(const trisycl::context& ctx) {
00169     return buffer_cache.count(ctx);
00170   }
00171
00172
00173 /** Create a \c boost::compute::buffer for this \c trisycl::buffer in the
00174 cache and associate it with a given context
00175 */
00176   void create_in_cache(const trisycl::context& ctx, size_t size,
00177                        cl_mem_flags flags, void* data) {
00178     buffer_cache[ctx] = boost::compute::buffer
00179       { ctx.get_boost_compute(),
00180         size,
00181         flags,
00182         data
00183       };
00184   }
00185
00186
```

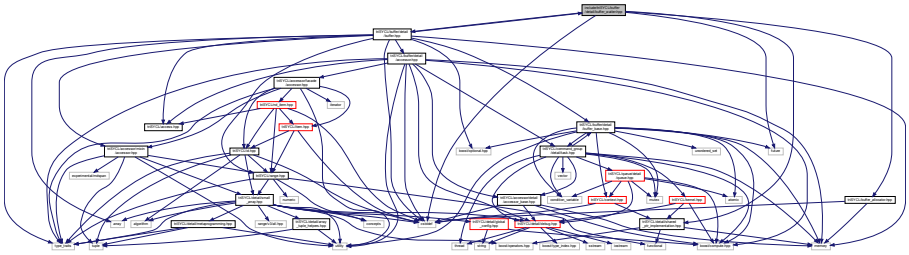
```
00187 /** Transfer the most up-to-date version of the data to the host
00188 if the host version is not already up-to-date
00189 */
00190   void sync_with_host(std::size_t size, void* data) {
00191     trisycl::context host_context;
00192     if (!is_data_up_to_date(host_context) && !fresh_ctx.empty()) {
00193       /* We know that the context(s) in \c fresh_ctx hold the most recent
00194 version of the buffer
00195 */
00196       auto fresh_context = *(fresh_ctx.begin());
00197       auto fresh_q = fresh_context.get_boost_queue();
00198       fresh_q.enqueue_read_buffer(buffer_cache[fresh_context], 0, size, data);
00199       fresh_ctx.insert(host_context);
00200     }
00201   }
00202
00203
00204 /** When a transfer is requested this function is called, it will
00205 update the state of the buffer according to the context in which
00206 the accessor is created and the access mode
00207 */
00208   void update_buffer_state(const trisycl::context& target_ctx,
00209                            access::mode mode, std::size_t size, void* data) {
00210     /* The \c cl_buffer we put in the cache might get accessed again in the
00211 future, this means that we have to always to create it in read/write
00212 mode to be able to write to it if it is accessed through a
00213 write accessor in the future
00214 */
00215     auto constexpr flag = CL_MEM_READ_WRITE;
00216
00217     /* The buffer is accessed in read mode, we want to transfer the data only if
00218 necessary.  We start a transfer if the data on the target context is not
00219 up to date and then update the fresh context set.
00220 */
00221     if (mode == access::mode::read) {
00222
00223       if (is_data_up_to_date(target_ctx))
00224         // If read mode and the data is up-to-date there is nothing to do
00225         return;
00226
00227       // The data is not up-to-date, we need a transfer
00228       // We also want to be sure that the host holds the most recent data
00229       sync_with_host(size, data);
00230
00231       if (!target_ctx.is_host()) {
00232         // If the target context is a device context
00233         if (!is_cached(target_ctx)) {
00234           /* If not cached, we create the buffer and copy the data
00235 at the same time
00236 */
00237           create_in_cache(target_ctx, size,
00238                           (flag | CL_MEM_COPY_HOST_PTR), data);
00239           fresh_ctx.insert(target_ctx);
00240           return;
00241         }
00242
00243         /* Else we transfer the data to the existing buffer associated
00244 with the target context buffer
00245 */
00246         auto q = target_ctx.get_boost_queue();
00247         q.enqueue_write_buffer(buffer_cache[target_ctx], 0, size, data);
00248         fresh_ctx.insert(target_ctx);
00249       }
00250       return;
00251     }
00252
00253     /* The buffer might be written to, this means that we have to consider
00254 every version of the data obsolete except in the target context
00255
00256 We go through the same process as in read mode but in addition
00257 we empty the fresh context set and just add the target context
00258
00259 If the data is up to date on the target we just have to update
00260 the context set and nothing else
00261 */
00262     if (!is_data_up_to_date(target_ctx)) {
00263
00264       if (   mode == access::mode::read_write
00265           || mode == access::mode::write
00266           || mode == access::mode::atomic) {
00267         // If the data is not up-to-date in the target context
00268         // We want to host to be up-to-date
00269         sync_with_host(size, data);
00270
00271         if (!target_ctx.is_host()) {
00272           // If the target context is a device context
00273           if (!is_cached(target_ctx)) {
```

```
00274             create_in_cache(target_ctx, size,
00275                             (flag | CL_MEM_COPY_HOST_PTR), data);
00276           }
00277           else {
00278             // We update the buffer associated with the target context
00279             auto q = target_ctx.get_boost_queue();
00280             q.enqueue_write_buffer(buffer_cache[target_ctx], 0, size, data);
00281           }
00282         }
00283       }
00284
00285       /* When in discard mode we don't need to transfer any data, we just create
00286 the \c cl_buffer if it doesn't exist in the cache
00287 */
00288       if (   mode == access::mode::discard_write
00289           || mode == access::mode::discard_read_write) {
00290         /* We only need to create the buffer if it doesn't exist
00291 but without copying any data because of the discard mode
00292 */
00293         if (!target_ctx.is_host() && !is_cached(target_ctx)) {
00294           // If the context doesn't exist we create it.
00295           /* We don't want to transfer any data so we don't
00296 add \c CL_MEM_COPY_HOST_PTR
00297 */
00298           create_in_cache(target_ctx, size, flag, 0);
00299         }
00300       }
00301     }
00302     /* Here we are sure that we are in some kind of write mode,
00303 we indicate that all contexts except the target context
00304 are not up-to-date anymore
00305 */
00306     fresh_ctx.clear();
00307     fresh_ctx.insert(target_ctx);
00308   }
00309
00310
00311 /// Returns the cl_buffer for a given context.
00312   boost::compute::buffer get_cl_buffer(const trisycl::context& context) {
00313     return buffer_cache[context];
00314   }
00315
00316 #endif
00317
00318 };
00319
00320 /// @} End the data Doxygen group
00321
00322 }
00323 }
00324
00325 /*
00326 # Some Emacs stuff:
00327 ### Local Variables:
00328 ### ispell-local-dictionary:  "american"
00329 ### eval:  (flyspell-prog-mode)
00330 ### End:
00331 */
00332
00333 #endif // TRISYCL_SYCL_BUFFER_DETAIL_BUFFER_BASE_HPP
```

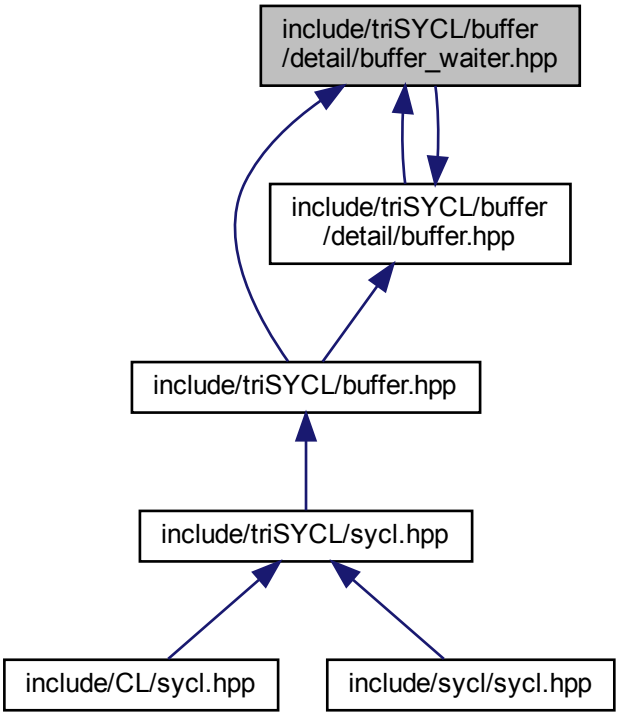
13.25 include/triSYCL/buffer/detail/buffer_waiter.hpp File Reference

```
#include <cstddef>
#include <future>
#include "triSYCL/buffer/detail/buffer.hpp"
#include "triSYCL/buffer_allocator.hpp"
#include "triSYCL/detail/shared_ptr_implementation.hpp"
```

Include dependency graph for buffer_waiter.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::buffer_waiter< T, Dimensions, Allocator >

  A helper class to wait for the final buffer destruction if the conditions for blocking are met. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Functions

- template<typename T , int Dimensions = 1, typename Allocator = buffer_allocator<std::remove_const_t<T>>>
  auto trisycl::detail::waiter (detail::buffer< T, Dimensions > *b)

  Helper function to create a new buffer_waiter.

13.26 buffer_waiter.hpp

Go to the documentation of this file.

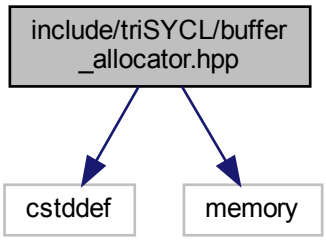
```
00001 #ifndef TRISYCL_SYCL_BUFFER_DETAIL_BUFFER_WAITER_HPP
00002 #define TRISYCL_SYCL_BUFFER_DETAIL_BUFFER_WAITER_HPP
00003
00004 /** \file A helper class to wait for the buffer<> detail
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <cstddef>
00013 #include <future>
00014
00015 #include "triSYCL/buffer/detail/buffer.hpp"
00016 #include "triSYCL/buffer_allocator.hpp"
00017 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00018
00019 namespace trisycl::detail {
00020
00021 /** \addtogroup data Data access and storage in SYCL
00022 @{
00023 */
00024
00025 /** A helper class to wait for the final buffer destruction if the
00026 conditions for blocking are met
00027 */
00028 template <typename T,
00029           int Dimensions = 1,
00030           typename Allocator = buffer_allocator<std::remove_const_t<T>>>
00031 class buffer_waiter :
00032     public detail::shared_ptr_implementation<buffer_waiter<T,
00033                                                            Dimensions,
00034                                                            Allocator>,
00035                                              detail::buffer<T, Dimensions>>,
00036     detail::debug<buffer_waiter<T, Dimensions, Allocator>> {
00037
00038   // The type encapsulating the implementation
00039   using implementation_t = typename buffer_waiter::shared_ptr_implementation;
00040
00041   // Allows the comparison operation to access the implementation
00042   friend implementation_t;
00043
00044 public:
00045
00046   // Make the implementation member directly accessible in this class
00047   using implementation_t::implementation;
00048
00049 /// Create a new buffer_waiter on top of a detail::buffer
00050   buffer_waiter(detail::buffer<T, Dimensions> *b) :  implementation_t { b } {}
00051
00052
00053 /** The buffer_waiter destructor waits for any data to be written
00054 back to the host, if any
00055 */
00056   ~buffer_waiter() {
00057     /* Get a future from the implementation if we have to wait for its
00058 destruction */
00059     auto f = implementation->get_destructor_future();
00060     if (f) {
00061       /* No longer carry for the implementation buffer which is free to
00062 live its life up to its destruction */
00063       implementation.reset();
00064       TRISYCL_DUMP_T("~buffer_waiter() is waiting");
00065       // Then wait for its end in some other thread
00066       f->wait();
00067       TRISYCL_DUMP_T("~buffer_waiter() is done");
00068     }
00069   }
00070 };
```

```
00071
00072
00073 /// Helper function to create a new buffer_waiter
00074 template <typename T,
00075           int Dimensions = 1,
00076           typename Allocator = buffer_allocator<std::remove_const_t<T>»
00077 inline auto waiter(detail::buffer<T, Dimensions> *b) {
00078   return new buffer_waiter<T, Dimensions, Allocator> { b };
00079 }
00080
00081 /// @} End the data Doxygen group
00082
00083 }
00084
00085 /*
00086 # Some Emacs stuff:
00087 ### Local Variables:
00088 ### ispell-local-dictionary:  "american"
00089 ### eval:  (flyspell-prog-mode)
00090 ### End:
00091 */
00092
00093 #endif // TRISYCL_SYCL_BUFFER_DETAIL_BUFFER_WAITER_HPP
```

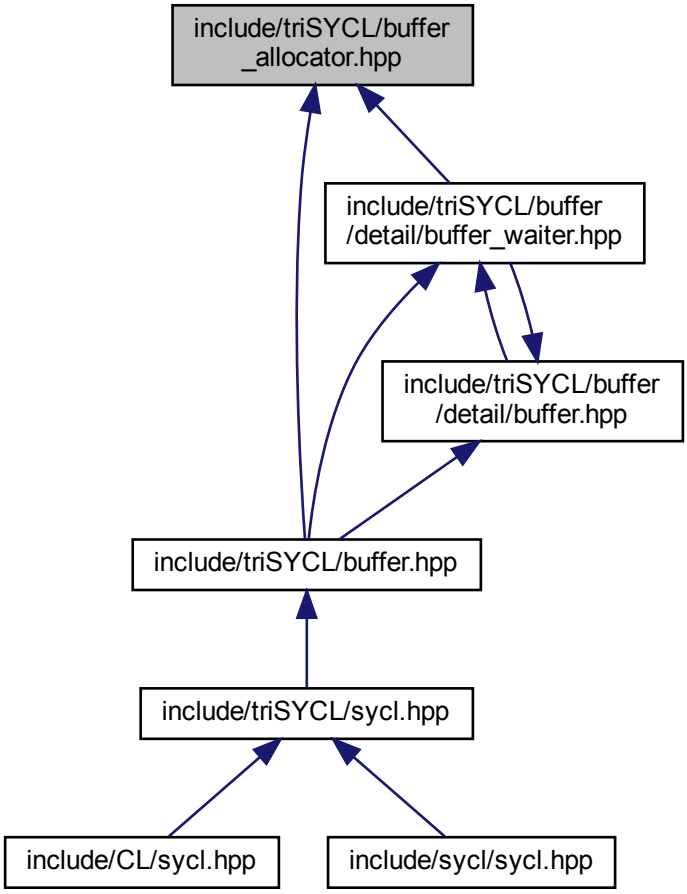
13.27 include/triSYCL/buffer_allocator.hpp File Reference

```
#include <cstddef>
#include <memory>
```

Include dependency graph for buffer_allocator.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

13.28 buffer_allocator.hpp

Go to the documentation of this file.

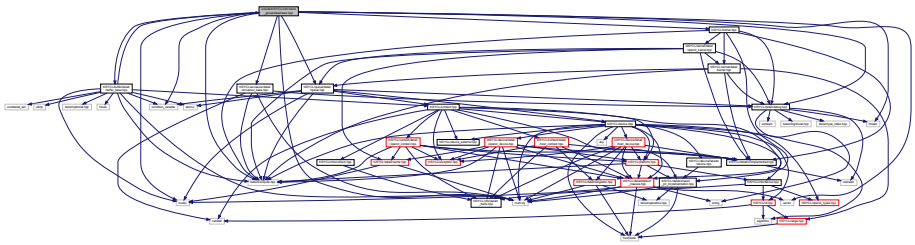
```
00001 #ifndef TRISYCL_SYCL_BUFFER_ALLOCATOR_HPP
00002 #define TRISYCL_SYCL_BUFFER_ALLOCATOR_HPP
00003
00004 /** \file The OpenCL SYCL buffer_allocator
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <cstddef>
```

```
00013 #include <memory>
00014
00015 namespace trisycl {
00016
00017 /** \addtogroup data Data access and storage in SYCL
00018 @{
00019 */
00020
00021 /** The default buffer allocator used by the runtime, when no allocator is
00022 defined by the user
00023
00024 Reuse the C++ default allocator.
00025 */
00026 template <typename T>
00027 using buffer_allocator = std::allocator<T>;
00028
00029 /// @} End the data Doxygen group
00030
00031 }
00032
00033 /*
00034 # Some Emacs stuff:
00035 ### Local Variables:
00036 ### ispell-local-dictionary:  "american"
00037 ### eval:  (flyspell-prog-mode)
00038 ### End:
00039 */
00040
00041 #endif // TRISYCL_SYCL_BUFFER_ALLOCATOR_HPP
```

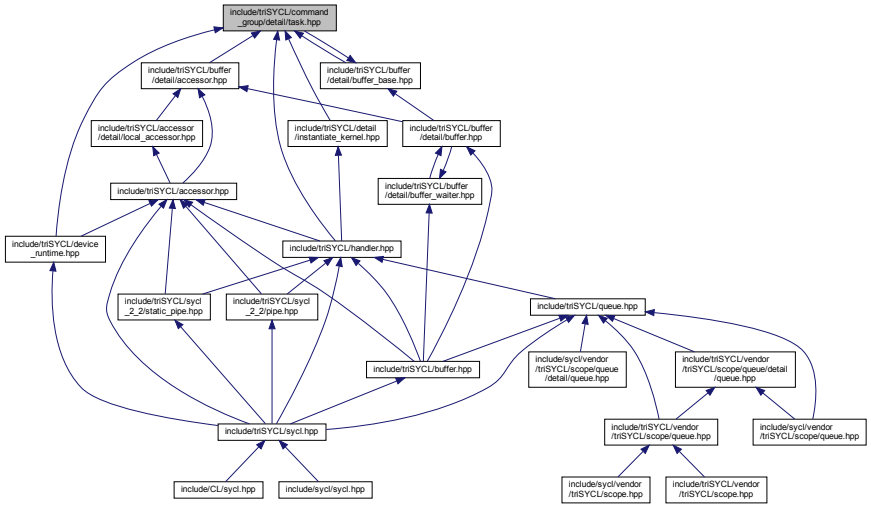
13.29 include/triSYCL/command_group/detail/task.hpp File Reference

```
#include <condition_variable>
#include <memory>
#include <mutex>
#include <thread>
#include <vector>
#include <boost/compute.hpp>
#include "triSYCL/accessor/detail/accessor_base.hpp"
#include "triSYCL/buffer/detail/buffer_base.hpp"
#include "triSYCL/detail/debug.hpp"
#include "triSYCL/kernel.hpp"
#include "triSYCL/queue/detail/queue.hpp"
```

Include dependency graph for task.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::detail::task

  The abstraction to represent SYCL tasks executing inside command_group.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.30 task.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_TASK_HPP
00002 #define TRISYCL_SYCL_TASK_HPP
00003
00004 /** \file The concept of task behind the scene
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <condition_variable>
00013 #include <memory>
00014 #include <mutex>
00015 #include <thread>
00016 #include <vector>
00017
00018 #ifdef TRISYCL_OPENCL
00019 #include <boost/compute.hpp>
00020 #endif
00021
00022 #include "triSYCL/accessor/detail/accessor_base.hpp"
00023 #include "triSYCL/buffer/detail/buffer_base.hpp"
```

```
00024 #include "triSYCL/detail/debug.hpp"
00025 #include "triSYCL/kernel.hpp"
00026 #include "triSYCL/queue/detail/queue.hpp"
00027
00028 namespace trisycl::detail {
00029
00030 /** The abstraction to represent SYCL tasks executing inside command_group
00031
00032 "enable_shared_from_this" allows to access the shared_ptr behind the
00033 scene.
00034 */
00035 struct task :  public std::enable_shared_from_this<task>,
00036               public detail::debug<task> {
00037
00038 /** List of the buffers used by this task
00039
00040 \todo Use a set to check that some buffers are not used many
00041 times at least on writing
00042 */
00043   std::vector<std::shared_ptr<detail::buffer_base>> buffers_in_use;
00044
00045 /// The tasks producing the buffers used by this task
00046   std::vector<std::shared_ptr<detail::task>> producer_tasks;
00047
00048 /// Keep track of any prologue to be executed before the kernel
00049   std::vector<std::function<void(void)>> prologues;
00050
00051 /// Keep track of any epilogue to be executed after the kernel
00052   std::vector<std::function<void(void)>> epilogues;
00053
00054 /// Store if the execution ended, to be notified by task_ready
00055   bool execution_ended = false;
00056
00057 /// To signal when this task is ready
00058   std::condition_variable ready;
00059
00060 /// To protect the access to the condition variable
00061   std::mutex ready_mutex;
00062
00063 /** Keep track of the queue used to submission to notify kernel completion
00064 or to run OpenCL kernels on */
00065   std::shared_ptr<detail::queue> owner_queue;
00066
00067 /// The OpenCL-compatible kernel run by this task, if any
00068   std::shared_ptr<detail::kernel> kernel;
00069
00070 /** The accessors indexed by their creation order
00071
00072 This is used to relate a kernel parameter of a kernel generated
00073 by the device compiler to its accessor.  */
00074   std::vector<std::weak_ptr<detail::accessor_base>> accessors;
00075
00076
00077 /// Create a task from a submitting queue
00078   task(const std::shared_ptr<detail::queue> &q)
00079     :  owner_queue { q } {}
00080
00081
00082 /// Add a new task to the task graph and schedule for execution
00083   void schedule(std::function<void(void)> f) {
00084     /* To keep a copy of the task shared_ptr after the end of the
00085 command group, capture it by copy in the following lambda.
00086 */
00087     auto execution = [f = std::move(f), task = shared_from_this()] {
00088       // Wait for the required tasks to be ready
00089       task->wait_for_producers();
00090       task->prelude();
00091       TRISYCL_DUMP_T("Execute the kernel");
00092       // Execute the kernel
00093       f();
00094       task->postlude();
00095       // Release the buffers that have been written by this task
00096       task->release_buffers();
00097       // Notify the waiting tasks that we are done
00098       task->notify_consumers();
00099       // Notify the queue we are done
00100       task->owner_queue->kernel_end();
00101       TRISYCL_DUMP_T("Task thread exit");
00102     };
00103     /* Notify the queue that there is a kernel submitted to the
00104 queue.  Do not do it in the task contructor so that we can deal
00105 with command group without kernel and if we put it inside the
00106 thread, the queue may have finished before the thread is
00107 scheduled */
00108     owner_queue->kernel_start();
00109     /* \todo it may be implementable with packaged_task that would
00110 deal with exceptions in kernels
```

```
00111 */
00112 #ifndef TRISYCL_NO_ASYNC
00113     /* If in asynchronous execution mode, execute the functor in a new
00114 thread */
00115     std::thread thread(execution);
00116     TRISYCL_DUMP_T("Task thread started");
00117 /** Detach the thread since it will synchronize by its own means
00118
00119 \todo This is an issue if there is an exception in the kernel
00120 */
00121     thread.detach();
00122 #else
00123     // Just a synchronous execution otherwise
00124     execution();
00125 #endif
00126   }
00127
00128
00129 /// Wait for the required producer tasks to be ready
00130   void wait_for_producers() {
00131     TRISYCL_DUMP_T("Task " << this << " waits for the producer tasks");
00132     for (auto &t :  producer_tasks)
00133       t->wait();
00134     // We can let the producers rest in peace
00135     producer_tasks.clear();
00136   }
00137
00138
00139 /// Release the buffers that have  been used by this task
00140   void release_buffers() {
00141     TRISYCL_DUMP_T("Task " << this << " releases the written buffers");
00142     for (auto b:  buffers_in_use)
00143       b->release();
00144     buffers_in_use.clear();
00145   }
00146
00147
00148 /// Notify the waiting tasks that we are done
00149   void notify_consumers() {
00150     TRISYCL_DUMP_T("Notify all the task waiting for this task " << this);
00151    {
00152      std::unique_lock<std::mutex> ul { ready_mutex };
00153      execution_ended = true;
00154    }
00155     /* \todo Verify that the memory model with the notify does not
00156 require some fence or atomic */
00157     ready.notify_all();
00158   }
00159
00160
00161 /** Wait for this task to be ready
00162
00163 This is to be called from another thread
00164 */
00165   void wait() {
00166     TRISYCL_DUMP_T("The task wait for task " << this << " to end");
00167     std::unique_lock<std::mutex> ul { ready_mutex };
00168     ready.wait(ul, [&] { return execution_ended; });
00169   }
00170
00171
00172 /** Register a buffer to this task
00173
00174 This is how the dependency graph is incrementally built.
00175 */
00176   void add_buffer(std::shared_ptr<detail::buffer_base> &buf,
00177                   bool is_write_mode) {
00178     TRISYCL_DUMP_T("Add buffer " << buf << " in task " << this);
00179     /* Keep track of the use of the buffer to notify its release at
00180 the end of the execution */
00181     buffers_in_use.push_back(buf);
00182     // To be sure the buffer does not disappear before the kernel can run
00183     buf->use();
00184
00185     std::shared_ptr<detail::task> latest_producer;
00186     if (is_write_mode) {
00187       /* Set this task as the latest producer of the buffer so that
00188 another kernel may wait on this task */
00189       latest_producer = buf->set_latest_producer(shared_from_this());
00190     }
00191     else
00192       latest_producer = buf->get_latest_producer();
00193
00194     /* If the buffer is to be produced by a task, add the task in the
00195 producer list to wait on it before running the task core
00196
00197 If a buffer is accessed first in write mode and then in read mode,
```

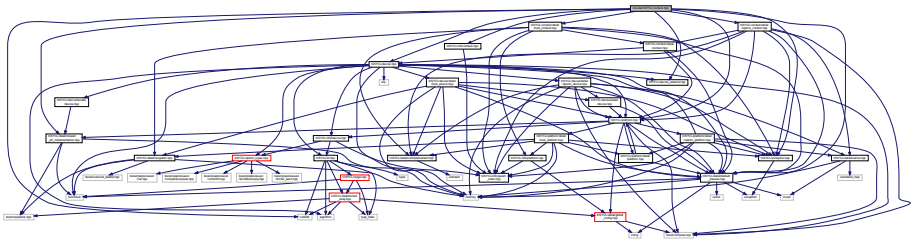
```
00198 the task will add itself as a producer and will wait for itself
00199 when calling \c wait_for_producers, we avoid this by checking that
00200 \c latest_producer is not \c this
00201 */
00202     if (latest_producer && latest_producer != shared_from_this())
00203       producer_tasks.push_back(latest_producer);
00204   }
00205
00206
00207 /// Execute the prologues
00208   void prelude() {
00209     TRISYCL_DUMP_T("task::prelude");
00210
00211     for (const auto &p :  prologues)
00212       p();
00213     /* Free the functors that may own an accessor owning a buffer
00214 preventing the command group to complete */
00215     prologues.clear();
00216   }
00217
00218
00219 /// Execute the epilogues
00220   void postlude() {
00221     for (const auto &p :  epilogues)
00222       p();
00223     /* Free the functors that may own an accessor owning a buffer
00224 preventing the command group to complete */
00225     epilogues.clear();
00226   }
00227
00228
00229 /// Add a function to the prelude to run before kernel execution
00230   void add_prelude(const std::function<void(void)> &f) {
00231     TRISYCL_DUMP_T("task::add_prelude");
00232
00233     prologues.push_back(f);
00234   }
00235
00236
00237 /// Add a function to the postlude to run after kernel execution
00238   void add_postlude(const std::function<void(void)> &f) {
00239     epilogues.push_back(f);
00240   }
00241
00242
00243 /// Get the queue behind the task to run a kernel on
00244   auto get_queue() {
00245     return owner_queue;
00246   }
00247
00248
00249 /// Set the OpenCL-compatible kernel running this task if any
00250   void set_kernel(const std::shared_ptr<trisycl::detail::kernel> &k) {
00251     kernel = k;
00252   }
00253
00254
00255 /** Get the kernel running if any
00256
00257 \todo Specify this error in the spec
00258 */
00259   trisycl::detail::kernel &get_kernel() {
00260     if (!kernel)
00261       throw non_cl_error("Cannot use an OpenCL kernel in this context");
00262     return *kernel;
00263   }
00264
00265
00266 /// Set a kernel argument by address
00267   void set_arg(std::size_t arg_index, std::size_t arg_size, const void *scalar_value) {
00268 #ifdef TRISYCL_OPENCL
00269     // Forward to the OpenCL kernel
00270     get_kernel().get_boost_compute().set_arg(arg_index,
00271                                              arg_size,
00272                                              scalar_value);
00273 #else
00274     throw non_cl_error("Not compiled with OpenCL support");
00275 #endif
00276   }
00277
00278
00279 /// Set a kernel argument by value
00280   template <typename T>
00281   void set_arg(std::size_t arg_index, const T &scalar_value) {
00282 #ifdef TRISYCL_OPENCL
00283     // Forward to the OpenCL kernel
00284     get_kernel().get_boost_compute().set_arg(arg_index, scalar_value);
```

```
00285 #else
00286     throw non_cl_error("Not compiled with OpenCL support");
00287 #endif
00288   }
00289
00290
00291 #ifdef TRISYCL_OPENCL
00292 /// Get the Boost.Compute buffer for an accessor of the task
00293   auto get_compute_buffer(std::size_t order) {
00294     return accessors[order].lock()->get_cl_buffer();
00295   }
00296 #endif
00297
00298
00299 /** Register an accessor and return its registration order
00300
00301 I would prefer to name this method \c register but this is a C++
00302 keyword...
00303 */
00304 std::size_t register_accessor(std::weak_ptr<detail::accessor_base> a) {
00305   accessors.push_back(a);
00306   return accessors.size() - 1;
00307 }
00308
00309 };
00310
00311 }
00312
00313 /*
00314 # Some Emacs stuff:
00315 ### Local Variables:
00316 ### ispell-local-dictionary:  "american"
00317 ### eval:  (flyspell-prog-mode)
00318 ### End:
00319 */
00320
00321 #endif // TRISYCL_SYCL_TASK_HPP
```

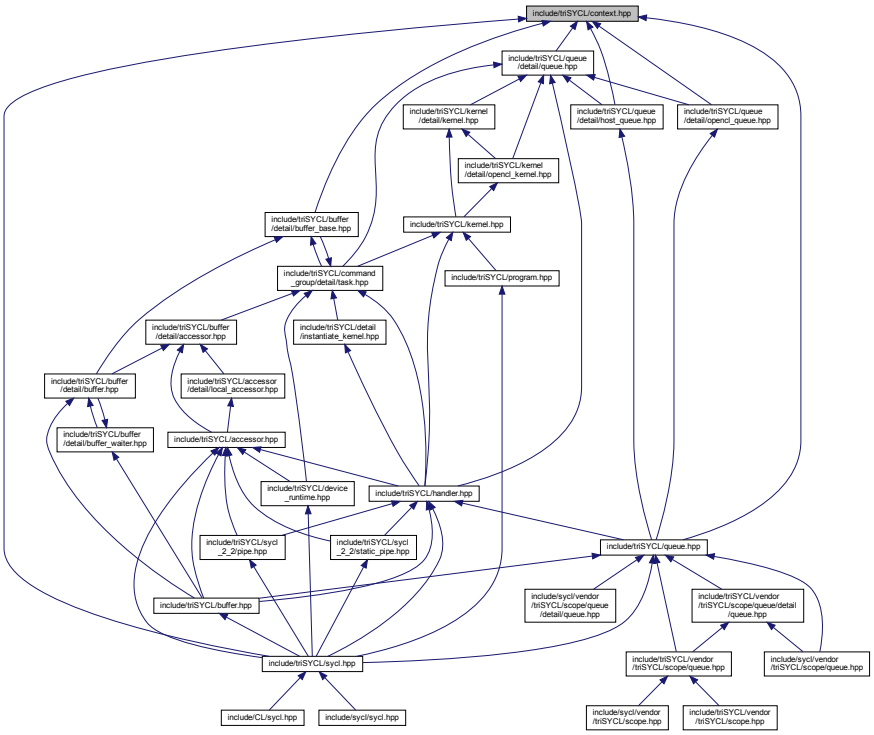
13.31 include/triSYCL/context.hpp File Reference

```
#include <cstddef>
#include "triSYCL/context/detail/host_context.hpp"
#include "triSYCL/context/detail/opencl_context.hpp"
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/detail/shared_ptr_implementation.hpp"
#include "triSYCL/device.hpp"
#include "triSYCL/device_selector.hpp"
#include "triSYCL/exception.hpp"
#include "triSYCL/info/context.hpp"
#include "triSYCL/platform.hpp"
```

Include dependency graph for context.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::context

  SYCL context. More...
- struct std::hash< trisycl::context >

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace std

Functions

- template<> auto trisycl::context::get_info< info::context::reference_count > () const
- template<> auto trisycl::context::get_info< info::context::platform > () const
- template<> auto trisycl::context::get_info< info::context::devices > () const

13.32 context.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_CONTEXT_HPP
00002 #define TRISYCL_SYCL_CONTEXT_HPP
00003
00004 /** \file The OpenCL SYCL context
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <cstddef>
00013
00014 #include "triSYCL/context/detail/host_context.hpp"
00015 #ifdef TRISYCL_OPENCL
00016 #include "triSYCL/context/detail/opencl_context.hpp"
00017 #endif
00018
00019 #include "triSYCL/detail/default_classes.hpp"
00020 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00021 #include "triSYCL/device.hpp"
00022 #include "triSYCL/device_selector.hpp"
00023 #include "triSYCL/exception.hpp"
00024 #include "triSYCL/info/context.hpp"
00025 #include "triSYCL/platform.hpp"
00026
00027 namespace trisycl {
00028
00029 /** \addtogroup execution Platforms, contexts, devices and queues
00030 @{
00031 */
00032
00033 /** SYCL context
00034
00035 The context class encapsulates an OpenCL context, which is implicitly
00036 created and the lifetime of the context instance defines the lifetime
00037 of the underlying OpenCL context instance.
00038
00039 On destruction clReleaseContext is called.
00040
00041 The default context is the SYCL host context containing only the SYCL
00042 host device.
00043
00044 \todo The implementation is quite minimal for now.
00045 */
00046 class context
00047
00048   /* Use the underlying context implementation that can be shared in the
00049 SYCL model */
00050   :  public detail::shared_ptr_implementation<context, detail::context> {
00051
00052   // The type encapsulating the implementation
00053   using implementation_t =
00054     detail::shared_ptr_implementation<context, detail::context>;
00055
00056   context(vector_class<device> devList, async_handler asyncHandler,
00057       const device_selector &deviceSelector) {
00058     if (devList.empty())
00059       devList = {deviceSelector.select_device()};
00060     if (devList[0].is_host())
00061       implementation = detail::host_context::instance();
00062 #ifdef TRISYCL_OPENCL
00063     else
00064       implementation = detail::opencl_context::instance(
00065         boost::compute::context { devList[0].get_boost_compute () }
00066       );
00067 #endif
00068   }
00069
00070 public:
00071
00072     // Make the implementation member directly accessible in this class
00073   using implementation_t::implementation;
00074
00075 /** Constructs a context object for SYCL host using an async_handler for
00076 handling asynchronous errors
00077
00078 Note that the default case asyncHandler = nullptr is handled by the
00079 default constructor.
00080 */
00081   explicit context(async_handler asyncHandler)
00082     :  context { device{}, asyncHandler } {}
```

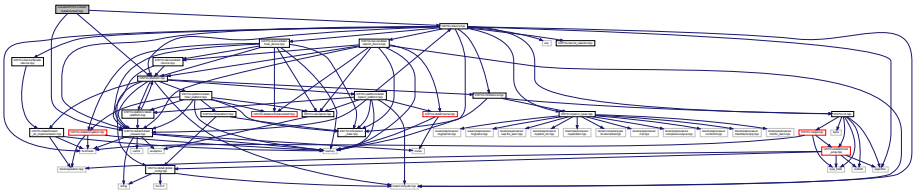
```
00083
00084 #ifdef TRISYCL_OPENCL
00085 /** Make a SYCL context from an OpenCL context
00086
00087 The constructor executes a retain on the \c cl_context.
00088
00089 Return synchronous errors via the SYCL exception class and
00090 asynchronous errors are handled via the \c async_handler, if
00091 provided.
00092 */
00093   context(cl_context clContext, async_handler asyncHandler = nullptr)
00094     : context { boost::compute::context { clContext }, asyncHandler } {}
00095
00096
00097 /// Build a SYCL context from a Boost.Compute context
00098   context(const boost::compute::context &c,
00099           async_handler asyncHandler = nullptr)
00100     : implementation_t { detail::opencl_context::instance(c) } {}
00101 #endif
00102
00103 /** Constructs a context object using a device_selector object
00104
00105 The context is constructed with a single device retrieved from the
00106 device_selector object provided.
00107
00108 Return synchronous errors via the SYCL exception class and
00109 asynchronous errors are handled via the async_handler, if provided.
00110 */
00111   context(const device_selector &deviceSelector, async_handler asyncHandler = nullptr)
00112     : context (vector_class<device>{}, asyncHandler, deviceSelector) {}
00113
00114 /** Constructs a context object using a device object
00115
00116 Return synchronous errors via the SYCL exception class and
00117 asynchronous errors are handled via the async_handler, if provided.
00118 */
00119   context(const device &dev, async_handler asyncHandler = nullptr)
00120     : context { vector_class<device>{dev}, asyncHandler } {}
00121
00122 /** Constructs a context object using a platform object
00123
00124 Return synchronous errors via the SYCL exception class and
00125 asynchronous errors are handled via the async_handler, if provided.
00126 */
00127   context(const platform &plt, async_handler asyncHandler = nullptr)
00128     : context { plt.get_devices(), asyncHandler } {}
00129
00130   /* Constructs a context object using a vector_class of device objects
00131
00132 Return synchronous errors via the SYCL exception class and
00133 asynchronous errors are handled via the async_handler, if provided.
00134
00135 \todo Update the specification to replace vector by collection
00136 concept.
00137 */
00138   context(const vector_class<device> &deviceList,
00139           async_handler asyncHandler = nullptr);
00140
00141 /** Default constructor that chooses the context according the
00142 heuristics of the default selector
00143
00144 Return synchronous errors via the SYCL exception class.
00145
00146 Get the default constructors back.
00147 */
00148   context() : context { async_handler {} } {}
00149
00150 #ifdef TRISYCL_OPENCL
00151 /** Return the underlying \c cl_context object, after retaining
00152 the \c cl_context.
00153
00154 Retains a reference to the returned \c cl_context object.
00155
00156 Caller should release it when finished.
00157 */
00158   cl_context get()const {
00159     return implementation->get();
00160   }
00161
00162 /** Return the underlying \c boost::compute::context
00163 of the \c trisycl::context
00164 */
00165   boost::compute::context &get_boost_compute()const {
00166     return implementation->get_boost_compute();
00167   }
00168
00169 /** Return the internal queue that is associated to the context and
```

```
00170 used by triSYCL to move data between some different contests for
00171 example
00172 */
00173   boost::compute::command_queue &get_boost_queue()const {
00174     return implementation->get_boost_queue();
00175   }
00176 #endif
00177
00178
00179 /// Specifies whether the context is in SYCL Host Execution Mode.
00180   bool is_host()const {
00181     return implementation->is_host();
00182   }
00183
00184
00185 /** Returns the SYCL platform that the context is initialized for
00186
00187 \todo To be implemented
00188 */
00189   platform get_platform()const {
00190     return implementation->get_platform();
00191   };
00192
00193
00194 /** Returns the set of devices that are part of this context
00195
00196 \todo To be implemented
00197 */
00198   vector_class<device> get_devices()const {
00199     return implementation->get_devices();
00200   }
00201
00202
00203 /** Queries OpenCL information for the under-lying cl context
00204
00205 \todo To be implemented
00206 */
00207   template <info::context param>
00208   inline auto get_info() const;
00209
00210 };
00211
00212 template<>
00213 inline auto context::get_info<info::context::reference_count>()const {
00214   return implementation->get_reference_count();
00215 }
00216
00217 template<>
00218 inline auto context::get_info<info::context::platform>()const {
00219   return get_platform();
00220 }
00221
00222 template<>
00223 inline auto context::get_info<info::context::devices>()const {
00224   return get_devices();
00225 }
00226
00227
00228 /// @} to end the execution Doxygen group
00229
00230 }
00231
00232 namespace std {
00233
00234   template <> struct hash<trisycl::context> {
00235     auto operator()(const trisycl::context &c)const {
00236     return c.hash();
00237     }
00238   };
00239
00240 }
00241
00242 /*
00243 # Some Emacs stuff:
00244 ### Local Variables:
00245 ### ispell-local-dictionary:  "american"
00246 ### eval:  (flyspell-prog-mode)
00247 ### End:
00248 */
00249
00250 #endif // TRISYCL_SYCL_CONTEXT_HPP
```

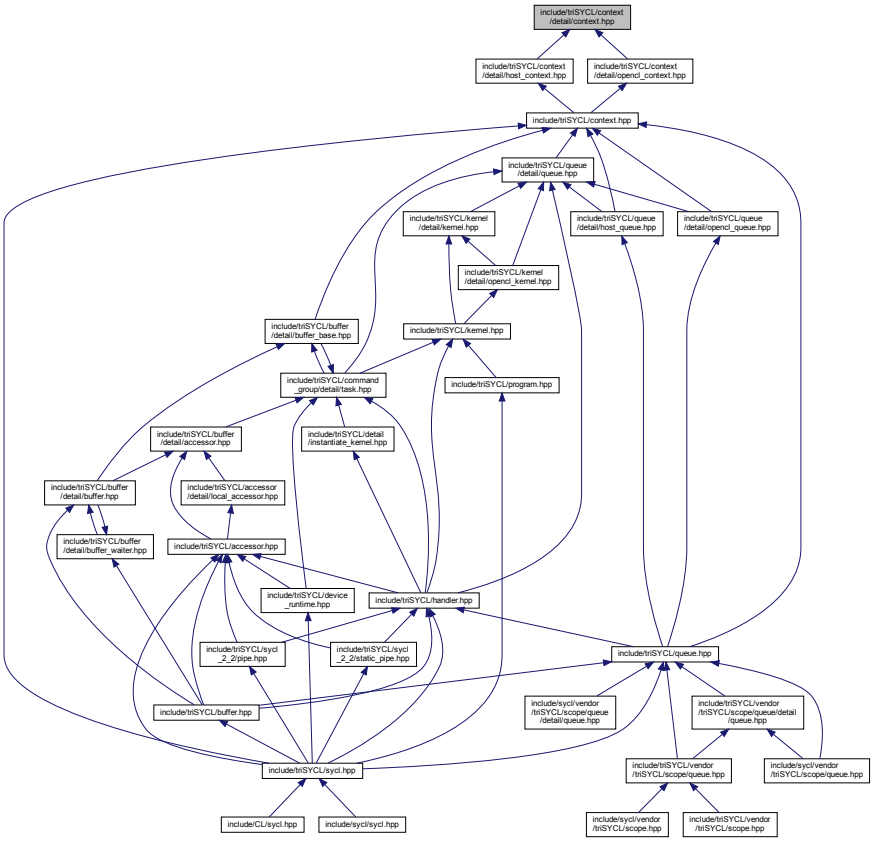
13.33 include/triSYCL/context/detail/context.hpp File Reference

```
#include "triSYCL/device.hpp"
#include "triSYCL/platform.hpp"
#include "triSYCL/detail/default_classes.hpp"
```

Include dependency graph for context.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::context

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

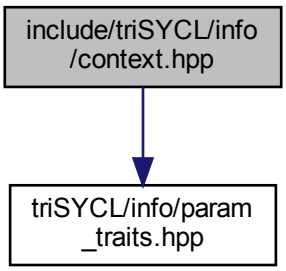
13.34 context.hpp

Go to the documentation of this file.

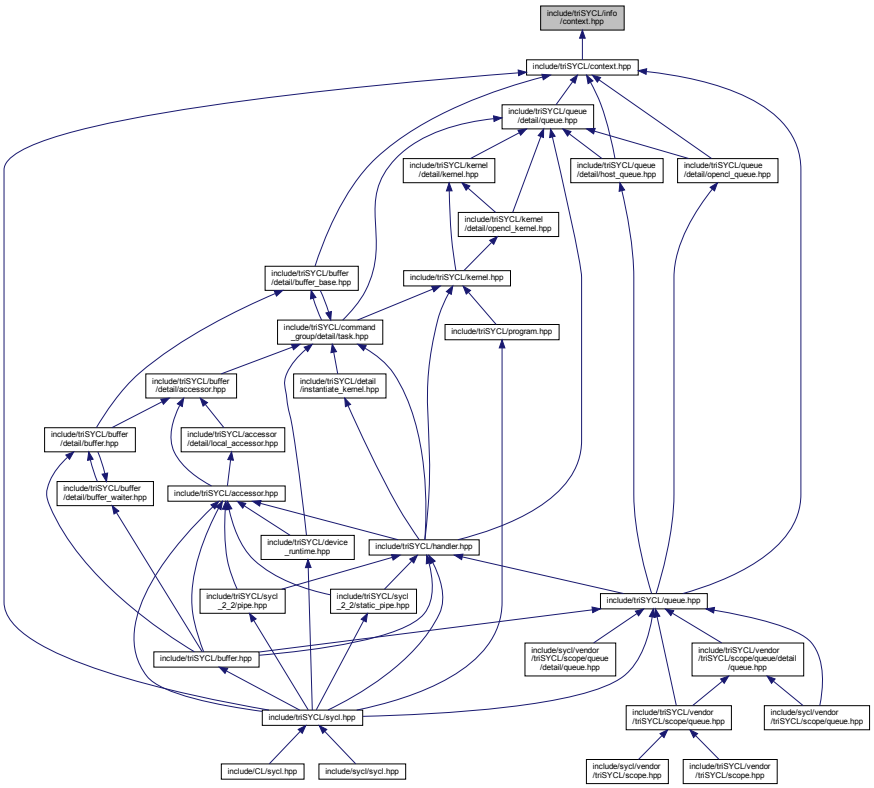
```
00001 #ifndef TRISYCL_SYCL_CONTEXT_DETAIL_CONTEXT_HPP
00002 #define TRISYCL_SYCL_CONTEXT_DETAIL_CONTEXT_HPP
00003
00004 /** \file The OpenCL SYCL abstract context
00005
00006 a-doumoulakis at gmail dot com (Anastasios Doumoulakis)
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include "triSYCL/device.hpp"
00013 #include "triSYCL/platform.hpp"
00014 #include "triSYCL/detail/default_classes.hpp"
00015
00016 namespace trisycl::detail {
00017
00018 /** \addtogroup execution Platforms, contexts, devices and queues
00019 @{
00020 */
00021
00022 class context {
00023 public:
00024
00025 #ifdef TRISYCL_OPENCL
00026 /// Return the underlying \c cl_context of the \c trisycl::context
00027   virtual cl_context get() const = 0;
00028
00029 /** Return the underlying \c boost::compute::context
00030 of the \c trisycl::context
00031 */
00032   virtual boost::compute::context &get_boost_compute() = 0;
00033
00034 /** Return the underlying \c boost::compute::command_queue associated
00035 with the context
00036 */
00037   virtual boost::compute::command_queue &get_boost_queue() = 0;
00038 #endif
00039
00040 /// Returns true is the context is a SYCL host context
00041   virtual bool is_host() const = 0;
00042
00043 /// Returns the SYCL platform that the context is initialized for
00044   virtual trisycl::platform get_platform() const = 0;
00045
00046 /** \todo virtual cannot be templated
00047 template <info::context Param>
00048 typename info::param_traits<info::context, Param>::type
00049 get_info() const = 0;
00050 */
00051
00052 /// Returns the set of devices that are part of this context.
00053   virtual vector_class<trisycl::device> get_devices() const = 0;
00054
00055 /// Returns the reference count of underlying CL context, or 0 on host.
00056   virtual trisycl::cl_uint get_reference_count() const = 0;
00057
00058 /// Virtual to call the real destructor
00059   virtual ~context() {}
00060
00061 };
00062
00063 /// @} to end the execution Doxygen group
00064
00065 }
00066
00067 /*
00068 # Some Emacs stuff:
00069 ### Local Variables:
00070 ### ispell-local-dictionary:  "american"
00071 ### eval:  (flyspell-prog-mode)
00072 ### End:
00073 */
00074
00075 #endif // TRISYCL_SYCL_DEVICE_DETAIL_DEVICE_HPP
```

13.35 include/triSYCL/info/context.hpp File Reference

`#include "triSYCL/info/param_traits.hpp"`
Include dependency graph for context.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::info

Enumerations

- enum class trisycl::info::context : int { trisycl::info::context::reference_count , trisycl::info::context::platform , trisycl::info::context::devices }

  Context information descriptors.

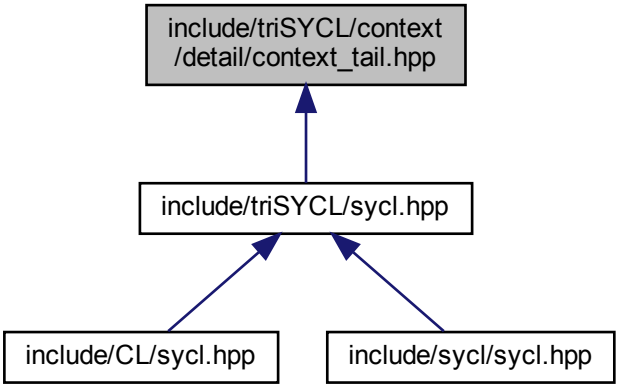
13.36 context.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_INFO_CONTEXT_HPP
00002 #define TRISYCL_SYCL_INFO_CONTEXT_HPP
00003
00004 /** \file The OpenCL SYCL context information parameters
00005
00006 Anastasi at Xilinx dot com
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include "triSYCL/info/param_traits.hpp"
00013
00014 namespace trisycl {
00015
00016 /** \addtogroup execution Platforms, contexts, devices and queues
00017 @{
00018 */
00019 namespace info {
00020
00021 /** Context information descriptors
00022
00023 \todo Should be unsigned int to be consistent with others?
00024 */
00025 enum class context :  int {
00026   reference_count,
00027   platform,
00028   devices
00029 };
00030
00031
00032 /** Query the return type for get_info() on context stuff
00033
00034 \todo To be implemented, return always void.
00035 */
00036 TRISYCL_INFO_PARAM_TRAITS_ANY_T(info::context, void)
00037 TRISYCL_INFO_PARAM_TRAITS(info::context::reference_count, trisycl::cl_uint)
00038 TRISYCL_INFO_PARAM_TRAITS(info::context::platform, trisycl::platform)
00039 TRISYCL_INFO_PARAM_TRAITS(info::context::devices, vector_class<trisycl::device>)
00040 }
00041 }
00042
00043 /*
00044 # Some Emacs stuff:
00045 ### Local Variables:
00046 ### ispell-local-dictionary:  "american"
00047 ### eval:  (flyspell-prog-mode)
00048 ### End:
00049 */
00050
00051
00052 #endif //TRISYCL_SYCL_INFO_CONTEXT_HPP
```

13.37 include/triSYCL/context/detail/context_tail.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

13.38 context_tail.hpp

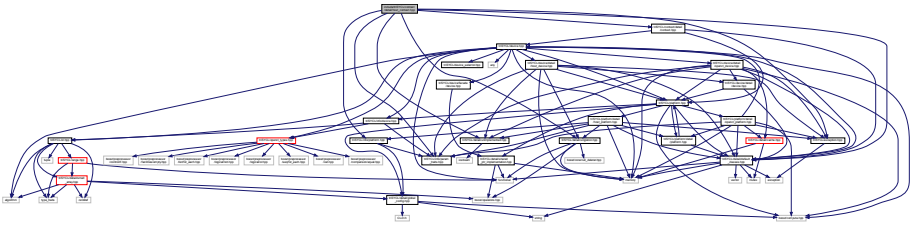
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_CONTEXT_DETAIL_CONTEXT_TAIL_HPP
00002 #define TRISYCL_SYCL_CONTEXT_DETAIL_CONTEXT_TAIL_HPP
00003
00004 /** \file The ending part of of the OpenCL SYCL context
00005
00006 This is here to break a dependence between context and device_selector
00007
00008 Dave Airlie
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 namespace trisycl {
00015
00016 /* has to be inline as it references default_selector */
00017 inline context::context(const vector_class<device> &deviceList,
00018             async_handler asyncHandler)
00019   :  context(deviceList, asyncHandler, default_selector {}) {}
00020
00021 }
00022 /*
00023 # Some Emacs stuff:
00024 ### Local Variables:
00025 ### ispell-local-dictionary:  "american"
00026 ### eval:  (flyspell-prog-mode)
00027 ### End:
00028 */
00029
00030 #endif // TRISYCL_SYCL_CONTEXT_DETAIL_CONTEXT_TAIL_HPP
```

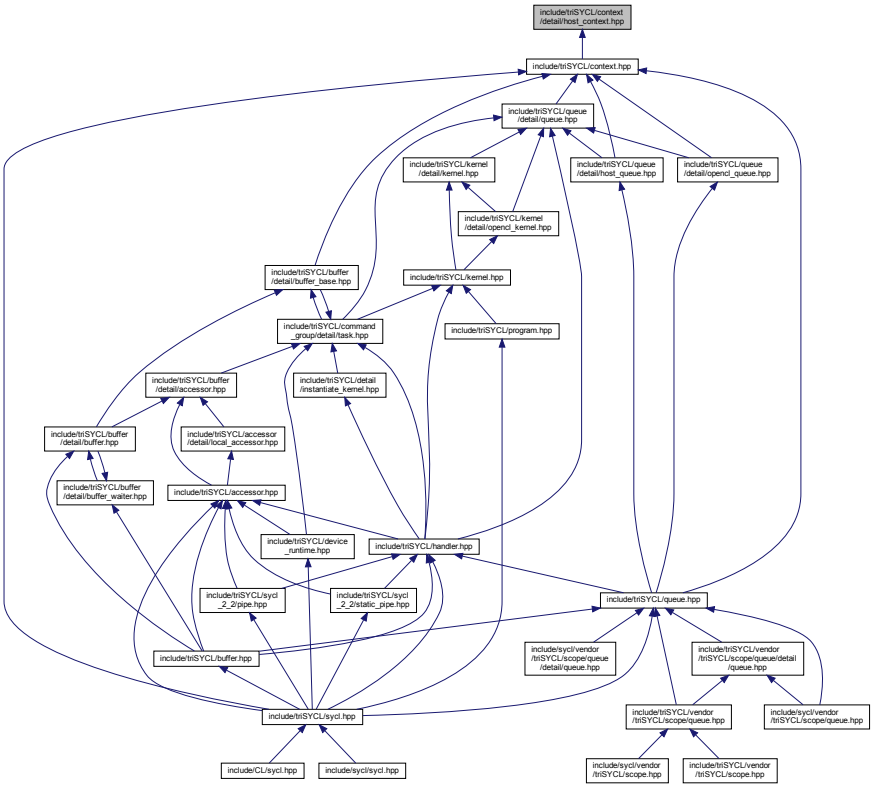
13.39 include/triSYCL/context/detail/host_context.hpp File Reference

```
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/detail/unimplemented.hpp"
#include "triSYCL/detail/singleton.hpp"
#include "triSYCL/exception.hpp"
#include "triSYCL/info/param_traits.hpp"
#include "triSYCL/info/platform.hpp"
#include "triSYCL/context/detail/context.hpp"
```

Include dependency graph for host_context.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::host_context

  SYCL host context. *More...*

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.40 host_context.hpp

Go to the documentation of this file.

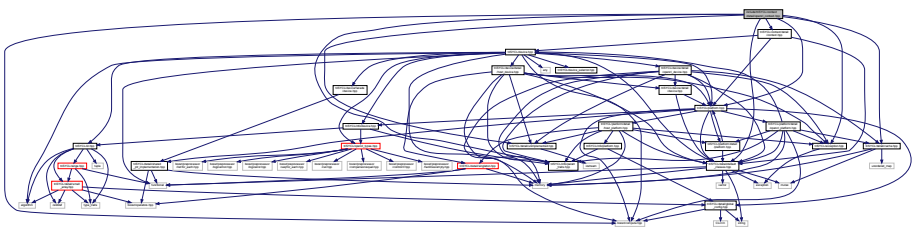
```
00001 #ifndef TRISYCL_SYCL_CONTEXT_DETAIL_HOST_CONTEXT_HPP
00002 #define TRISYCL_SYCL_CONTEXT_DETAIL_HOST_CONTEXT_HPP
00003
00004 /** \file The OpenCL SYCL host context implementation
00005
00006 a-doumoulakis at gmail dot com (Anastasios Doumoulakis)
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include "triSYCL/detail/default_classes.hpp"
00013 #include "triSYCL/detail/unimplemented.hpp"
00014 #include "triSYCL/detail/singleton.hpp"
00015 #include "triSYCL/exception.hpp"
00016 #include "triSYCL/info/param_traits.hpp"
00017 #include "triSYCL/info/platform.hpp"
00018 #include "triSYCL/context/detail/context.hpp"
00019
00020 namespace trisycl::detail {
00021
00022
00023 /** \addtogroup execution Platforms, contexts, devices and queues
00024 @{
00025 */
00026
00027 /** SYCL host context
00028
00029 \todo The implementation is quite minimal for now.  :-)
00030 */
00031 class host_context :  public detail::context,
00032                       public detail::singleton<host_context> {
00033
00034 public:
00035
00036 #ifdef TRISYCL_OPENCL
00037 /** Return the underlying \c cl_context of the \c trisycl::context
00038
00039 This throws an error since there is no OpenCL context associated
00040 to the host device.
00041 */
00042   cl_context get()const override {
00043     throw non_cl_error("The host context has no OpenCL context");
00044   }
00045
00046
00047 /** Return the SYCL platform that the context is initialized for
00048
00049 This throws an error since there is no \c boost::compute context
00050 associated to the host device.
00051 */
00052   boost::compute::context &get_boost_compute()override {
00053     throw non_cl_error("The host context has no OpenCL context");
00054   }
00055
00056
00057 /** Return the internal OpenCL queue that is associated to the host
00058 context
00059
00060 This throws an error since there is no \c
00061 boost::compute::command_queue associated to the host context.
00062 */
00063   boost::compute::command_queue &get_boost_queue()override {
00064     throw non_cl_error("The host context cannot have an OpenCL queue");
00065   }
00066 #endif
00067
00068
```

```
00069 /// Return true since the context is a SYCL host context
00070   bool is_host()const override {
00071     return true;
00072   }
00073
00074
00075 /** Return the platform of the context
00076
00077 Return synchronous errors via the SYCL exception class.
00078 */
00079   trisycl::platform get_platform()const override {
00080     // Return the host platform
00081     return {};
00082   }
00083
00084 #if 0
00085 /** Query the context for OpenCL \c info::context info
00086
00087 Return synchronous errors via the SYCL exception class.
00088
00089 \todo To be implemented
00090 */
00091   template <info::context Param>
00092   typename info::param_traits<info::context, Param>::type
00093   get_info()const override {
00094     TRISYCL_UNIMPL;
00095     return {};
00096   }
00097 #endif
00098
00099
00100 /** Returns the set of devices that are part of this context.
00101 It should only return the host device itself.
00102
00103 \todo To be implemented
00104 */
00105   vector_class<trisycl::device>
00106   get_devices()const override {
00107     // Return just the host device
00108     return { {} };
00109   }
00110
00111 /// Return 0 since the context is a SYCL host context
00112   trisycl::cl_uint get_reference_count()const override {
00113     return 0;
00114   }
00115 };
00116
00117 /// @} to end the execution Doxygen group
00118
00119 }
00120 #endif // TRISYCL_SYCL_CONTEXT_DETAIL_HOST_CONTEXT_HPP
```

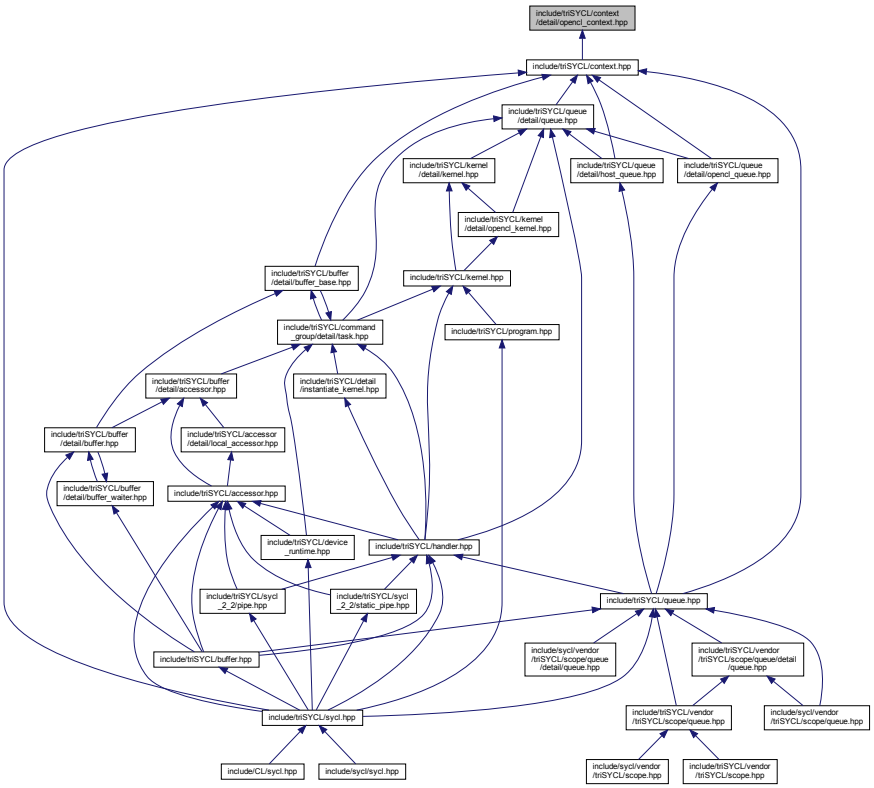
13.41 include/triSYCL/context/detail/opencl_context.hpp File Reference

```
#include <boost/compute.hpp>
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/detail/unimplemented.hpp"
#include "triSYCL/detail/cache.hpp"
#include "triSYCL/context/detail/context.hpp"
#include "triSYCL/platform.hpp"
#include "triSYCL/info/param_traits.hpp"
#include "triSYCL/exception.hpp"
```

Include dependency graph for opencl_context.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::opencl_context

  SYCL OpenCL context.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Variables

- TRISYCL_WEAK_ATTRIB_PREFIX detail::cache< cl_context, detail::opencl_context > opencl_context↩ ::cache trisycl::detail::TRISYCL_WEAK_ATTRIB_SUFFIX

13.42 opencl_context.hpp

Go to the documentation of this file.

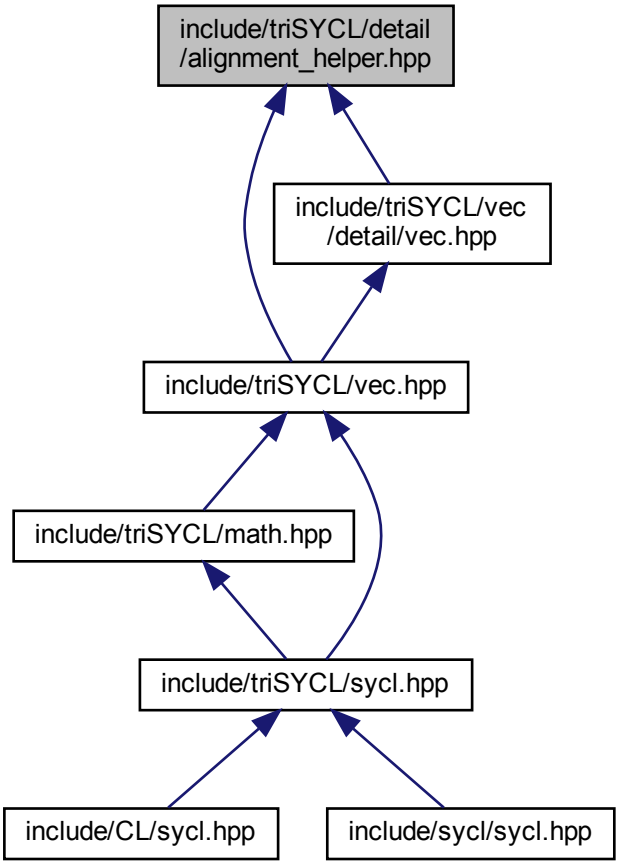
```
00001 #ifndef TRISYCL_SYCL_CONTEXT_DETAIL_OPENCL_CONTEXT_HPP
00002 #define TRISYCL_SYCL_CONTEXT_DETAIL_OPENCL_CONTEXT_HPP
00003
00004 /** \file The SYCL OpenCL context implementation
00005
00006 a-doumoulakis at gmail dot com (Anastasios Doumoulakis)
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012
00013 #include <boost/compute.hpp>
00014
00015 #include "triSYCL/detail/default_classes.hpp"
00016 #include "triSYCL/detail/unimplemented.hpp"
00017 #include "triSYCL/detail/cache.hpp"
00018
00019 #include "triSYCL/context/detail/context.hpp"
00020 #include "triSYCL/platform.hpp"
00021 #include "triSYCL/info/param_traits.hpp"
00022 #include "triSYCL/exception.hpp"
00023
00024
00025 namespace trisycl::detail {
00026
00027 /// SYCL OpenCL context
00028 class opencl_context :  public detail::context {
00029
00030 /// User the Boost Compute abstraction of the OpenCL context
00031   boost::compute::context c;
00032
00033 /** A boost \c command_queue associated to an OpenCL context for
00034 when we need to transfer data but no queue is given
00035 (eg.  When an buffer accessor is created)
00036 */
00037   boost::compute::command_queue q;
00038
00039 /** A cache to always return the same alive context for a given OpenCL
00040 context
00041
00042 C++11 guaranties the static construction is thread-safe
00043 */
00044   static detail::cache<cl_context, detail::opencl_context> cache;
00045
00046 public:
00047
00048 /// Return the underlying \c cl_context of the \c trisycl::context
00049   cl_context get()const override {
00050     return c.get();
00051   }
00052
00053
00054 /** Return the underlying \c boost::compute::context
00055 of the \c trisycl::context
00056 */
00057   boost::compute::context &get_boost_compute()override {
00058     return c;
00059   }
00060
00061
00062 /// Return the queue that is associated to the context
00063   boost::compute::command_queue &get_boost_queue()override {
00064     return q;
00065   }
00066
00067
00068 /// Return false because the context is not a SYCL host context
00069   bool is_host()const override {
00070     return false;
00071   }
00072
00073 #if 0
00074 /** Query the context for OpenCL \c info::context info
00075
00076 Return synchronous errors via the SYCL exception class.
00077
00078 \todo To be implemented
00079 */
00080   template <info::context Param>
00081   typename info::param_traits<info::context, Param>::type
00082   get_info()const override{
```

```
00083     TRISYCL_UNIMPL;
00084     return {};
00085   }
00086 #endif
00087
00088 /** Return the platform of the context
00089
00090 Return synchronous errors via the SYCL exception class.
00091
00092 \todo To be implemented
00093 */
00094   trisycl::platform get_platform()const override {
00095     return trisycl::platform(c.get_devices()[0].platform());
00096   }
00097
00098
00099 /** Returns the set of devices that are part of this context.
00100
00101 \todo To be implemented
00102 */
00103   vector_class<trisycl::device> get_devices()const override {
00104     vector_class<trisycl::device> devices;
00105
00106     for (const auto &d :  c.get_devices())
00107       devices.push_back(d);
00108     return devices;
00109   }
00110
00111   trisycl::cl_uint get_reference_count()const override {
00112     return c.get_info<CL_CONTEXT_REFERENCE_COUNT>();
00113   }
00114
00115 /// Get a singleton instance of the \c opencl_context
00116   static std::shared_ptr<opencl_context>
00117   instance(const boost::compute::context &c) {
00118     return cache.get_or_register(c.get(),
00119                                  [&] { return new opencl_context { c }; });
00120   }
00121
00122
00123 private:
00124
00125 /// Only the instance factory can build it
00126   opencl_context(const boost::compute::context &c) :
00127     c { c },
00128     q { boost::compute::command_queue { c, c.get_device() } } {}
00129
00130
00131 public:
00132 /// Unregister from the cache on destruction
00133   ~opencl_context()override {
00134     cache.remove(c.get());
00135   }
00136
00137 };
00138
00139
00140 /* Allocate the cache here but since this is a pure-header library,
00141 use a weak symbol so that only one remains when SYCL headers are
00142 used in different compilation units of a program
00143 */
00144 TRISYCL_WEAK_ATTRIB_PREFIX
00145 detail::cache<cl_context, detail::opencl_context> opencl_context::cache
00146 TRISYCL_WEAK_ATTRIB_SUFFIX;
00147
00148 }
00149
00150 /*
00151 # Some Emacs stuff:
00152 ### Local Variables:
00153 ### ispell-local-dictionary:  "american"
00154 ### eval:  (flyspell-prog-mode)
00155 ### End:
00156 */
00157
00158 #endif //TRISYCL_SYCL_CONTEXT_DETAIL_OPENCL_CONTEXT_HPP
```

13.43 include/triSYCL/detail/alignment_helper.hpp File Reference

A simple trait to help generate the appropriate alignment data for a SYCL type when specified appropriately.

This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::alignment< T >
- class trisycl::detail::alignment<::trisycl::vec< DataType, NumElements > >

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Variables

- template<typename T >
  constexpr auto trisycl::detail::alignment_v = alignment<T>::value

13.43.1 Detailed Description

A simple trait to help generate the appropriate alignment data for a SYCL type when specified appropriately.

Currently only used and specialized for the vec classes which have SYCL spec required alignments.

andrew point gozillon at yahoo point com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file alignment_helper.hpp.

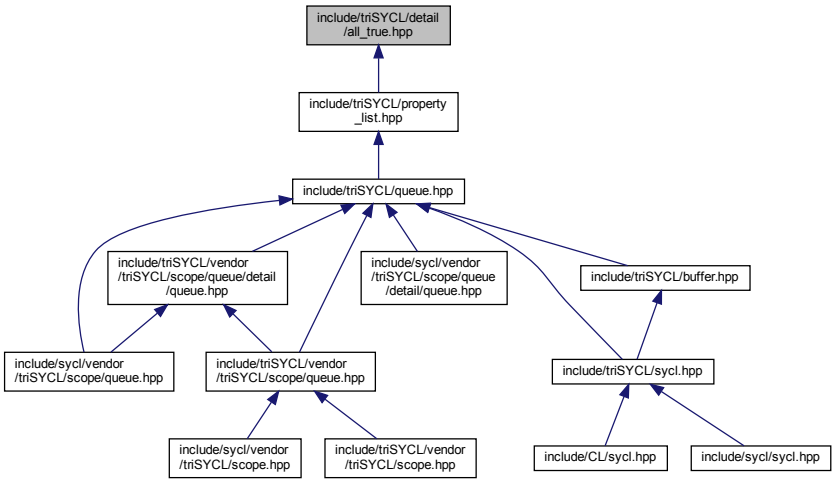
13.44 alignment_helper.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_ALIGNMENT_HELPER_HPP
00002 #define TRISYCL_SYCL_DETAIL_ALIGNMENT_HELPER_HPP
00003
00004 /** \file
00005
00006 A simple trait to help generate the appropriate alignment data for a
00007 SYCL type when specified appropriately.  Currently only used and specialized
00008 for the vec classes which have SYCL spec required alignments.
00009
00010 andrew point gozillon at yahoo point com
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 namespace trisycl {
00017 /// forward declaration to use in the detail class
00018 template<typename, int>
00019 class vec;
00020 }
00021
00022 namespace trisycl::detail {
00023   template <typename T>
00024   class alignment {
00025     public:  static constexpr int value = 0;
00026   };
00027
00028   template <typename DataType, int NumElements>
00029   class alignment<::trisycl::vec<DataType, NumElements>> {
00030     // Vec3's are SYCL spec defined to be aligned as a Vec4.
00031     static constexpr int get_align() {
00032       if  constexpr (NumElements == 3)
00033         return 4 * sizeof(DataType);
00034       else
00035         return NumElements * sizeof(DataType);
00036     }
00037
00038     public:  static constexpr int value = get_align();
00039   };
00040
00041   template <typename T>
00042   constexpr auto alignment_v = alignment<T>::value;
00043 }
00044
00045 #endif
```

13.45 include/triSYCL/detail/all_true.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Typedefs

- template<bool... v>
  using trisycl::detail::all_true = std::is_same< bool_pack< true, v... >, bool_pack< v..., true > >

13.46 all_true.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_ALL_TRUE_HPP
00002 #define TRISYCL_SYCL_DETAIL_ALL_TRUE_HPP
00003
00004 /** \file Add all_true wrapper.
00005 Dave Airlie
00006
00007 This file is distributed under the University of Illinois Open Source
00008 License.  See LICENSE.TXT for details.
00009 */
00010
00011 namespace trisycl::detail {
00012
00013 template <bool...> struct bool_pack;
00014
00015 template <bool...  v>
00016 using all_true = std::is_same<bool_pack<true, v...>, bool_pack<v..., true»;
00017
00018 }
00019
```

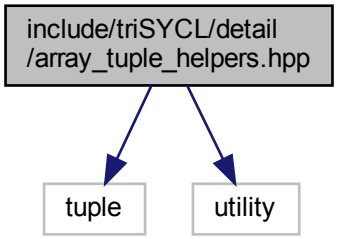
```
00020 /*
00021 # Some Emacs stuff:
00022 ### Local Variables:
00023 ### ispell-local-dictionary:  "american"
00024 ### eval:  (flyspell-prog-mode)
00025 ### End:
00026 */
00027
00028 #endif // TRISYCL_SYCL_DETAIL_ALL_TRUE_HPP
```

13.47 include/triSYCL/detail/array_tuple_helpers.hpp File Reference

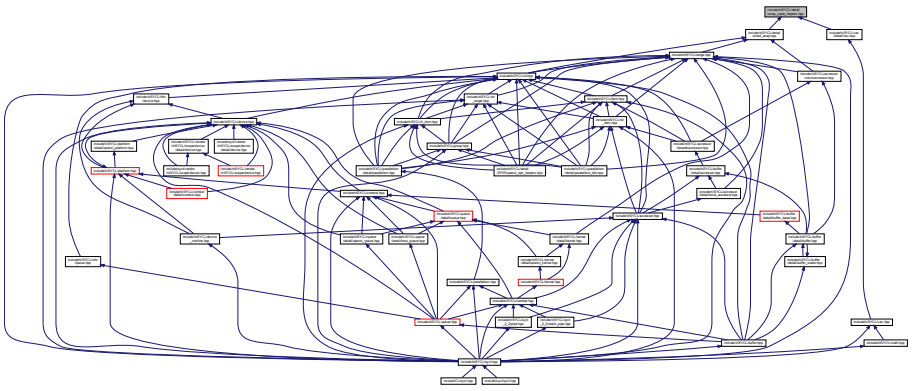
Some helpers to do array-tuple conversions.

```
#include <tuple>
#include <utility>
```

Include dependency graph for array_tuple_helpers.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::detail::expand_to_vector< V, Tuple, expansion >

  Allows optional expansion of a 1-element tuple to a V::dimension tuple to replicate scalar values in vector initialization. More...
- struct trisycl::detail::expand_to_vector< V, Tuple, true >

  Specialization in the case we ask for expansion. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Functions

- template<typename V , typename Tuple , std::size_t... Is>
  std::array< typename V::element_type, V::dimension > trisycl::detail::tuple_to_array_iterate (Tuple t, std::index_sequence< Is... >)

  Helper to construct an array from initializer elements provided as a tuple.
- template<typename V , typename Tuple >
  auto trisycl::detail::tuple_to_array (Tuple t)

  Construct an array from initializer elements provided as a tuple.
- template<typename V , typename Tuple >
  auto trisycl::detail::expand (Tuple t)

  Create the array data of V from a tuple of initializer.

13.47.1 Detailed Description

Some helpers to do array-tuple conversions.

Used for example to implement trisycl::vec<> class.

Ronan at Keryell point FR

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file array_tuple_helpers.hpp.

13.48 array_tuple_helpers.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_ARRAY_TUPLE_HELPERS_HPP
00002 #define TRISYCL_SYCL_DETAIL_ARRAY_TUPLE_HELPERS_HPP
00003
00004 /** \file
00005
00006 Some helpers to do array-tuple conversions
00007
00008 Used for example to implement trisycl::vec<> class.
00009
00010 Ronan at Keryell point FR
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include <tuple>
00017 #include <utility>
00018
00019 namespace trisycl::detail {
00020
00021 /** \addtogroup array_tuple_helpers Helpers to do array and tuple conversion
00022
00023 @{
00024 */
```

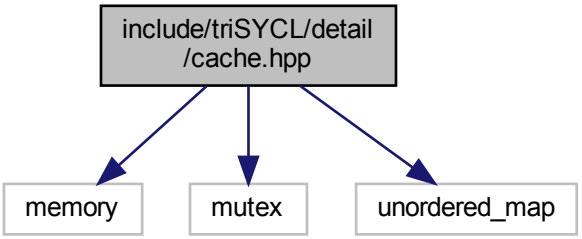
```
00025
00026 /** Helper to construct an array from initializer elements provided as a
00027 tuple
00028
00029 The trick is to get the std::index_sequence<> that represent 0,
00030 1,..., dimension-1 as a variadic template pack Is that we can
00031 iterate on, in this function.
00032 */
00033 template <typename V, typename Tuple, std::size_t...  Is>
00034 std::array<typename V::element_type, V::dimension>
00035 tuple_to_array_iterate(Tuple t, std::index_sequence<Is...>) {
00036   /* The effect is like a static for-loop with Is counting from 0 to
00037 dimension-1 and thus constructing a uniform initialization { }
00038 construction from each tuple element:
00039 { std::get<0>(t), std::get<1>(t), ..., std::get<dimension-1>(t) }
00040
00041 The static cast is here to avoid the warning when there is a loss
00042 of precision, for example when initializing an int from a float.
00043 */
00044   return { { static_cast<typename V::element_type>(std::get<Is>(t))...  } };
00045 }
00046
00047
00048 /** Construct an array from initializer elements provided as a tuple
00049 */
00050 template <typename V, typename Tuple>
00051 auto tuple_to_array(Tuple t) {
00052   /* Construct an index_sequence with 0, 1, ..., (size of the tuple-1)
00053 so that tuple_to_array_iterate can statically iterate on it */
00054   return tuple_to_array_iterate<V>(
00055       t, std::make_index_sequence<std::tuple_size_v<Tuple» {});
00056 }
00057
00058 /** Allows optional expansion of a 1-element tuple to a V::dimension
00059 tuple to replicate scalar values in vector initialization
00060 */
00061 template <typename V, typename Tuple, bool expansion = false>
00062 struct expand_to_vector {
00063   static_assert(V::dimension == std::tuple_size_v<Tuple>,
00064                 "The number of elements in initialization should match the "
00065                 "dimension of the vector");
00066
00067   // By default, act as a pass-through and do not do any expansion
00068   static auto expand(Tuple t) { return t; }
00069 };
00070
00071 /** Specialization in the case we ask for expansion */
00072 template <typename V, typename Tuple> struct expand_to_vector<V, Tuple, true> {
00073   static_assert(std::tuple_size_v<Tuple> == 1,
00074                 "Since it is a vector initialization from a scalar there "
00075                 "should be only one initializer value");
00076
00077 /** Construct a tuple from a value
00078
00079 \param value is used to initialize each tuple element
00080
00081 \param size is the number of elements of the tuple to be generated
00082
00083 The trick is to get the std::index_sequence<> that represent 0,
00084 1,..., dimension-1 as a variadic template pack Is that we can
00085 iterate on, in this function.
00086 */
00087   template <typename Value, std::size_t...  Is>
00088   static auto fill_tuple(Value e, std::index_sequence<Is...>) {
00089     /* The effect is like a static for-loop with Is counting from 0 to
00090 dimension-1 and thus replicating the pattern to have
00091 make_tuple( (0, e), (1, e), ...  (n - 1, e) )
00092
00093 Since the "," operator is just here to throw away the Is value
00094 (which is needed for the pack expansion...), at the end this is
00095 equivalent to:
00096 make_tuple( e, e, ..., e )
00097 */
00098     return std::make_tuple(((void)Is, e)...);
00099   }
00100
00101
00102 /** We expand the 1-element tuple by replicating into a tuple with the
00103 size of the vector */
00104   static auto expand(Tuple t) {
00105     return fill_tuple(std::get<0>(t),
00106                       std::make_index_sequence<V::dimension>{});
00107   }
00108 };
00109
00110 /** Create the array data of V from a tuple of initializer
00111
```

```
00112 If there is only 1 initializer, this is a scalar initialization of a
00113 vector and the value is expanded to all the vector elements first.
00114 */
00115 template <typename V, typename Tuple>
00116 auto expand(Tuple t) {
00117   return tuple_to_array<V>(expand_to_vector<V,
00118                            decltype(t),
00119                            /* Only ask the expansion to all vector
00120 element if there only a scalar
00121 initializer */
00122                            std::tuple_size_v<Tuple> == 1>{}.expand(t));
00123 }
00124
00125 }
00126
00127 /*
00128 # Some Emacs stuff:
00129 ### Local Variables:
00130 ### ispell-local-dictionary:  "american"
00131 ### eval:  (flyspell-prog-mode)
00132 ### End:
00133 */
00134
00135 #endif // TRISYCL_SYCL_DETAIL_ARRAY_TUPLE_HELPERS_HPP
```

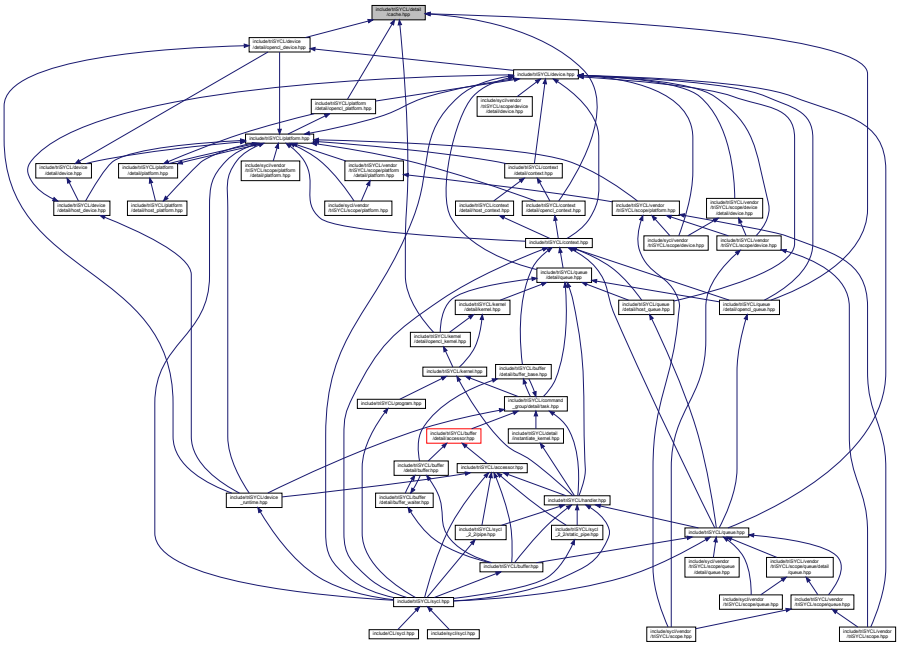
13.49 include/triSYCL/detail/cache.hpp File Reference

```
#include <memory>
#include <mutex>
#include <unordered_map>
```

Include dependency graph for cache.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::cache< Key, Value >

  A simple thread safe cache mechanism to cache std::shared_ptr of values indexed by keys.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.50 cache.hpp

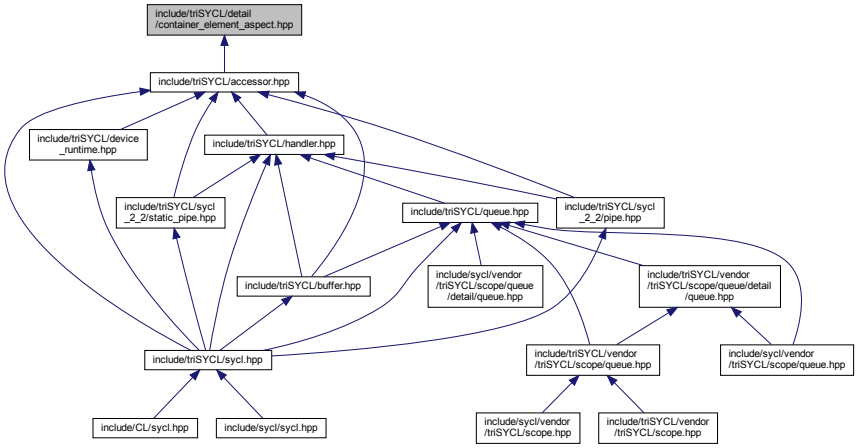
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_CACHE_HPP
00002 #define TRISYCL_SYCL_DETAIL_CACHE_HPP
00003
00004 /** \file A simple thread-safe cache
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <memory>
00013 #include <mutex>
00014 #include <unordered_map>
00015
00016 namespace trisycl::detail {
00017
00018
```

```
00019 /** A simple thread safe cache mechanism to cache std::shared_ptr of
00020 values indexed by keys
00021
00022 Since internally only std::weak_ptr are stored, this does not
00023 prevent object deletion but it is up to the programmer not to use
00024 this cache to retrieve deleted objects.
00025 */
00026 template <typename Key, typename Value>
00027 class cache {
00028
00029 public:
00030
00031 /// The type of the keys used to indexed the cache
00032   using key_type = Key;
00033
00034 /// The base type of the values stored in the cache
00035   using value_type = Value;
00036
00037 private:
00038
00039 /// The caching storage
00040   std::unordered_map<key_type, std::weak_ptr<value_type>> c;
00041
00042 /// To make the cache thread-safe
00043   std::mutex m;
00044
00045 public:
00046
00047 /** Get a value stored in the cache if present or insert by calling
00048 a generator function
00049
00050 \param[in] k is the key used to retrieve the value
00051
00052 \param[in] create_element is the function to be called if the
00053 key is not found in the cache to generate a value which is
00054 inserted for the key.  This function has to produce a value
00055 convertible to a shared_ptr
00056
00057 \return a shared_ptr to the value retrieved or inserted
00058 */
00059   template <typename Functor>
00060   std::shared_ptr<value_type> get_or_register(const key_type &k,
00061                                               Functor &&create_element) {
00062     std::lock_guard<std::mutex> lg { m };
00063
00064     auto i = c.find(k);
00065     if (i != c.end())
00066       if (auto observe = i->second.lock())
00067         // Returns \c shared_ptr only if target object is still alive
00068         return observe;
00069
00070     // Otherwise create and insert a new element
00071     std::shared_ptr<value_type> e { create_element() };
00072     c.insert({ k, e });
00073     return e;
00074   }
00075
00076
00077 /** Remove an entry from the cache
00078
00079 \param[in] k is the key associated to the value to remove from
00080 the cache
00081 */
00082   void remove(const key_type &k) {
00083     std::lock_guard<std::mutex> lg { m };
00084     c.erase(k);
00085   }
00086
00087 };
00088
00089 }
00090
00091 /*
00092 # Some Emacs stuff:
00093 ### Local Variables:
00094 ### ispell-local-dictionary:  "american"
00095 ### eval:  (flyspell-prog-mode)
00096 ### End:
00097 */
00098
00099 #endif // TRISYCL_SYCL_DEVICE_CACHE_HPP
```

13.51 include/triSYCL/detail/container_element_aspect.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::detail::container_element_aspect< T >

  A mix-in to add some container element aspects. *More...*

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.52 container_element_aspect.hpp

Go to the documentation of this file.

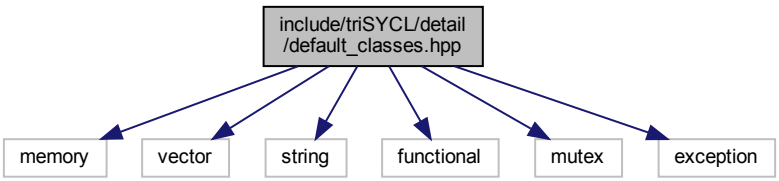
```
00001 #ifndef TRISYCL_SYCL_DETAIL_CONTAINER_ELEMENT_ASPECT_HPP
00002 #define TRISYCL_SYCL_DETAIL_CONTAINER_ELEMENT_ASPECT_HPP
00003
00004 /** \file Implement basic types à la STL related to container
00005 elements, such as value_type, reference...
00006
00007 Ronan at Keryell point FR
00008
00009 This file is distributed under the University of Illinois Open Source
00010 License.  See LICENSE.TXT for details.
00011 */
00012
00013 namespace trisycl::detail {
00014
00015 /** \addtogroup helpers Some helpers for the implementation
00016 @{
00017 */
00018
00019 /// A mix-in to add some container element aspects
```

```
00020 template <typename T>
00021 struct container_element_aspect {
00022
00023   using value_type = T;
00024   using pointer = value_type*;
00025   using const_pointer = const value_type*;
00026   using reference = value_type&;
00027   using const_reference = const value_type&;
00028
00029 };
00030
00031 /// @} End the helpers Doxygen group
00032
00033 }
00034
00035 /*
00036 # Some Emacs stuff:
00037 ### Local Variables:
00038 ### ispell-local-dictionary:  "american"
00039 ### eval:  (flyspell-prog-mode)
00040 ### End:
00041 */
00042
00043 #endif // TRISYCL_SYCL_DETAIL_CONTAINER_ELEMENT_ASPECT_HPP
```

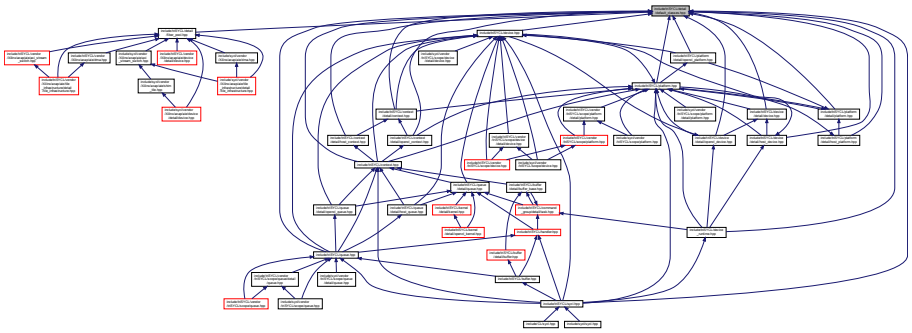
13.53 include/triSYCL/detail/default_classes.hpp File Reference

```
#include <memory>
#include <vector>
#include <string>
#include <functional>
#include <mutex>
#include <exception>
```

Include dependency graph for default_classes.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

Typedefs

- template<class T , class Alloc = std::allocator<T>>
  using trisycl::vector_class = std::vector< T, Alloc >
- using trisycl::string_class = std::string
- template<class R , class... ArgTypes>
  using trisycl::function_class = std::function< R(ArgTypes...)>
- using trisycl::mutex_class = std::mutex
- template<class T , class D = std::default_delete<T>>
  using trisycl::unique_ptr_class = std::unique_ptr< T[ ], D >
- template<class T >
  using trisycl::shared_ptr_class = std::shared_ptr< T >
- template<class T >
  using trisycl::weak_ptr_class = std::weak_ptr< T >
- template<class T >
  using trisycl::hash_class = std::hash< T >
- using trisycl::exception_ptr_class = std::exception_ptr

13.54 default_classes.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_DEFAULT_CLASSES_HPP
00002 #define TRISYCL_SYCL_DETAIL_DEFAULT_CLASSES_HPP
00003
00004 /** \file The OpenCL SYCL default classes to use from the STL according to
00005 section 3.2 of SYCL 1.2 specification
00006
00007 Ronan at Keryell point FR
00008
00009 This file is distributed under the University of Illinois Open Source
00010 License.  See LICENSE.TXT for details.
00011 */
00012
00013 /** \addtogroup defaults Manage default configuration and types
00014 @{
00015 */
00016
00017 #ifndef CL_SYCL_NO_STD_VECTOR
00018 /** The vector type to be used as SYCL vector
00019 */
00020 #include <memory>
00021 #include <vector>
00022 namespace trisycl {
00023
00024 template <class T, class Alloc = std::allocator<T»
00025 using vector_class = std::vector<T, Alloc>;
00026
00027 }
00028 #endif
00029
00030
00031 #ifndef CL_SYCL_NO_STD_STRING
00032 /** The string type to be used as SYCL string
00033 */
00034 #include <string>
00035 namespace trisycl {
00036
00037 using string_class = std::string;
00038
00039 }
00040 #endif
00041
```

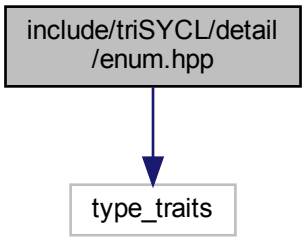
```
00042
00043 #ifndef CL_SYCL_NO_STD_FUNCTION
00044 /** The functional type to be used as SYCL function
00045 */
00046 #include <functional>
00047 namespace trisycl {
00048
00049 template <class R, class...  ArgTypes>
00050 using function_class = std::function<R(ArgTypes...)>;
00051
00052 }
00053 #endif
00054
00055
00056 #ifndef CL_SYCL_NO_STD_MUTEX
00057 /** The mutex type to be used as SYCL mutex
00058 */
00059 #include <mutex>
00060 namespace trisycl {
00061
00062 using mutex_class = std::mutex;
00063
00064 }
00065 #endif
00066
00067
00068 #ifndef CL_SYCL_NO_STD_UNIQUE_PTR
00069 /** The unique pointer type to be used as SYCL unique pointer
00070 */
00071 #include <memory>
00072 namespace trisycl {
00073
00074 template <class T, class D = std::default_delete<T»
00075 using unique_ptr_class = std::unique_ptr<T[], D>;
00076
00077 }
00078 #endif
00079
00080
00081 #ifndef CL_SYCL_NO_STD_SHARED_PTR
00082 /** The shared pointer type to be used as SYCL shared pointer
00083 */
00084 #include <memory>
00085 namespace trisycl {
00086
00087 template <class T>
00088 using shared_ptr_class = std::shared_ptr<T>;
00089
00090 }
00091 #endif
00092
00093
00094 #ifndef CL_SYCL_NO_STD_WEAK_PTR
00095 /** The weak pointer type to be used as SYCL weak pointer
00096 */
00097 #include <memory>
00098 namespace trisycl {
00099
00100 template <class T>
00101 using weak_ptr_class = std::weak_ptr<T>;
00102
00103 }
00104 #endif
00105
00106
00107 #ifndef CL_SYCL_NO_STD_HASH
00108 /** The hash type to be used as SYCL hash
00109 */
00110 #include <functional>
00111 namespace trisycl {
00112
00113 template <class T>
00114 using hash_class = std::hash<T>;
00115
00116 }
00117 #endif
00118
00119
00120 #ifndef CL_SYCL_NO_STD_EXCEPT
00121 /** The exception pointer type to be used as SYCL exception pointer
00122 */
00123 #include <exception>
00124 namespace trisycl {
00125
00126 using exception_ptr_class = std::exception_ptr;
00127
00128 }
```

```
00129 #endif
00130
00131
00132 /// @} End the defaults Doxygen group
00133
00134 /*
00135 # Some Emacs stuff:
00136 ### Local Variables:
00137 ### ispell-local-dictionary:  "american"
00138 ### eval:  (flyspell-prog-mode)
00139 ### End:
00140 */
00141
00142 #endif // TRISYCL_SYCL_DETAIL_DEFAULT_CLASSES_HPP
```

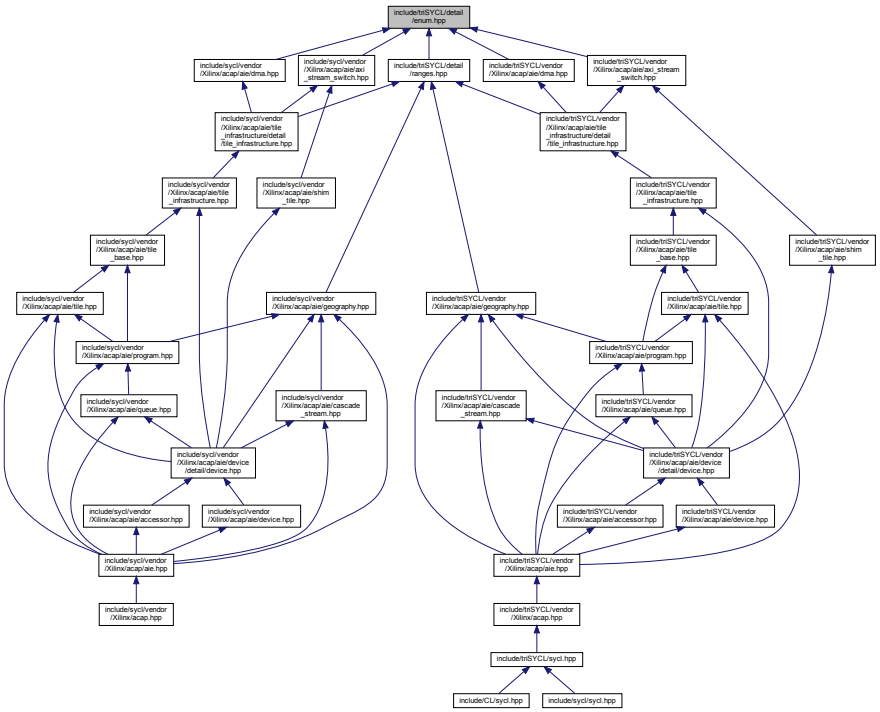
13.55 include/triSYCL/detail/enum.hpp File Reference

Some enum-related helpers.

`#include <type_traits>`
Include dependency graph for enum.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Variables

- auto constexpr trisycl::detail::underlying_value

  Get the enum value as an integral value of the underlying integer type.
- auto constexpr trisycl::detail::enum_distance

  Get the distance between 2 enum underlying values.
- auto constexpr trisycl::detail::enum_count

  Count the number of enums between 2 enum boundaries, including the boundaries.

13.55.1 Detailed Description

Some enum-related helpers.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file enum.hpp.

13.56 enum.hpp

Go to the documentation of this file.

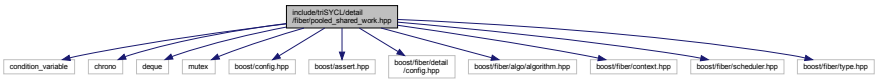
```
00001 #ifndef TRISYCL_SYCL_DETAIL_ENUM_HPP
00002 #define TRISYCL_SYCL_DETAIL_ENUM_HPP
00003
00004 /** \file
00005
00006 Some enum-related helpers
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include <type_traits>
00015
00016 namespace trisycl::detail {
00017
00018 /** Get the enum value as an integral value of the underlying integer
00019 type
00020 */
00021 auto constexpr underlying_value = [] (auto e) {
00022   using value_t = std::underlying_type_t<decltype(e)>;
00023   return static_cast<value_t>(e);
00024 };
00025
00026
00027 /// Get the distance between 2 enum underlying values
00028 auto constexpr enum_distance = [] (auto enum_start, auto enum_end) {
00029   static_assert(std::is_same_v<decltype(enum_start), decltype(enum_end)>,
00030                 "enum_start and auto enum_end should have the same type");
00031   return underlying_value(enum_end) - underlying_value(enum_start);
00032 };
00033
00034
00035 /** Count the number of enums between 2 enum boundaries,
00036 including the boundaries */
00037 auto constexpr enum_count  = [] (auto enum_start, auto enum_end) {
00038   return 1 + enum_distance(enum_start, enum_end);
00039 };
00040
00041
00042 }
00043
00044 /*
00045 # Some Emacs stuff:
00046 ### Local Variables:
00047 ### ispell-local-dictionary:  "american"
00048 ### eval:  (flyspell-prog-mode)
00049 ### End:
00050 */
00051
00052 #endif // TRISYCL_SYCL_DETAIL_ENUM_HPP
```

13.57 include/triSYCL/detail/fiber/pooled_shared_work.hpp File Reference

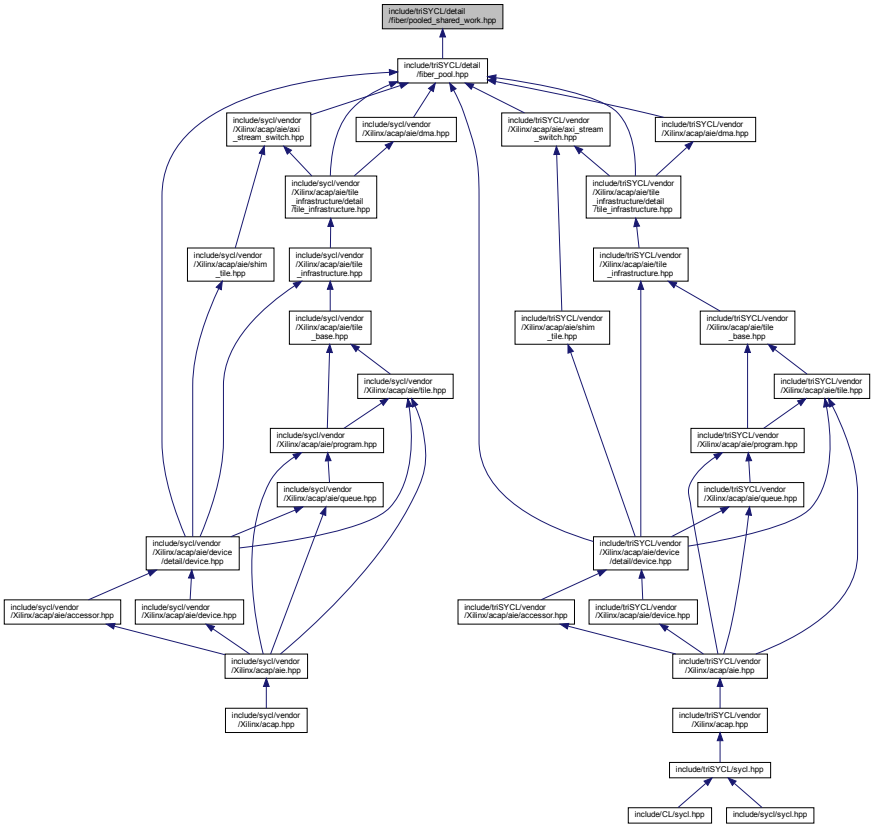
```
#include <condition_variable>
#include <chrono>
#include <deque>
#include <mutex>
#include <boost/config.hpp>
#include <boost/assert.hpp>
#include <boost/fiber/detail/config.hpp>
#include <boost/fiber/algo/algorithm.hpp>
#include <boost/fiber/context.hpp>
#include <boost/fiber/scheduler.hpp>
```

```
#include "boost/fiber/type.hpp"
```

Include dependency graph for pooled_shared_work.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::pooled_shared_work
- struct trisycl::detail::pooled_shared_work::pool_ctx

  Shared storage among the working threads.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.58 pooled_shared_work.hpp

Go to the documentation of this file.

```
00001 //          Copyright Nat Goodspeed + Oliver Kowalke 2015
00002 //                                  + Ronan Keryell 2020
00003 //
00004 // Distributed under the Boost Software License, Version 1.0.
00005 //    (See accompanying file LICENSE_1_0.txt or copy at
00006 //          http://www.boost.org/LICENSE_1_0.txt)
00007
00008 // This file is adapted from
00009 // https://github.com/boostorg/fiber/blob/develop/include/boost/fiber/algo/shared_work.hpp
00010 //
00011 //
00012 // It provides a scheduler sharing some work across a given number of
00013 // threads.
00014 //
00015 // The fiber pool can be stopped and several instances can coexist in
00016 // the same program.
00017 //
00018 // The main change to the original scheduler is to replace static
00019 // variables by a shared context.
00020
00021 #ifndef TRISYCL_SYCL_DETAIL_FIBER_POOLED_SHARED_WORK_H
00022 #define TRISYCL_SYCL_DETAIL_FIBER_POOLED_SHARED_WORK_H
00023
00024 #include <condition_variable>
00025 #include <chrono>
00026 #include <deque>
00027 #include <mutex>
00028
00029 #include <boost/config.hpp>
00030 #include <boost/assert.hpp>
00031
00032 #include <boost/fiber/detail/config.hpp>
00033 #include <boost/fiber/algo/algorithm.hpp>
00034 #include <boost/fiber/context.hpp>
00035 #include <boost/fiber/scheduler.hpp>
00036 #include "boost/fiber/type.hpp"
00037
00038 namespace trisycl::detail {
00039
00040 namespace bf = boost::fibers;
00041
00042 class pooled_shared_work :  public boost::fibers::algo::algorithm {
00043
00044   using rqueue_type = std::deque<bf::context *>;
00045   using lqueue_type = bf::scheduler::ready_queue_type;
00046
00047  public:
00048
00049 /// Shared storage among the working threads
00050   struct pool_ctx {
00051     pool_ctx(bool suspend) :  suspend_ { suspend }
00052     {}
00053
00054 /// Indicate if a thread without work goes to sleep instead of busy-waiting
00055     const bool suspend_;
00056
00057 /// The global queue storing the runnable fibers
00058     rqueue_type rqueue_ {};
00059
00060 /// For concurrent access to the global queue
00061     std::mutex rqueue_mtx_ {};
00062   };
00063
00064 /// Type tracking the common worker data
00065   using ctx = std::shared_ptr<pool_ctx>;
00066
00067  private:
00068
00069 /// Some shared datastructure among the working threads
00070   ctx pool_ctx_;
00071
00072 /// The runnable local fibers bound to the thread
00073   lqueue_type lqueue_ {};
00074
00075 /// The thread-local suspend/notify mechanics
00076   std::mutex mtx_ {};
00077   std::condition_variable cnd_ {};
00078   bool flag_ { false };
00079
00080  public:
00081
00082   static ctx
```

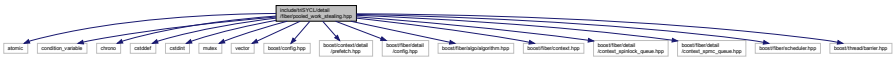
```
00083   create_pool_ctx(bool suspend) {
00084     return std::make_shared<pool_ctx>(suspend);
00085   }
00086 
00087 
00088   pooled_shared_work(const ctx &pc) :  pool_ctx_ { pc }
00089   {}
00090 
00091   pooled_shared_work(pooled_shared_work const&) = delete;
00092   pooled_shared_work(pooled_shared_work &&) = delete;
00093 
00094   pooled_shared_work & operator=(pooled_shared_work const&) = delete;
00095   pooled_shared_work & operator=(pooled_shared_work &&) = delete;
00096 
00097   void awakened(bf::context * ctx) noexcept override {
00098     if (ctx->is_context(bf::type::pinned_context)) { /*<
00099 recognize when we're passed this thread's main fiber (or an
00100 implicit library helper fiber):  never put those on the shared
00101 queue
00102 >*/
00103       lqueue_.push_back(*ctx);
00104     } else {
00105       ctx->detach();
00106       std::unique_lock lk { pool_ctx_->rqueue_mtx_ }; /*<
00107 worker fiber, enqueue on shared queue
00108 >*/
00109       pool_ctx_->rqueue_.push_back(ctx);
00110     }
00111   }
00112 
00113 
00114   bf::context * pick_next() noexceptoverride {
00115     bf::context * ctx = nullptr;
00116     std::unique_lock lk { pool_ctx_->rqueue_mtx_ };
00117     auto &rq = pool_ctx_->rqueue_;
00118     if (!rq.empty()) { /*<
00119 pop an item from the ready queue
00120 >*/
00121       ctx = rq.front();
00122       rq.pop_front();
00123       lk.unlock();
00124       BOOST_ASSERT(nullptr != ctx);
00125       bf::context::active()->attach(ctx); /*<
00126 attach context to current scheduler via the active fiber
00127 of this thread
00128 >*/
00129     } else {
00130       lk.unlock();
00131       if (!lqueue_.empty()) { /*<
00132 nothing in the ready queue, return main or dispatcher fiber
00133 >*/
00134         ctx = & lqueue_.front();
00135         lqueue_.pop_front();
00136       }
00137     }
00138     return ctx;
00139   }
00140 
00141 
00142   bool has_ready_fibers() const noexceptoverride {
00143     std::unique_lock lock { pool_ctx_->rqueue_mtx_ };
00144     return !pool_ctx_->rqueue_.empty() || !lqueue_.empty();
00145   }
00146 
00147 
00148   void suspend_until(std::chrono::steady_clock::time_point const& time_point)
00149     noexcept override {
00150     if (pool_ctx_->suspend_) {
00151       if (std::chrono::steady_clock::time_point::max() == time_point) {
00152         std::unique_lock lk { mtx_ };
00153         cnd_.wait(lk, [this](){ return flag_; });
00154         flag_ = false;
00155       } else {
00156         std::unique_lock lk { mtx_ };
00157         cnd_.wait_until(lk, time_point, [this](){ return flag_; });
00158         flag_ = false;
00159       }
00160     }
00161   }
00162 
00163 
00164   void notify() noexceptoverride {
00165     if (pool_ctx_->suspend_) {
00166       std::unique_lock lk { mtx_ };
00167       flag_ = true;
00168       lk.unlock();
00169       cnd_.notify_all();
```

```
00170     }
00171   }
00172
00173 };
00174
00175 }
00176
00177 #endif // TRISYCL_SYCL_DETAIL_FIBER_POOLED_SHARED_WORK_H
```

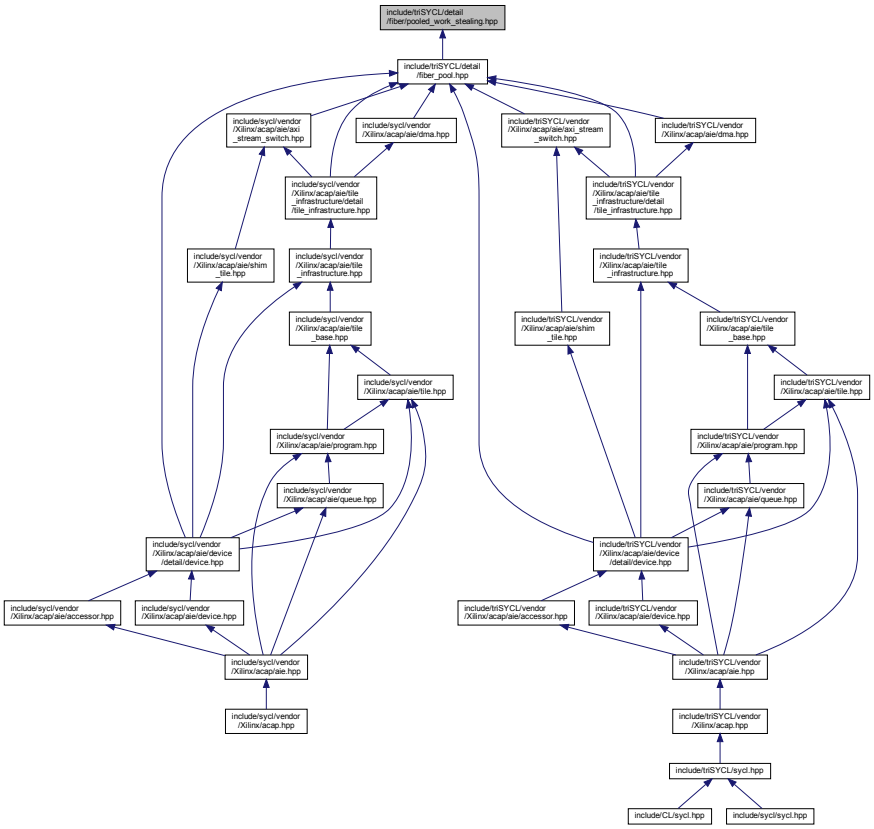
13.59 include/triSYCL/detail/fiber/pooled_work_stealing.hpp File Reference

```
#include <atomic>
#include <condition_variable>
#include <chrono>
#include <cstddef>
#include <cstdint>
#include <mutex>
#include <vector>
#include <boost/config.hpp>
#include <boost/context/detail/prefetch.hpp>
#include <boost/fiber/detail/config.hpp>
#include <boost/fiber/algo/algorithm.hpp>
#include <boost/fiber/context.hpp>
#include <boost/fiber/detail/context_spinlock_queue.hpp>
#include <boost/fiber/detail/context_spmc_queue.hpp>
#include <boost/fiber/scheduler.hpp>
#include <boost/thread/barrier.hpp>
```

Include dependency graph for pooled_work_stealing.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::pooled_work_stealing
- struct trisycl::detail::pooled_work_stealing::pool_ctx

  Shared storage among the working threads.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.60 pooled_work_stealing.hpp

Go to the documentation of this file.

```
00001 //          Copyright Oliver Kowalke 2015 / Lennart Braun 2019
00002 //          / Ronan Keryell 2020
00003 //
00004 // Distributed under the Boost Software License, Version 1.0.
00005 //    (See accompanying file LICENSE_1_0.txt or copy at
00006 //          http://www.boost.org/LICENSE_1_0.txt)
00007
```

```
00008 // This file is adapted from
00009 // https://github.com/boostorg/fiber/blob/develop/include/boost/fiber/algo/work_stealing.hpp
00010 //
00011 // It provides a work stealing scheduler across a given number of threads.
00012 //
00013 // The fiber pool can be stopped and several instances can coexist in
00014 // the same program.
00015 //
00016 // The main change to the original scheduler is to replace static
00017 // variables by a shared context.
00018
00019 #ifndef BOOST_FIBERS_ALGO_POOLED_WORK_STEALING_H
00020 #define BOOST_FIBERS_ALGO_POOLED_WORK_STEALING_H
00021
00022 #include <atomic>
00023 #include <condition_variable>
00024 #include <chrono>
00025 #include <cstddef>
00026 #include <cstdint>
00027 #include <mutex>
00028 #include <vector>
00029
00030 #include <boost/config.hpp>
00031 #include <boost/context/detail/prefetch.hpp>
00032 #include <boost/fiber/detail/config.hpp>
00033 #include <boost/fiber/algo/algorithm.hpp>
00034 #include <boost/fiber/context.hpp>
00035 #include <boost/fiber/detail/context_spinlock_queue.hpp>
00036 #include <boost/fiber/detail/context_spmc_queue.hpp>
00037 #include <boost/fiber/scheduler.hpp>
00038 #include <boost/thread/barrier.hpp>
00039
00040 namespace trisycl::detail {
00041
00042 class pooled_work_stealing :  public boost::fibers::algo::algorithm {
00043
00044  public:
00045
00046 /// Shared storage among the working threads
00047   struct pool_ctx {
00048     pool_ctx(std::uint32_t thread_count, bool suspend)
00049       :  thread_count_ { thread_count }
00050       , suspend_ { suspend }
00051       , schedulers_ { thread_count, nullptr }
00052       , barrier_ { thread_count }
00053     {}
00054
00055 /// Number of threads in the worker pool
00056     const std::uint32_t thread_count_;
00057
00058 /// Indicate if a thread without work goes to sleep instead of busy-waiting
00059     const bool suspend_;
00060
00061 /// Counter used to give a unique id_ to the worker
00062     std::atomic<std::uint32_t> counter_ = 0;
00063
00064 /// Keep track of each worker scheduler
00065     std::vector<boost::intrusive_ptr<pooled_work_stealing» schedulers_;
00066
00067 /// Synchronize all the working thread after starting and before finishing
00068     boost::barrier barrier_;
00069   };
00070
00071 /// Type tracking the common worker data
00072   using ctx = std::shared_ptr<pool_ctx>;
00073
00074  private:
00075
00076 /// Some shared datastructure among the working threads
00077   ctx pool_ctx_;
00078
00079 /// The thread order in the working pool.  0 is first starting thread
00080   std::uint32_t id_;
00081
00082 /// The queue of thread-local runnable fibers
00083 #ifdef BOOST_FIBERS_USE_SPMC_QUEUE
00084   boost::fibers::detail::context_spmc_queue rqueue_ {};
00085 #else
00086   boost::fibers::detail::context_spinlock_queue rqueue_ {};
00087 #endif
00088
00089 /// The thread-local suspend/notify mechanics
00090   std::mutex mtx_ {};
00091   std::condition_variable cnd_ {};
00092   bool flag_ { false };
00093
00094  public:
```

```
00095
00096   static ctx
00097   create_pool_ctx(std::uint32_t thread_count, bool suspend) {
00098     return std::make_shared<pool_ctx>(thread_count, suspend);
00099   }
00100
00101
00102   pooled_work_stealing(const ctx &pc)
00103     :  pool_ctx_ { pc }
00104     , id_ { pool_ctx_->counter_++ } {
00105       pool_ctx_->schedulers_[id_] = this;
00106       pool_ctx_->barrier_.wait();
00107     }
00108
00109
00110   // This will not be called until all other fibers are ready.
00111   ~pooled_work_stealing() = default;
00112
00113   pooled_work_stealing(pooled_work_stealing const&) = delete;
00114   pooled_work_stealing(pooled_work_stealing &&) = delete;
00115
00116   pooled_work_stealing & operator=(pooled_work_stealing const&) = delete;
00117   pooled_work_stealing & operator=(pooled_work_stealing &&) = delete;
00118
00119
00120   void awakened(boost::fibers::context * ctx) noexcept override {
00121     if (!ctx->is_context(boost::fibers::type::pinned_context))
00122       ctx->detach();
00123     rqueue_.push(ctx);
00124   }
00125
00126
00127   bf::context * pick_next() noexceptoverride {
00128     bf::context * victim = rqueue_.pop();
00129     if (nullptr != victim) {
00130       boost::context::detail::prefetch_range(victim, sizeof(*victim));
00131       if (!victim->is_context(bf::type::pinned_context)) {
00132         bf::context::active()->attach(victim);
00133       }
00134     }
00135     else {
00136       std::size_t size = pool_ctx_->thread_count_;
00137       //  Work stealing is only possible with more than 1 thread
00138       if (BOOST_LIKELY(size > 1)) {
00139         std::size_t count = 0;
00140         static thread_local std::minstd_rand generator { std::random_device{}() };
00141         std::uniform_int_distribution< std::uint32_t > distribution
00142           { 0, static_cast<std::uint32_t>(size - 1) };
00143         do {
00144           std::uint32_t id = 0;
00145           do {
00146             ++count;
00147             // Random selection of one logical CPU that belongs to the
00148             // local NUMA node
00149             id = distribution(generator);
00150             // Prevent stealing from own scheduler
00151           } while (id == id_);
00152           // Steal context from other scheduler
00153           victim = pool_ctx_->schedulers_[id]->steal();
00154         } while (nullptr == victim && count < size);
00155         if (nullptr != victim) {
00156           boost::context::detail::prefetch_range(victim, sizeof(*victim));
00157           BOOST_ASSERT(!victim->is_context(bf::type::pinned_context));
00158           bf::context::active()->attach(victim);
00159         }
00160       }
00161     }
00162     return victim;
00163   }
00164
00165
00166   virtual boost::fibers::context * steal() noexcept {
00167     return rqueue_.steal();
00168   }
00169
00170
00171   bool has_ready_fibers() const noexceptoverride {
00172     return !rqueue_.empty();
00173   }
00174
00175
00176   void suspend_until(std::chrono::steady_clock::time_point const& time_point)
00177     noexcept override {
00178     if (pool_ctx_->suspend_) {
00179       if (std::chrono::steady_clock::time_point::max() == time_point) {
00180         std::unique_lock lk { mtx_ };
00181         cnd_.wait(lk, [&] { return flag_; });
```

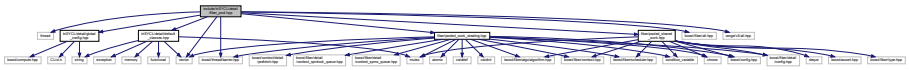
```
00182         flag_ = false;
00183       }
00184       else {
00185         std::unique_lock lk { mtx_ };
00186         cnd_.wait_until(lk, time_point, [&] { return flag_; });
00187         flag_ = false;
00188       }
00189     }
00190   }
00191
00192
00193   void notify() noexceptoverride {
00194     if (pool_ctx_->suspend_) {
00195       std::unique_lock lk { mtx_ };
00196       flag_ = true;
00197       lk.unlock();
00198       cnd_.notify_all();
00199     }
00200   }
00201
00202 };
00203
00204 }
00205
00206 #endif //  TRISYCL_SYCL_DETAIL_FIBER_POOLED_WORK_STEALING_H
```

13.61 include/triSYCL/detail/fiber_pool.hpp File Reference

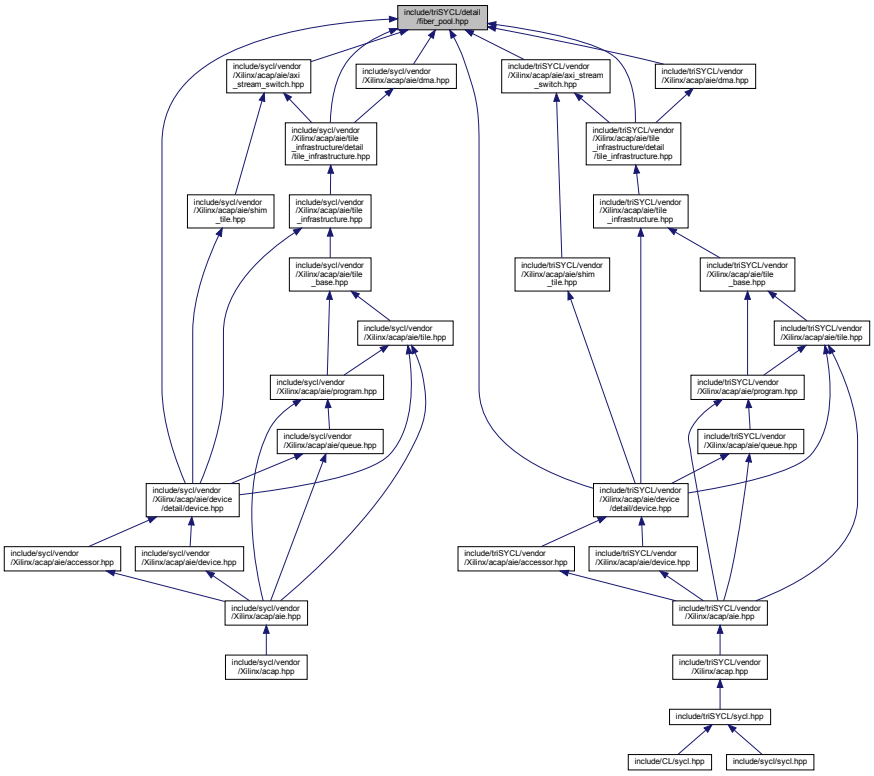
A Boost.Fiber pool implemented on a std::thread pool.

```
#include <thread>
#include <vector>
#include <boost/fiber/all.hpp>
#include <boost/thread/barrier.hpp>
#include <range/v3/all.hpp>
#include "triSYCL/detail/global_config.hpp"
#include "triSYCL/detail/default_classes.hpp"
#include "fiber/pooled_shared_work.hpp"
#include "fiber/pooled_work_stealing.hpp"
```

Include dependency graph for fiber_pool.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::fiber_pool

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.61.1 Detailed Description

A Boost.Fiber pool implemented on a std::thread pool.

The use case is for circuit emulation when there are a lot of fibers launched at the beginning and they have to run concurrently.

The fiber executors can use several std::thread.

It allows the execution of some callable and return some future for later shepherding.

Ronan at Keryell point FR

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file fiber_pool.hpp.

13.62 fiber_pool.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_FIBER_POOL_HPP
00002 #define TRISYCL_SYCL_DETAIL_FIBER_POOL_HPP
00003
00004 /** \file
00005
00006 A Boost.Fiber pool implemented on a std::thread pool
00007
00008 The use case is for circuit emulation when there are a lot of
00009 fibers launched at the beginning and they have to run concurrently.
00010
00011 The fiber executors can use several std::thread.
00012
00013 It allows the execution of some callable and return some future
00014 for later shepherding.
00015
00016
00017 Ronan at Keryell point FR
00018
00019 This file is distributed under the University of Illinois Open Source
00020 License.  See LICENSE.TXT for details.
00021 */
00022
00023 #include <thread>
00024 #include <vector>
00025 #include <boost/fiber/all.hpp>
00026 #include <boost/thread/barrier.hpp>
00027 #include <range/v3/all.hpp>
00028
00029 /** Some global triSYCL configuration */
00030 #include "triSYCL/detail/global_config.hpp"
00031 #include "triSYCL/detail/default_classes.hpp"
00032
00033 #include "fiber/pooled_shared_work.hpp"
00034 #include "fiber/pooled_work_stealing.hpp"
00035
00036 namespace trisycl::detail {
00037
00038 class fiber_pool {
00039
00040 public:
00041
00042 /// To select some various Boost.Fibers schedulers
00043   enum class sched {
00044     round_robin,
00045     shared_work,
00046     work_stealing
00047     // \todo Add numa
00048   };
00049
00050 private:
00051
00052 /// The thread running the Boost.Fiber schedulers to do the work
00053   std::vector<std::future<void>> working_threads;
00054
00055 /// The queue to submit work
00056   boost::fibers::unbuffered_channel<std::function<void(void)>> submission;
00057
00058   //static auto constexpr starting_mode = boost::fibers::launch::post;
00059   static auto constexpr starting_mode = boost::fibers::launch::dispatch;
00060
00061 /// To synchronize all the threads before they can run some fibers
00062   boost::barrier starting_block;
00063
00064 /// To synchronize all the threads after their last fiber
00065   boost::fibers::barrier finish_line;
00066
00067 /// To avoid joining several times
00068   bool joinable = true;
00069
00070 /// The model of scheduler
00071   sched s;
00072
00073   // Pool context for the work-stealing scheduler
00074   pooled_work_stealing::ctx pc_stealing;
00075
00076   // Pool context for the work-sharing scheduler
00077   pooled_shared_work::ctx pc_shared;
00078
00079 public:
00080
00081 /// The type of the future used by the implementation underneath
00082   template <typename T>
```

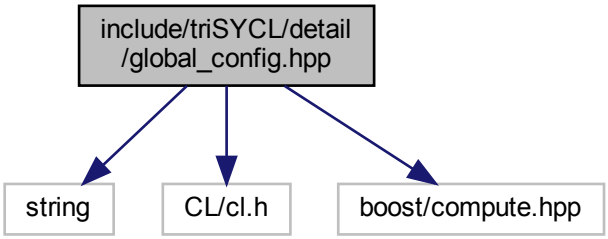
```
00083   using future = boost::fibers::future<T>;
00084
00085 /// Create a fiber_pool
00086   fiber_pool(int thread_number,
00087              sched scheduler,
00088              bool suspend)
00089     :  starting_block { static_cast<unsigned int>(thread_number) + 1 }
00090     , finish_line { static_cast<unsigned int>(thread_number) }
00091     , s { scheduler }
00092   {
00093     if (scheduler == sched::shared_work)
00094       // This scheduler needs a shared context
00095       pc_shared = pooled_shared_work::create_pool_ctx(suspend);
00096     else if (scheduler == sched::work_stealing)
00097       // This scheduler needs a shared context
00098       pc_stealing = pooled_work_stealing::create_pool_ctx
00099         (thread_number, suspend);
00100     // Start the working threads
00101     working_threads = ranges::iota_view { 0, thread_number }
00102                     | ranges::views::transform([&] (int i) {
00103                         return std::async(std::launch::async,
00104                                           [&, i] { run(i); }); })
00105                     | ranges::to<std::vector>;
00106     // Wait for all thread workers to be ready
00107     starting_block.count_down_and_wait();
00108   }
00109
00110
00111 /** Submit some work on a new fiber
00112
00113 \param[in] work is the callable to execute, taking no arguments
00114 and returning a result of some type R
00115
00116 \return a future<R>
00117 */
00118   template <typename Callable>
00119   auto submit(Callable && work) {
00120     // Put the packaged_task into a shared_ptr to fit it later in a
00121     // lambda in a std::function since the type is move-only
00122     auto pt = std::make_shared<boost::fibers::packaged_task<void(void)»
00123       ([w = std::forward<Callable>(work)] () mutable { return w(); });
00124     auto f = pt->get_future();
00125     // Submit a lambda to do the type erasure so the submission queue
00126     // type is independent of the type of the packaged_task
00127     submission.push([p = std::move(pt)] () mutable { (*p)(); });
00128     // Return the future to the client to get the result or the exception
00129     return f;
00130   }
00131
00132
00133 /// Close the submission
00134   void close() {
00135     // Can be done many times, so no protection required here
00136     submission.close();
00137   }
00138
00139
00140 /// Wait for all the threads are done
00141   void join() {
00142     // Can be done only once
00143     if (joinable) {
00144       // Close the submission if not done already
00145       close();
00146       for (auto &t :  working_threads)
00147         // A Boost.Fiber scheduler will block its thread if they still
00148         // have some work to do
00149         t.get();
00150       joinable = false;
00151     }
00152   }
00153
00154
00155 /// Wait for some remaining work to be done
00156   ~fiber_pool() {
00157     // Join first if not done already
00158     join();
00159     // There is a circular dependecy of shared_ptr/intrusive_ptr between
00160     // pc_stealing and its schedulers_.  after we join we know none of the
00161     // schedulers_ are still running so we break the cycle and everything
00162     // cleansup itself.
00163     if (pc_stealing)
00164       pc_stealing->schedulers_.clear();
00165   }
00166
00167 private:
00168
00169 /// The thread worker job
```

```
00170   void run(int i) {
00171     if (s == sched::shared_work)
00172       boost::fibers::use_scheduling_algorithm<pooled_shared_work>(pc_shared);
00173     else if (s == sched::work_stealing)
00174       boost::fibers::use_scheduling_algorithm
00175         <pooled_work_stealing>(pc_stealing);
00176     // Otherwise a round-robin scheduler is used and the fibers will
00177     // use only 1 thread since there is no thread migration in that
00178     // case
00179
00180     // Wait for all thread workers to be ready
00181     starting_block.count_down_and_wait();
00182
00183     // Only the first thread receives and starts the work
00184     if (i == 0) {
00185       // Keep track of each fiber execution to avoid quitting before completion
00186       std::vector<boost::fibers::fiber> fibers;
00187       for (;;) {
00188         decltype(submission)::value_type work;
00189         if (submission.pop(work)
00190             == boost::fibers::channel_op_status::closed)
00191           // Someone asked to stop accepting work
00192           break;
00193         // Launch the work on a new fiber
00194         fibers.emplace_back(starting_mode, std::move(work));
00195       }
00196       // Now wait for the completion of each fiber
00197       for (auto &f :  fibers)
00198         f.join();
00199     }
00200     // Wait for all the threads to finish their fiber execution
00201     finish_line.wait();
00202   }
00203
00204 };
00205
00206 }
00207
00208 /*
00209 # Some Emacs stuff:
00210 ### Local Variables:
00211 ### ispell-local-dictionary:  "american"
00212 ### eval:  (flyspell-prog-mode)
00213 ### End:
00214 */
00215
00216 #endif // TRISYCL_SYCL_DETAIL_FIBER_POOL_HPP
```

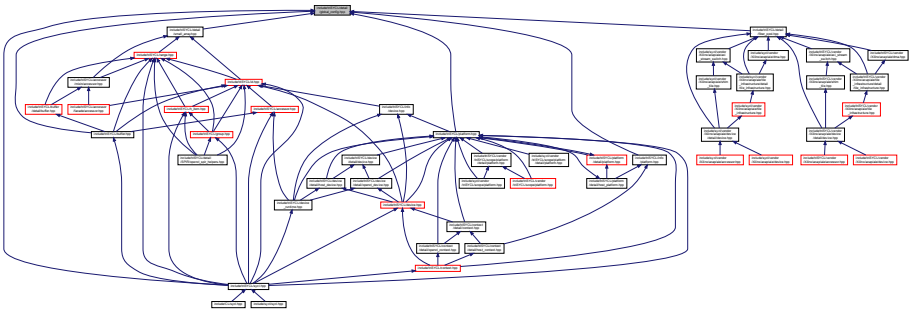
13.63 include/triSYCL/detail/global_config.hpp File Reference

```
#include <string>
#include <CL/cl.h>
#include <boost/compute.hpp>
```

Include dependency graph for global_config.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Macros

- #define CL_SYCL_LANGUAGE_VERSION 121

  This implements SYCL 1.2.1.
- #define TRISYCL_CL_LANGUAGE_VERSION 121

  This implements triSYCL 1.2.1.
- #define BOOST_COMPUTE_CL_VERSION_1_2
- #define TRISYCL_SYCL_NAMESPACE trisycl

  Define what is the real SYCL namespace to use.
- #define __SYCL_SINGLE_SOURCE__

  This source is compiled by a single source compiler.
- #define TRISYCL_MAKE_BOOST_CIRCULARBUFFER_THREAD_SAFE
- #define TRISYCL_OPENCL_ONLY(...) __VA_ARGS__

  Define TRISYCL_OPENCL to add OpenCL.
- #define TRISYCL_DEVICE_ONLY(...)
- #define TRISYCL_DEBUG_ONLY(...) __VA_ARGS__
- #define TRISYCL_WEAK_ATTRIB_PREFIX
- #define TRISYCL_WEAK_ATTRIB_SUFFIX __attribute__((weak))

Variables

- constexpr bool trisycl::detail::allow_any_dimension

  Map TRISYCL_ALLOW_ANY_DIMENSION to a constexpr variable.
- constexpr bool trisycl::detail::use_native_work_item

  Map TRISYCL_USE_OPENCL_ND_RANGE to a constexpr variable.

13.63.1 Macro Definition Documentation

13.63.1.1 TRISYCL_SYCL_NAMESPACE

```
#define TRISYCL_SYCL_NAMESPACE trisycl
```

Define what is the real SYCL namespace to use.

Definition at line 44 of file global_config.hpp.

13.63.1.2 TRISYCL_WEAK_ATTRIB_PREFIX

```
#define TRISYCL_WEAK_ATTRIB_PREFIX
```

Definition at line 119 of file global_config.hpp.

13.63.1.3 TRISYCL_WEAK_ATTRIB_SUFFIX

```
#define TRISYCL_WEAK_ATTRIB_SUFFIX __attribute__((weak))
```

Definition at line 120 of file global_config.hpp.

13.64 global_config.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_GLOBAL_CONFIG_HPP
00002 #define TRISYCL_SYCL_DETAIL_GLOBAL_CONFIG_HPP
00003
00004 /** \file The OpenCL SYCL details on the global triSYCL configuration
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <string>
00013
00014 /** \addtogroup defaults Manage default configuration and types
00015 @{
00016 */
00017
00018 // The following symbols can be set to implement a different version
00019 #ifndef CL_SYCL_LANGUAGE_VERSION
00020 /// This implements SYCL 1.2.1
00021 #define CL_SYCL_LANGUAGE_VERSION 121
00022 #endif
00023
00024 #ifndef TRISYCL_CL_LANGUAGE_VERSION
00025 /// This implements triSYCL 1.2.1
00026 #define TRISYCL_CL_LANGUAGE_VERSION 121
00027 #endif
00028
00029 #define BOOST_COMPUTE_CL_VERSION_1_2
00030
00031 namespace trisycl::detail {
00032 /// Map TRISYCL_ALLOW_ANY_DIMENSION to a constexpr variable
00033   constexpr bool allow_any_dimension =
00034 #ifdef TRISYCL_ALLOW_ANY_DIMENSION
00035 /// Allow any dimension in SYCL object, not only between 1 and 3
00036     true
00037 #else
```

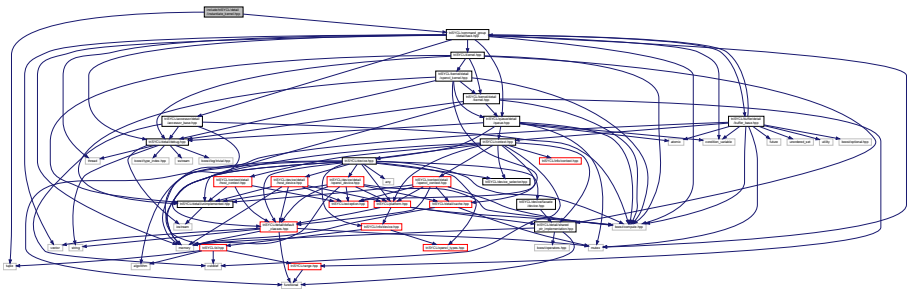
```
00038     false
00039 #endif
00040     ;
00041
00042 /// Define what is the real SYCL namespace to use
00043 #ifndef TRISYCL_SYCL_NAMESPACE
00044 #define TRISYCL_SYCL_NAMESPACE trisycl
00045 #endif
00046
00047 /// Map TRISYCL_USE_OPENCL_ND_RANGE to a constexpr variable
00048   constexpr bool use_native_work_item =
00049 #ifdef TRISYCL_USE_OPENCL_ND_RANGE
00050     // Use the OpenCL-like native mechanism to provide work-item dimensions
00051     true
00052 #else
00053     // Emulate work-item dimensions with a loop nest
00054     false
00055 #endif
00056     ;
00057   } // namespace trisycl::detail
00058
00059 /// This source is compiled by a single source compiler
00060 #define __SYCL_SINGLE_SOURCE__
00061
00062
00063 /* Work-around an old Boost.CircularBuffer bug if a pre 1.62 Boost
00064 version is used */
00065 #define TRISYCL_MAKE_BOOST_CIRCULARBUFFER_THREAD_SAFE
00066
00067
00068 /** Define TRISYCL_OPENCL to add OpenCL
00069
00070 triSYCL can indeed work without OpenCL if only host support is needed.
00071 */
00072 #ifdef TRISYCL_OPENCL
00073
00074 // SYCL interoperation API with OpenCL requires some OpenCL C types:
00075 #if defined(__APPLE__)
00076 #include <OpenCL/cl.h>
00077 #else
00078 #include <CL/cl.h>
00079 #endif
00080 // But the triSYCL OpenCL implementation is actually based on Boost.Compute
00081 #include <boost/compute.hpp>
00082 /// A macro to keep some stuff in OpenCL mode
00083 // Use variadic macros so we can have stray commas in the argument
00084 #define TRISYCL_OPENCL_ONLY(...)  __VA_ARGS__
00085 #else
00086 /// A macro to skip stuff when not supporting OpenCL
00087 #define TRISYCL_OPENCL_ONLY(...)
00088 #endif
00089
00090 #ifdef TRISYCL_DEVICE
00091 /// Define the SYCL marker according to specification
00092 #define __SYCL_DEVICE_ONLY__
00093 // To keep something when in device mode
00094 // Use variadic macros so we can have stray commas in the argument
00095 #define TRISYCL_DEVICE_ONLY(...)  __VA_ARGS__
00096 #else
00097 // Do not keep when not in device mode
00098 #define TRISYCL_DEVICE_ONLY(...)
00099 #endif
00100
00101 #ifdef TRISYCL_DEBUG
00102 // To keep something when in debug mode only
00103 // Use variadic macros so we can have stray commas in the argument
00104 #define TRISYCL_DEBUG_ONLY(...)  __VA_ARGS__
00105 #else
00106 // Do not keep when not in debug mode
00107 #define TRISYCL_DEBUG_ONLY(...)
00108 #endif
00109
00110
00111 /// @} End the defaults Doxygen group
00112
00113 // Compiler specific weak linking (until changing to C++17 inline variables/functions)
00114 #ifndef TRISYCL_WEAK_ATTRIB_PREFIX
00115 #ifdef _MSC_VER
00116 #define TRISYCL_WEAK_ATTRIB_PREFIX __declspec(selectany)
00117 #define TRISYCL_WEAK_ATTRIB_SUFFIX
00118 #else
00119 #define TRISYCL_WEAK_ATTRIB_PREFIX
00120 #define TRISYCL_WEAK_ATTRIB_SUFFIX __attribute__((weak))
00121 #endif
00122 #endif
00123
00124 // Suppress usage/leak of macros originating from Visual C++ headers
```

```
00125 #ifdef _MSC_VER
00126 #define NOMINMAX
00127 #endif
00128
00129
00130 namespace trisycl {
00131   // Inside this implementation.  make the predefined string literals available
00132   using namespace std::literals::string_literals;
00133 }
00134
00135
00136 /*
00137 # Some Emacs stuff:
00138 ### Local Variables:
00139 ### ispell-local-dictionary:  "american"
00140 ### eval:  (flyspell-prog-mode)
00141 ### End:
00142 */
00143
00144 #endif // TRISYCL_SYCL_DETAIL_GLOBAL_CONFIG_HPP
```

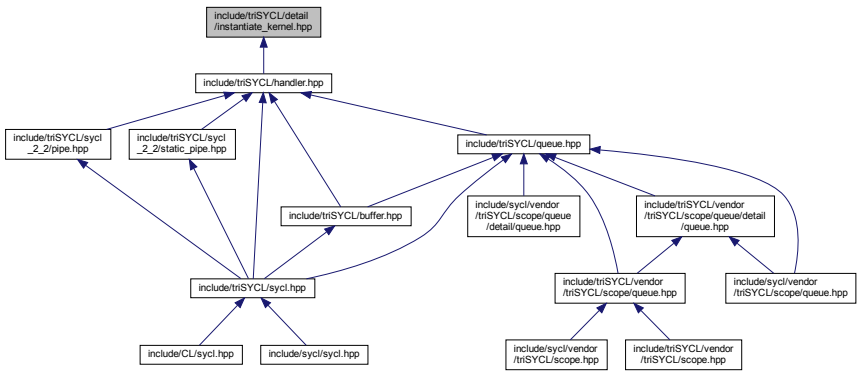
13.65 include/triSYCL/detail/instantiate_kernel.hpp File Reference

```
#include <tuple>
#include "triSYCL/command_group/detail/task.hpp"
```

Include dependency graph for instantiate_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Functions

- template<typename KernelName , typename Functor >
  void trisycl::detail::instantiate_kernel (Functor f) noexcept

  Instantiate the kernel code.
- TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX trisycl::detail::set_kernel_task_marker (detail::task &) noexcept

  Set the task to be used to launch the kernel.
- template<typename KernelName , typename Kernel >
  void trisycl::detail::launch_device_kernel (detail::task &t, Kernel k) noexcept

  Gather in a single function what is necessary for the device compiler to outline the kernel or to set up the kernel and serialize the arguments.

Variables

- auto trisycl::detail::prevent_arguments_from_optimization

  Avoid the interprocedural optimization to remove these arguments in the kernel instantiation by relying on some dummy static variables.

13.66 instantiate_kernel.hpp

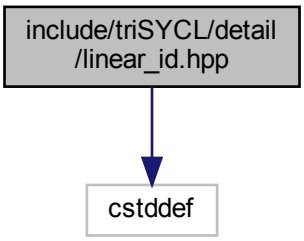
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_INSTANTIATE_KERNEL_HPP
00002 #define TRISYCL_SYCL_DETAIL_INSTANTIATE_KERNEL_HPP
00003
00004 /** \file Some helpers to share information between the SYCL program
00005 and the device compiler.
00006
00007 For example, this brings a function to outline a SYCL kernel at a
00008 specific place.
00009
00010 Ronan at Keryell point FR
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include <tuple>
00017
00018 #include "triSYCL/command_group/detail/task.hpp"
00019
00020 namespace trisycl::detail {
00021
00022 /** Avoid the interprocedural optimization to remove these arguments
00023 in the kernel instantiation by relying on some dummy static variables.
00024
00025 Using \c [[gnu::used]] does not work on arguments but only
00026 on static variable, so use this function.
00027 */
00028 auto inline prevent_arguments_from_optimization = [] (auto & ...args) {
00029   /* Just keep track of the address of all the given objects,
00030 otherwise the objects are copied, may throw, are registered for
00031 destruction with \c atexit(), etc.  */
00032   [[gnu::used]] static auto keep = std::make_tuple(& args...);
00033 };
00034
00035
00036 /** Instantiate the kernel code
00037
00038 To have a clear view of what a kernel is at the LLVM IR level even
00039 when a lot of optimizations are done (-O3), call the kernel from
00040 this function marked as \c noinline, otherwise the kernel code is
00041 inlined directly where it is used.
00042
00043 Actually some optimizations/code restructuring might duplicate the
00044 call to \c instantiate_kernel but the \c noduplicate attribute
00045 seems incompatible with \c noinline in Clang/LLVM 3.9.1, because
00046 the inlining happen when both attributes are used...
00047 */
```

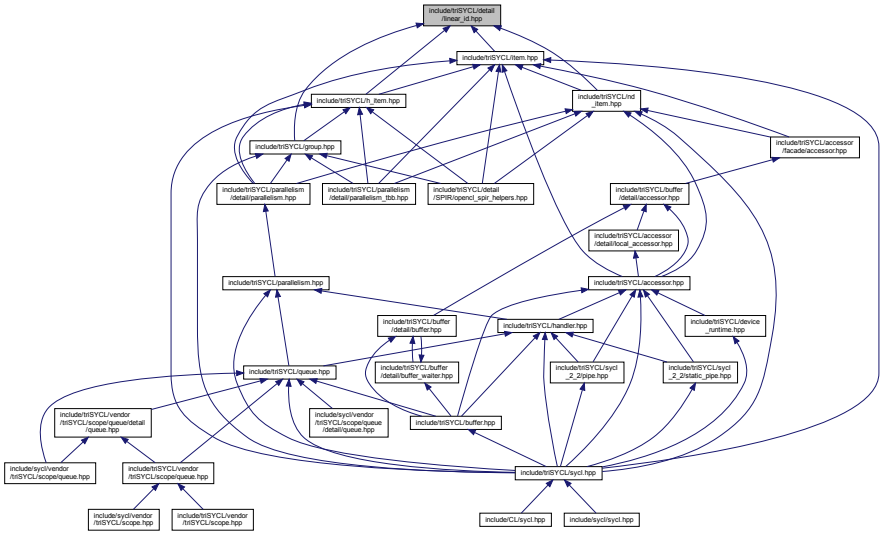
```
00048 template <typename KernelName,
00049           typename Functor>
00050 [[gnu::noinline]] void
00051 instantiate_kernel(Functor f) noexcept {
00052   /* The outlining compiler is expected to do some massage here or
00053 around and to insert some calls to \c serialize_arg and so on */
00054   f();
00055 }
00056
00057
00058 /** Set the task to be used to launch the kernel
00059
00060 This is actually a dummy function so that the device compiler can
00061 just grab the task for passing it to the argument serialization
00062 functions from the device runtime.
00063
00064 It is better to use a weak linkage than having this function with
00065 extern linkage:
00066
00067 - if the device compiler is used the function is kept by the
00068 compiler because full link and LTO is not applied before the
00069 device compiler passes are applied.  So there is no partial
00070 evaluation in the compiler according to what is returned by this
00071 function;
00072
00073 - if the device compiler is not used, the code still links and can
00074 be compiled.
00075 */
00076 TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX
00077 set_kernel_task_marker(detail::task &) noexcept {
00078 }
00079
00080
00081 /** Gather in a single function what is necessary for the device
00082 compiler to outline the kernel or to set up the kernel and
00083 serialize the arguments.
00084
00085 The \c noinline attribute is used to force this to appear as a
00086 real function the device compiler can massage.
00087 */
00088 template <typename KernelName,
00089           typename Kernel>
00090 [[gnu::noinline]]
00091 void launch_device_kernel(detail::task &t,
00092                           Kernel k) noexcept {
00093 /** Setup the task to be used to launch the kernel.
00094
00095 This is used by the outlining compiler.  */
00096   detail::set_kernel_task_marker(t);
00097   /* A simplified version for the device just to be able to extract
00098 the kernel itself.
00099
00100 It is not a real outlining but a good approximation to be
00101 massaged by a compiler later.
00102
00103 Put the iteration space loops in the kernels for now, which makes
00104 sense for CPU emulation or FPGA pipelined execution.
00105 */
00106   instantiate_kernel<KernelName>(k);
00107 }
00108
00109 }
00110
00111 /*
00112 # Some Emacs stuff:
00113 ### Local Variables:
00114 ### ispell-local-dictionary:  "american"
00115 ### eval:  (flyspell-prog-mode)
00116 ### End:
00117 */
00118
00119 #endif // TRISYCL_SYCL_DETAIL_INSTANTIATE_KERNEL_HPP
```

13.67 include/triSYCL/detail/linear_id.hpp File Reference

`#include <cstddef>`
Include dependency graph for linear_id.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Functions

- template<typename Range , typename Id >
  size_t constexpr trisycl::detail::linear_id (Range range, Id id, Id offset={})

  Compute a linearized array access used in the OpenCL 2 world.

13.68 linear_id.hpp

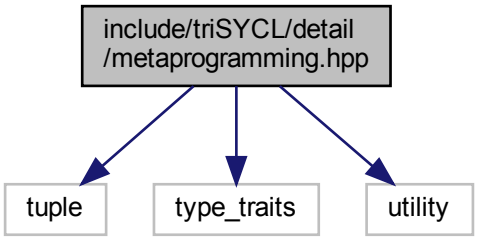
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_LINEAR_ID_HPP
00002 #define TRISYCL_SYCL_DETAIL_LINEAR_ID_HPP
00003
00004 /** \file Compute linearized array access
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <cstddef>
00013
00014 namespace trisycl::detail {
00015
00016 /** \addtogroup helpers Some helpers for the implementation
00017 @{
00018 */
00019
00020 /** Compute a linearized array access used in the OpenCL 2 world
00021
00022 Typically for the get_global_linear_id() and get_local_linear_id()
00023 functions.
00024 */
00025 template <typename Range, typename Id>
00026 size_t constexpr inline linear_id(Range range, Id id, Id offset = {}) {
00027   auto dims = std::distance(std::begin(range), std::end(range));
00028
00029   size_t linear_id = 0;
00030   /* A good compiler should unroll this and do partial evaluation to
00031 remove the first multiplication by 0 of this Horner evaluation and
00032 remove the 0 offset evaluation */
00033   for (int i = dims - 1; i >= 0; --i)
00034     linear_id = linear_id*range[i] + id[i] - offset[i];
00035
00036   return linear_id;
00037 }
00038
00039
00040 /// @} End the helpers Doxygen group
00041
00042 }
00043
00044 /*
00045 # Some Emacs stuff:
00046 ### Local Variables:
00047 ### ispell-local-dictionary:  "american"
00048 ### eval:  (flyspell-prog-mode)
00049 ### End:
00050 */
00051
00052 #endif // TRISYCL_SYCL_DETAIL_LINEAR_ID_HPP
```

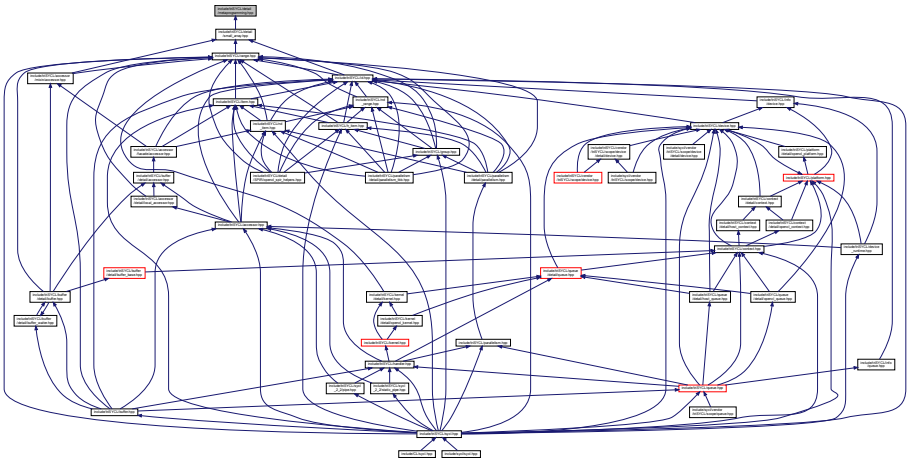
13.69 include/triSYCL/detail/metaprogramming.hpp File Reference

```
#include <tuple>
#include <type_traits>
#include <utility>
```

Include dependency graph for metaprogramming.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Functions

- auto & trisycl::detail::assign_from_tuple (auto &dest, auto &&tuple)

  Assign arguments from a tuple.

13.70 metaprogramming.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_METAPROGRAMMING_HPP
00002 #define TRISYCL_SYCL_DETAIL_METAPROGRAMMING_HPP
00003
00004 /** \file Some metaprogramming utilities
00005 */
00006
00007 #include <tuple>
00008 #include <type_traits>
00009 #include <utility>
00010
00011 namespace trisycl::detail {
00012
00013 /// Assign arguments from a tuple
00014 auto& assign_from_tuple(auto& dest, auto&& tuple) {
00015   auto expanded_assign = [&](auto&&...  elems) mutable {
00016     // Assign from each tuple element expanded
00017     dest = { std::forward<decltype(elems)>(elems)...  };
00018   };
00019   std::apply(expanded_assign, std::forward<decltype(tuple)>(tuple));
00020   // So we can use the result of the assignment expression if needed
00021   return dest;
00022 }
00023
00024 }
00025
00026 #endif // TRISYCL_SYCL_DETAIL_METAPROGRAMMING_HPP
```

13.71 include/triSYCL/detail/overloaded.hpp File Reference

Classes

- struct trisycl::detail::overloaded< Ts >

  Helper type for the visitor to do pattern matching on invocables https://en.cppreference.com/w/cpp/utility/variant/visit. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Functions

- template<class... Ts>
  trisycl::detail::overloaded (Ts...) -> overloaded< Ts... >

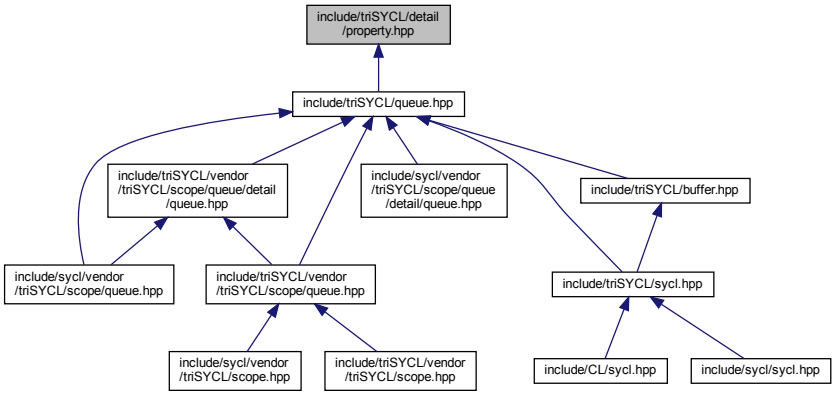
13.72 overloaded.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_OVERLOADED_HPP
00002 #define TRISYCL_SYCL_DETAIL_OVERLOADED_HPP
00003
00004 /** \file Helper type for the std::visit to do pattern matching on invocables
00005
00006 This file is distributed under the University of Illinois Open Source
00007 License.  See LICENSE.TXT for details.
00008 */
00009
00010 namespace trisycl::detail {
00011
00012 /** \addtogroup helpers Some helpers for the implementation
00013 @{
00014 */
00015
00016 /// Helper type for the visitor to do pattern matching on invocables
00017 /// https://en.cppreference.com/w/cpp/utility/variant/visit
00018 template<class...  Ts> struct overloaded :  Ts...  { using Ts::operator()...; };
00019
00020 // Explicit deduction guide (not needed as of C++20)
00021 template<class...  Ts> overloaded(Ts...)  -> overloaded<Ts...>;
00022
00023
00024 /// @} End the helpers Doxygen group
00025
00026 }
00027
00028
00029 #endif // TRISYCL_SYCL_DETAIL_OVERLOADED_HPP
```

13.73 include/triSYCL/detail/property.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::property

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.74 property.hpp

Go to the documentation of this file.

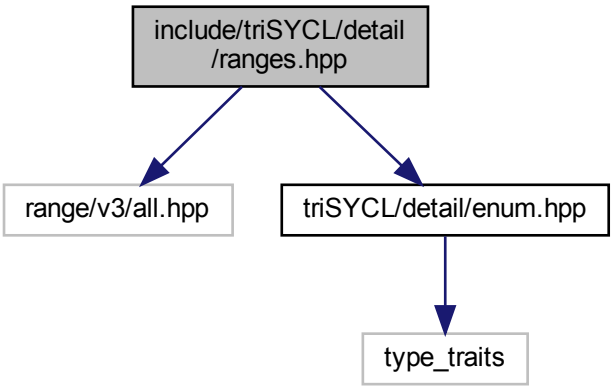
```
00001 #ifndef TRISYCL_SYCL_DETAIL_PROPERTY_HPP
00002 #define TRISYCL_SYCL_DETAIL_PROPERTY_HPP
00003
00004 namespace trisycl::detail {
00005
00006 class property {
00007 };
00008
00009 }
00010 #endif
```

13.75 include/triSYCL/detail/ranges.hpp File Reference

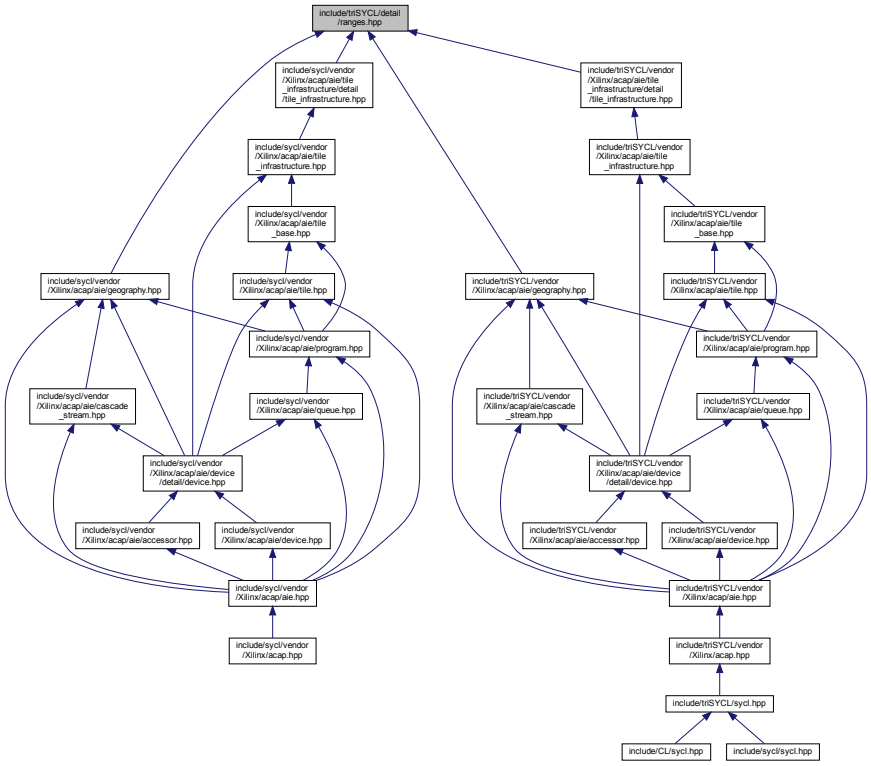
Some range-related helpers.

```
#include <range/v3/all.hpp>
#include "triSYCL/detail/enum.hpp"
```

Include dependency graph for ranges.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::views

Variables

- auto constexpr trisycl::views::enum_underlying_type

  Construct a range of integral values iterating between 2 enum values, inclusive.
- auto constexpr trisycl::views::enum_type

  Construct a range iterating between 2 enum values, inclusive.

13.75.1 Detailed Description

Some range-related helpers.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file ranges.hpp.

13.76 ranges.hpp

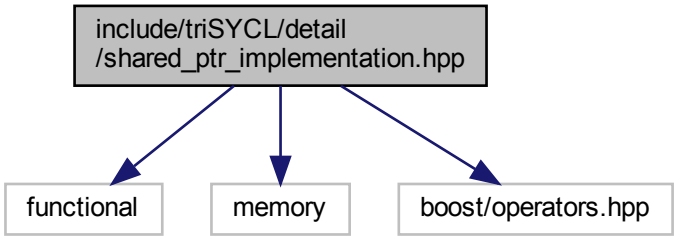
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_RANGES_HPP
00002 #define TRISYCL_SYCL_DETAIL_RANGES_HPP
00003
00004 /** \file
00005
00006 Some range-related helpers
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include <range/v3/all.hpp>
00015
00016 #include "triSYCL/detail/enum.hpp"
00017
00018 namespace trisycl::views {
00019
00020 /** Construct a range of integral values iterating between 2 enum
00021 values, inclusive
00022
00023 \param begin is the starting value
00024
00025 \param inclusive_end is the value to stop at.
00026
00027 Note that since \p inclusive_end is included in the range, the
00028 range has at least 1 element.
00029
00030 \return a range of integral values of the underlying type of the
00031 enum from \p begin to the \p inclusive_end
00032 */
00033 auto constexpr enum_underlying_type = [] (auto begin, auto inclusive_end) {
00034   using begin_t = decltype(begin);
00035   using inclusive_end_t = decltype(inclusive_end);
00036
00037   static_assert(std::is_same_v<begin_t, inclusive_end_t>,
00038                 "begin and inclusive_end should be from the same enum");
00039   return ranges::iota_view { detail::underlying_value(begin),
00040                              detail::underlying_value(inclusive_end) + 1 };
00041 };
00042
00043
00044 /** Construct a range iterating between 2 enum values, inclusive
00045
00046 \param begin is the starting value
00047
00048 \param inclusive_end is the value to stop at.
00049
00050 Note that since \p inclusive_end is included in the range, the
00051 range has at least 1 element.
00052
00053 \return a range of enum values from \p begin to the \p
00054 inclusive_end
00055 */
00056 auto constexpr enum_type = [] (auto begin, auto inclusive_end) {
00057   using begin_t = decltype(begin);
00058
00059   return enum_underlying_type(begin, inclusive_end)
00060     | ranges::views::transform([] (auto i) { return static_cast<begin_t>(i); });
00061 };
00062
00063 }
00064
00065 /*
00066 # Some Emacs stuff:
00067 ### Local Variables:
00068 ### ispell-local-dictionary:  "american"
00069 ### eval:  (flyspell-prog-mode)
00070 ### End:
00071 */
00072
00073 #endif // TRISYCL_SYCL_DETAIL_RANGES_HPP
```

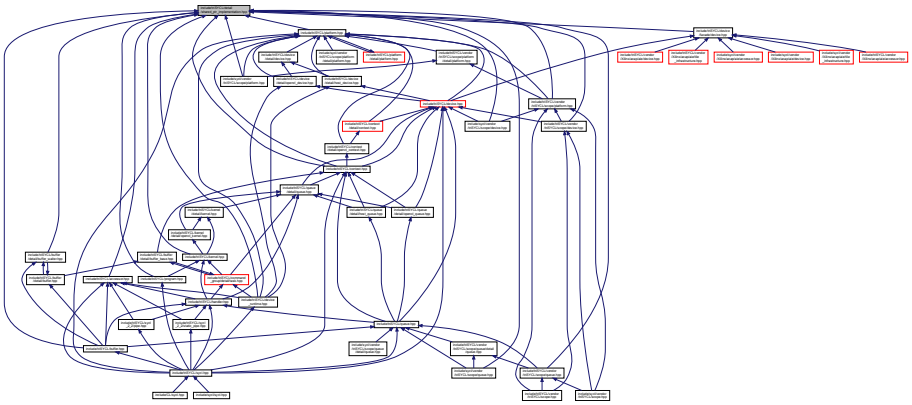
13.77 include/triSYCL/detail/shared_ptr_implementation.hpp File Reference

```
#include <functional>
#include <memory>
#include <boost/operators.hpp>
```

Include dependency graph for shared_ptr_implementation.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::detail::shared_ptr_implementation< Parent, Implementation >

  Provide an implementation as shared_ptr with total ordering and hashing to be used with algorithms and in (un)ordered containers.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.78 shared_ptr_implementation.hpp

Go to the documentation of this file.

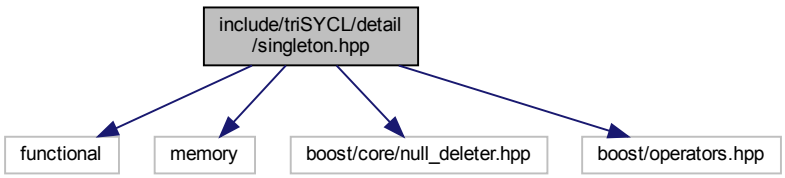
```
00001 #ifndef TRISYCL_SYCL_DETAIL_SHARED_PTR_IMPLEMENTATION_HPP
00002 #define TRISYCL_SYCL_DETAIL_SHARED_PTR_IMPLEMENTATION_HPP
00003
00004 /** \file Mix-in to add an implementation as shared_ptr with total
00005 ordering and hashing so that the class can be used with algorithms
00006 and in (un)ordered containers
00007
00008 Ronan at Keryell point FR
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include <functional>
00015 #include <memory>
00016
00017 #include <boost/operators.hpp>
00018
00019 namespace trisycl::detail {
00020
00021 /** Provide an implementation as shared_ptr with total ordering and
00022 hashing to be used with algorithms and in (un)ordered containers
00023
00024 To be used, a Parent class wanting an Implementation needs to
00025 inherit from.
00026
00027 The implementation ends up in a member really named
00028 "implementation".
00029
00030 \code
00031 public detail::shared_ptr_implementation<Parent, Implementation>
00032 \endcode
00033
00034 and also inject in std namespace a specialization for
00035 \code hash<Parent> \endcode
00036 */
00037 template <typename Parent, typename Implementation>
00038 struct shared_ptr_implementation :  public boost::totally_ordered<Parent> {
00039
00040 /// The implementation forward everything to this...  implementation
00041   std::shared_ptr<Implementation> implementation;
00042
00043 /// The implementation directly as a shared pointer
00044   shared_ptr_implementation(std::shared_ptr<Implementation> i)
00045     :  implementation { i } {}
00046
00047
00048 /// The implementation takes the ownership from a raw pointer
00049   shared_ptr_implementation(Implementation *i) :  implementation { i } {}
00050
00051
00052 /// Keep all other constructors to have usual shared_ptr behaviour
00053   shared_ptr_implementation() = default;
00054
00055
00056 /** Equality operator
00057
00058 This is generalized by boost::equality_comparable from
00059 boost::totally_ordered to implement the equality comparable
00060 concept
00061
00062 Use recipe for hidden friends from P1601R0:  "Recommendations for
00063 Specifying "Hidden Friends""
00064 */
00065   friend bool operator ==(const Parent& lhs, const Parent& rhs) {
00066     return lhs.implementation == rhs.implementation;
00067   }
00068
00069
00070 /** Inferior operator
00071
00072 This is generalized by boost::less_than_comparable from
00073 boost::totally_ordered to implement the equality comparable
00074 concept
00075
00076 Use recipe for hidden friends from P1601R0:  "Recommendations for
00077 Specifying "Hidden Friends""
00078
00079 \todo Add this to the spec
00080 */
00081   friend bool operator <(const Parent& lhs, const Parent& rhs) {
00082     return lhs.implementation < rhs.implementation;
```

```
00083   }
00084
00085
00086 /// Forward the hashing for unordered containers to the implementation
00087   auto hash()const {
00088     return std::hash<decltype(implementation)>{}(implementation);
00089   }
00090
00091 };
00092
00093 }
00094
00095 /*
00096 # Some Emacs stuff:
00097 ### Local Variables:
00098 ### ispell-local-dictionary:  "american"
00099 ### eval:  (flyspell-prog-mode)
00100 ### End:
00101 */
00102
00103 #endif // TRISYCL_SYCL_DETAIL_SHARED_PTR_IMPLEMENTATION_HPP
```

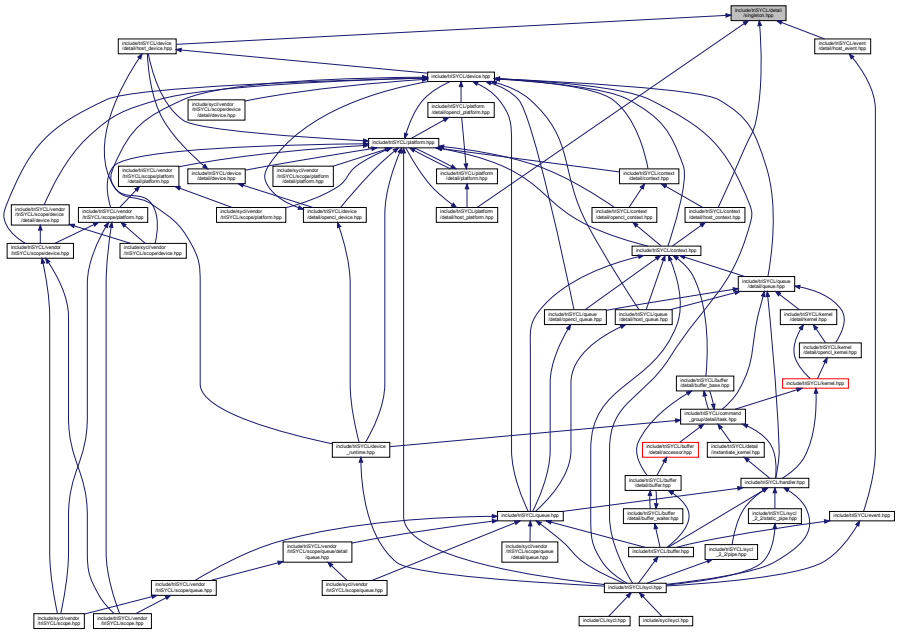
13.79 include/triSYCL/detail/singleton.hpp File Reference

```
#include <functional>
#include <memory>
#include <boost/core/null_deleter.hpp>
#include <boost/operators.hpp>
```

Include dependency graph for singleton.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::detail::singleton< T >

  Provide a singleton factory.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.80 singleton.hpp

Go to the documentation of this file.

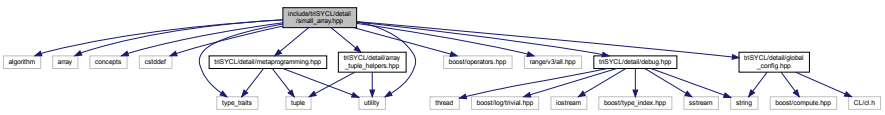
```
00001 #ifndef TRISYCL_SYCL_DETAIL_SINGLETON_HPP
00002 #define TRISYCL_SYCL_DETAIL_SINGLETON_HPP
00003
00004 /** \file Mix-in to add a singleton implementation with an instance() method
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <functional>
00013 #include <memory>
00014
00015 #include <boost/core/null_deleter.hpp>
00016 #include <boost/operators.hpp>
00017
00018
00019 namespace trisycl::detail {
```

```
00020
00021 /// Provide a singleton factory
00022 template <typename T>
00023 struct singleton {
00024
00025 /// Get a singleton instance of T
00026   static std::shared_ptr<T> instance() {
00027     // C++11 guaranties the static construction is thread-safe
00028     static T single;
00029 /** Use a null_deleter since the singleton should not be deleted,
00030 as allocated in the static area */
00031     static std::shared_ptr<T> sps { &single,
00032                                     boost::null_deleter {} };
00033
00034     return sps;
00035   }
00036
00037 };
00038
00039 }
00040
00041 /*
00042 # Some Emacs stuff:
00043 ### Local Variables:
00044 ### ispell-local-dictionary:  "american"
00045 ### eval:  (flyspell-prog-mode)
00046 ### End:
00047 */
00048
00049 #endif // TRISYCL_SYCL_DETAIL_SINGLETON_HPP
```

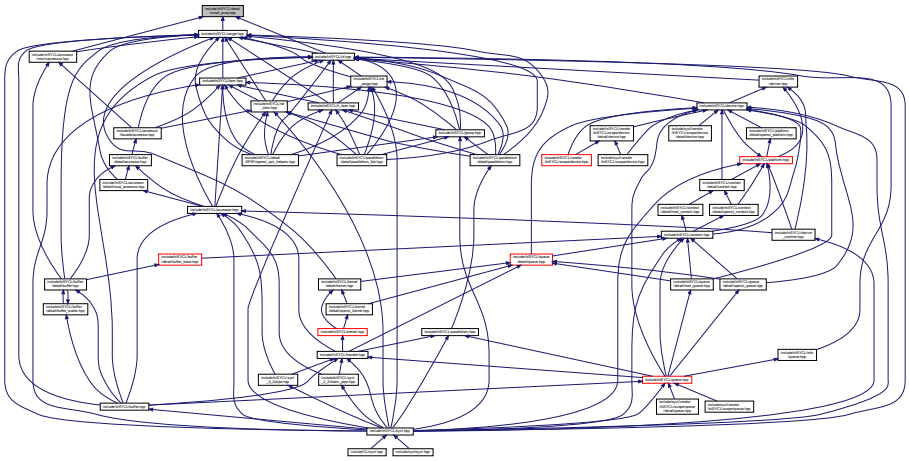
13.81 include/triSYCL/detail/small_array.hpp File Reference

```
#include <algorithm>
#include <array>
#include <concepts>
#include <cstddef>
#include <type_traits>
#include <utility>
#include <boost/operators.hpp>
#include <range/v3/all.hpp>
#include "triSYCL/detail/global_config.hpp"
#include "triSYCL/detail/array_tuple_helpers.hpp"
#include "triSYCL/detail/debug.hpp"
#include "triSYCL/detail/metaprogramming.hpp"
```

Include dependency graph for small_array.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::detail::small_array_tag

  Tag to define simply inheritance from small_array. More...
- struct trisycl::detail::small_array< BasicType, FinalType, Dims >

  Define a multi-dimensional index, used for example to locate a work item or a buffer element. More...
- struct trisycl::detail::small_array_sycl< BasicType, FinalType, Dims >

  A small array to implement SYCL objects like id, range, item... with 1, 2 or 3 elements (or even more if detail::allow_any_dimension is true), with the implicit constructors. More...
- struct std::tuple_size< T >

  Export the dimension as its tuple size.
- struct std::tuple_element< I, T >

  Each tuple element type is the same, the one of any small_array heir element.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.
- namespace std

Concepts

- concept trisycl::detail::is_small_array

  Concept describing a class inheriting from small_array.

Macros

- #define TRISYCL_BOOST_OPERATOR_VECTOR_OP(op)

  Helper macro to declare a vector operation with the given side-effect operator.
- #define TRISYCL_LOGICAL_OPERATOR_VECTOR_OP(op)

  Helper macro to declare a vector operation returning a new type containing the result of the operator.
- #define TRISYCL_UNARY_OPERATOR_VECTOR_OP(op)

  Helper macro to declare a vector unary operation.
- #define TRISYCL_UNARY_PREFIX_OPERATOR_VECTOR_OP(op)

  Helper macro to declare a vector prefix unary operation.
- #define TRISYCL_UNARY_POSTFIX_OPERATOR_VECTOR_OP(op)

  Helper macro to declare a vector prefix unary operation.
- #define TRISYCL_OPERATOR_BASIC_TYPE_OP(op)

  Helper macro to declare out-of-class operators for b vs a[] operations.

Functions

- template<std::size_t I, typename T >
  requires trisycl::detail::is_small_array<T>
  decltype(auto) std::get (T &&e)

  The element of the tuple is the matching element of the small_array heir.

13.82 small_array.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_SMALL_ARRAY_HPP
00002 #define TRISYCL_SYCL_DETAIL_SMALL_ARRAY_HPP
00003
00004 /** \file This is a small array class to build range<>, id<>, etc.
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <algorithm>
00013 #include <array>
00014 #include <concepts>
00015 #include <cstddef>
00016 #include <type_traits>
00017 #include <utility>
00018
00019 #include <boost/operators.hpp>
00020
00021 #include <range/v3/all.hpp>
00022
00023 #include "triSYCL/detail/global_config.hpp"
00024 #include "triSYCL/detail/array_tuple_helpers.hpp"
00025 #include "triSYCL/detail/debug.hpp"
00026 #include "triSYCL/detail/metaprogramming.hpp"
00027
00028
00029 namespace trisycl::detail {
00030
00031 /** \addtogroup helpers Some helpers for the implementation
00032 @{
00033 */
00034
00035 /// Tag to define simply inheritance from small_array
00036 struct small_array_tag {};
00037
00038 /// Concept describing a class inheriting from small_array
00039 template <typename T>
00040 concept is_small_array = std::derived_from<T, trisycl::detail::small_array_tag>;
00041
```

```
00042 /** Helper macro to declare a vector operation with the given side-effect
00043 operator.
00044
00045 This handles both a[i] op b[i] and a[i] op b, where b is a BasicType.
00046 */
00047 #define TRISYCL_BOOST_OPERATOR_VECTOR_OP(op)            \
00048 FinalType operator op(const FinalType &rhs) {         \
00049 for (std::size_t i = 0; i != Dims; ++i)             \
00050 (*this)[i] op rhs[i];                             \
00051 return *this;                                       \
00052 }                                                     \
00053 FinalType operator op(const BasicType &rhs) {         \
00054 for (std::size_t i = 0; i != Dims; ++i)             \
00055 (*this)[i] op rhs;                                \
00056 return *this;                                       \
00057 }                                                     \
00058
00059 /** Helper macro to declare a vector operation returning a new
00060 type containing the result of the operator.
00061
00062 This handles both a[] op b[] and a[] op b, where b is a BasicType.
00063 */
00064 #define TRISYCL_LOGICAL_OPERATOR_VECTOR_OP(op)                    \
00065 FinalType operator op(const FinalType &rhs) const {             \
00066 FinalType res;                                                \
00067 for (std::size_t i = 0; i != Dims; ++i)                       \
00068 res[i] = (*this)[i] op rhs[i];                              \
00069 return res;                                                   \
00070 }                                                             \
00071 /* Skip this for Dims = 1 to avoid ambiguity with implicit type \
00072 conversion between the vector type and its basic type */\
00073 template <typename FT = FinalType,                            \
00074 typename = std::enable_if_t<Dims != 1, FT>>           \
00075 FinalType operator op(const BasicType &rhs) {                 \
00076 FinalType res;                                                \
00077 for (std::size_t i = 0; i != Dims; ++i)                       \
00078 res[i] = (*this)[i] op rhs;                                 \
00079 return res;                                                   \
00080 }
00081
00082 /** Helper macro to declare a vector unary operation.
00083
00084 This handles op a.
00085 */
00086 #define TRISYCL_UNARY_OPERATOR_VECTOR_OP(op)     \
00087 FinalType operator op() const {                \
00088 FinalType result;                            \
00089 for (std::size_t i = 0; i != Dims; ++i)      \
00090 result[i] = op(*this)[i];                  \
00091 return result;                               \
00092 }
00093
00094 /** Helper macro to declare a vector prefix unary operation.
00095
00096 This handles ++a and --a.
00097 */
00098 #define TRISYCL_UNARY_PREFIX_OPERATOR_VECTOR_OP(op) \
00099 FinalType operator op() {                          \
00100 for (std::size_t i = 0; i != Dims; ++i)          \
00101 op(*this)[i];                                  \
00102 return *this;                                    \
00103 }
00104
00105 /** Helper macro to declare a vector prefix unary operation.
00106
00107 This handles a++ and a--.
00108 */
00109 #define TRISYCL_UNARY_POSTFIX_OPERATOR_VECTOR_OP(op)  \
00110 FinalType operator op(int) {                         \
00111 FinalType result = *this;                          \
00112 for (std::size_t i = 0; i != Dims; ++i)            \
00113 op(*this)[i];                                    \
00114 return result;                                     \
00115 }
00116
00117
00118 /** Define a multi-dimensional index, used for example to locate a work
00119 item or a buffer element
00120
00121 Unfortunately, even if std::array is an aggregate class allowing
00122 native list initialization, it is no longer an aggregate if we derive
00123 from an aggregate.  Thus we have to redeclare the constructors.
00124
00125 \param BasicType is the type element, such as int
00126
00127 \param Dims is the dimension number, typically between 1 and 3
00128
```

```
00129 \param FinalType is the final type, such as range<> or id<>, so that
00130 boost::operator can return the right type
00131
00132 std::array<> provides the collection concept, with .size(), == and !=
00133 too.
00134
00135 Can't use boost::shiftable, as it doesn't define the U, T overload
00136 outside the class, but when that is added, it gets ambiguity problems.
00137 */
00138 template <typename BasicType,
00139           typename FinalType,
00140           std::size_t Dims>
00141 struct small_array :  std::array<BasicType, Dims>,
00142   // To have all the usual arithmetic operations on this type and bitwise
00143   // operations.  Note these operation classes are chained via:
00144   // https://www.boost.org/doc/libs/1_69_0/libs/utility/operators.htm#chaining
00145   // without the chaining the empty base class optimization will break and the
00146   // class will bloat other types like vec.hpp
00147   boost::euclidean_ring_operators<FinalType,
00148   boost::euclidean_ring_operators<FinalType, BasicType,
00149   boost::bitwise<FinalType,
00150   boost::bitwise<FinalType, BasicType>>>,
00151   // Already provided by array<> lexicographically:
00152   // boost::equality_comparable<FinalType>,
00153   // boost::less_than_comparable<FinalType>,
00154   // Add a display() method
00155   detail::display_vector<FinalType>,
00156   // Just to find easily this is a small_array without templates
00157   detail::small_array_tag {
00158   /* Note that constexpr size() from the underlying std::array provides
00159 the same functionality */
00160   static constexpr size_t dimension = Dims;
00161
00162   using element_type = BasicType;
00163
00164 /** A constructor from another array
00165
00166 Make it explicit to avoid spurious range<> constructions from int *
00167 for example
00168 */
00169   template <typename SourceType>
00170   small_array(const SourceType src[Dims]) {
00171     // (*this)[0] is the first element of the underlying array
00172     std::copy_n(src, Dims, &(*this)[0]);
00173   }
00174
00175
00176 /// A constructor from another small_array of the same size
00177   template <typename SourceBasicType,
00178             typename SourceFinalType>
00179   small_array(const small_array<SourceBasicType,
00180               SourceFinalType,
00181               Dims> &src) {
00182     std::copy_n(&src[0], Dims, &(*this)[0]);
00183   }
00184
00185 /** Initialize the array from a list of exactly Dims elements
00186
00187 Strangely, even when using the array constructors, the
00188 initialization of the aggregate is not available.  So recreate an
00189 equivalent here.
00190 */
00191   template <typename...  Types>
00192     requires(sizeof...(Types) == Dims)
00193   small_array(const Types&...  args)
00194       :  std::array<BasicType, Dims> {
00195         // Allow a loss of precision in initialization with the static_cast
00196         { static_cast<BasicType>(args)...  }
00197       } {}
00198
00199 /// Construct a small_array from a std::array
00200   template <typename SourceBasicType>
00201   small_array(const std::array<SourceBasicType, Dims> &src)
00202   :  std::array<BasicType, Dims>(src) {}
00203
00204
00205 /// Keep other constructors from the underlying std::array
00206   using std::array<BasicType, Dims>::array;
00207
00208 /// Keep the synthesized constructors
00209   small_array() = default;
00210
00211 /// Return the element of the array
00212   auto get(std::size_t index)const {
00213     return (*this)[index];
00214   }
00215
```

```
00216   /* Implement minimal methods boost::euclidean_ring_operators needs to
00217 generate everything */
00218 /// Add + like operations on the id<> and others
00219   TRISYCL_BOOST_OPERATOR_VECTOR_OP(+=)
00220
00221   /// Add - like operations on the id<> and others
00222   TRISYCL_BOOST_OPERATOR_VECTOR_OP(-=)
00223
00224 /// Add * like operations on the id<> and others
00225   TRISYCL_BOOST_OPERATOR_VECTOR_OP(*=)
00226
00227 /// Add / like operations on the id<> and others
00228   TRISYCL_BOOST_OPERATOR_VECTOR_OP(/=)
00229
00230 /// Add % like operations on the id<> and others
00231   TRISYCL_BOOST_OPERATOR_VECTOR_OP(%=)
00232
00233 /// Add « like operations on the id<> and others
00234   TRISYCL_BOOST_OPERATOR_VECTOR_OP(«=)
00235
00236 /// Add » like operations on the id<> and others
00237   TRISYCL_BOOST_OPERATOR_VECTOR_OP(»=)
00238
00239 /// Add & like operations on the id<> and others
00240   TRISYCL_BOOST_OPERATOR_VECTOR_OP(&=)
00241
00242 /// Add ^ like operations on the id<> and others
00243   TRISYCL_BOOST_OPERATOR_VECTOR_OP(^=)
00244
00245 /// Add | like operations on the id<> and others
00246   TRISYCL_BOOST_OPERATOR_VECTOR_OP(|=)
00247
00248 #undef TRISYCL_BOOST_OPERATOR_VECTOR_OP
00249
00250 /// Add && operations on the id<> and others
00251   TRISYCL_LOGICAL_OPERATOR_VECTOR_OP(&&)
00252
00253   /// Add || operations on the id<> and others
00254   TRISYCL_LOGICAL_OPERATOR_VECTOR_OP(||)
00255
00256 /// Add comparison operations on the id<> and others
00257   TRISYCL_LOGICAL_OPERATOR_VECTOR_OP(<)
00258   TRISYCL_LOGICAL_OPERATOR_VECTOR_OP(>)
00259   TRISYCL_LOGICAL_OPERATOR_VECTOR_OP(<=)
00260   TRISYCL_LOGICAL_OPERATOR_VECTOR_OP(>=)
00261
00262 /// Add shiftable operators on the vector op
00263   TRISYCL_LOGICAL_OPERATOR_VECTOR_OP(«)
00264   TRISYCL_LOGICAL_OPERATOR_VECTOR_OP(»)
00265
00266 #undef TRISYCL_LOGICAL_OPERATOR_VECTOR_OP
00267
00268 /// Implement unary operations
00269   TRISYCL_UNARY_OPERATOR_VECTOR_OP(-)
00270   TRISYCL_UNARY_OPERATOR_VECTOR_OP(+)
00271
00272 #undef TRISYCL_UNARY_OPERATOR_VECTOR_OP
00273
00274 /// Implement prefix unary operations
00275   TRISYCL_UNARY_PREFIX_OPERATOR_VECTOR_OP(--)
00276   TRISYCL_UNARY_PREFIX_OPERATOR_VECTOR_OP(++)
00277
00278 #undef TRISYCL_UNARY_PREFIX_OPERATOR_VECTOR_OP
00279
00280 /// Implement postfix unary operations
00281   TRISYCL_UNARY_POSTFIX_OPERATOR_VECTOR_OP(--)
00282   TRISYCL_UNARY_POSTFIX_OPERATOR_VECTOR_OP(++)
00283
00284 #undef TRISYCL_UNARY_POSTFIX_OPERATOR_VECTOR_OP
00285
00286 /** Since the boost::operator work on the small_array, add an implicit
00287 conversion to produce the expected type */
00288   operator FinalType() {
00289     return static_cast<FinalType&>(*this);
00290   }
00291 };
00292
00293 /** Helper macro to declare out-of-class operators for b vs a[]
00294 operations.
00295 */
00296 #define TRISYCL_OPERATOR_BASIC_TYPE_OP(op)                              \
00297 template <typename BasicType, typename FinalType, std::size_t Dims>   \
00298 struct small_array<BasicType, FinalType, Dims> operator op(           \
00299 const BasicType &lhs,                                                \
00300 const struct small_array<BasicType, FinalType, Dims> &rhs)           \
00301 {                                                                      \
00302 FinalType res;                                                       \
```

```
00303 for (std::size_t i = 0; i != Dims; ++i)                                \
00304 res[i] = lhs op rhs[i];                                               \
00305 return res;                                                           \
00306 }
00307
00308 /* Comparison operators without Boost as Boost only specs bool return types,
00309 and the SYCL spec wants an arrayed of values returned.  */
00310 TRISYCL_OPERATOR_BASIC_TYPE_OP(>)
00311 TRISYCL_OPERATOR_BASIC_TYPE_OP(<)
00312 TRISYCL_OPERATOR_BASIC_TYPE_OP(<=)
00313 TRISYCL_OPERATOR_BASIC_TYPE_OP(>=)
00314 /* Shiftable operators without Boost as Boost doesn't specify
00315 the out-of-class U, T constructors */
00316 TRISYCL_OPERATOR_BASIC_TYPE_OP(«)
00317 TRISYCL_OPERATOR_BASIC_TYPE_OP(»)
00318
00319 #undef TRISYCL_OPERATOR_BASIC_TYPE_OP
00320
00321 /** A small array to implement SYCL objects like id, range,
00322 item...  with 1, 2 or 3 elements (or even more if
00323 detail::allow_any_dimension is true), with the implicit
00324 constructors */
00325 template <typename BasicType, typename FinalType, std::size_t Dims>
00326 struct small_array_sycl :  small_array<BasicType, FinalType, Dims> {
00327   static_assert(1 <= Dims, "SYCL dimensions are greater than or equal to 1");
00328   static_assert(detail::allow_any_dimension || Dims <= 3,
00329                 "SYCL dimensions are less than or equal to 3");
00330
00331 /// Keep other constructors
00332   small_array_sycl() = default;
00333
00334 /** Construct from a range-like object inheriting from a
00335 compatible std::array
00336
00337 This useful in implementation details for example to construct a
00338 sycl::id from a sycl::range.
00339 */
00340   template <typename Range>
00341   requires requires {
00342     ranges::copy(std::declval<Range>(), std::declval<BasicType*>());
00343   }
00344   small_array_sycl(Range&& r) {
00345     // Copy to internal value
00346     ranges::copy(std::forward<Range>(r), this->begin());
00347   }
00348
00349 /** Constructor taking either Dims arguments compatible with BasicType
00350 or just 1 argument compatible with BasicType and broadcast its
00351 value to all the elements */
00352   template <typename...  Args>
00353   // Check that each parameter is convertible to the BasicType
00354   // Full C++20:  requires (std::convertible_to<Args, BasicType> && ...)
00355   // In the meantime:
00356   requires requires { (static_cast<BasicType>(std::declval<Args>()), ...); }
00357   small_array_sycl(Args...  args) {
00358     if constexpr (sizeof...(Args) == Dims) {
00359       // If there is the right number of elements we initialize each
00360       // element from each parameter, allowing some narrowing
00361       static_cast<std::array<BasicType, Dims>&>(
00362           *this) = { static_cast<BasicType>(args)...  };
00363     } else if constexpr (sizeof...(Args) == 1) {
00364       // If there is only 1 argument, this is a broadcast
00365       detail::assign_from_tuple(
00366           static_cast<std::array<BasicType, Dims>&>(*this),
00367           detail::expand_to_vector<small_array_sycl, std::tuple<BasicType>,
00368                                    true> {}
00369               .expand(args...));
00370     } else
00371       static_assert(true, "There should be either 1 argument to broadcast the "
00372                           "value or 1 value per dimension");
00373   }
00374
00375 /** Conversion operator to scalar BasicType when only 1 dimension
00376
00377 This is used for example to implicitly convert a sycl::id<1> to
00378 a std::size_t
00379 */
00380   template <auto T = Dims>
00381     requires(Dims == 1)
00382   operator BasicType() {
00383     return this->get(0);
00384   }
00385 };
00386
00387 /// @} End the helpers Doxygen group
00388
00389 }
```

```
00390
00391 namespace std {
00392 // Declare a tuple-like interface for the heirs of the small_array
00393
00394 /// Export the dimension as its tuple size
00395 template <typename T>
00396   requires trisycl::detail::is_small_array<T>
00397 struct tuple_size<T>
00398     :  public std::integral_constant<std::size_t, T::dimension> {};
00399
00400 /// The element of the tuple is the matching element of the small_array heir
00401 template <std::size_t I, typename T>
00402   requires trisycl::detail::is_small_array<T> decltype(auto)
00403 get(T&& e) {
00404   return e.get(I);
00405 }
00406
00407 /// Each tuple element type is the same, the one of any small_array
00408 /// heir element
00409 template <std::size_t I, typename T>
00410   requires trisycl::detail::is_small_array<T>
00411 struct tuple_element<I, T> {
00412   using type = typename T::element_type;
00413 };
00414
00415 } // namespace std
00416
00417 /*
00418 # Some Emacs stuff:
00419 ### Local Variables:
00420 ### ispell-local-dictionary:  "american"
00421 ### eval:  (flyspell-prog-mode)
00422 ### End:
00423 */
00424
00425 #endif // TRISYCL_SYCL_DETAIL_SMALL_ARRAY_HPP
```

13.83 include/triSYCL/detail/SPIR/opencl_spir.h File Reference

Macros

- #define const_func __attribute__((const))

  Added according to pocl/examples/example1-spir64/generate_spir.sh.
- #define readonly __attribute__((pure))
- #define MAXFLOAT 0x1.fffffep127f

  Value of maximum non-infinite single-precision floating-point number.
- #define HUGE_VALF (__builtin_huge_valf())

  A positive float constant expression.
- #define HUGE_VAL (__builtin_huge_val())

  A positive double constant expression.
- #define INFINITY (__builtin_inff())

  A constant expression of type float representing positive or unsigned infinity.
- #define NAN as_float(__builtin_nanf((char const *)""))

  A constant expression of type float representing a quiet NaN.
- #define FP_ILOGB0 INT_MIN
- #define FP_ILOGBNAN INT_MAX
- #define FLT_DIG 6
- #define FLT_MANT_DIG 24
- #define FLT_MAX_10_EXP +38
- #define FLT_MAX_EXP +128
- #define FLT_MIN_10_EXP -37
- #define FLT_MIN_EXP -125
- #define FLT_RADIX 2
- #define FLT_MAX 0x1.fffffep127f
- #define FLT_MIN 0x1.0p-126f

- #define FLT_EPSILON 0x1.0p-23f
- #define DBL_DIG 15
- #define DBL_MANT_DIG 53
- #define DBL_MAX_10_EXP +308
- #define DBL_MAX_EXP +1024
- #define DBL_MIN_10_EXP -307
- #define DBL_MIN_EXP -1021
- #define DBL_RADIX 2
- #define DBL_MAX 0x1.fffffffffffffp1023
- #define DBL_MIN 0x1.0p-1022
- #define DBL_EPSILON 0x1.0p-52
- #define M_E_F 2.71828182845904523536028747135266250f
- #define M_LOG2E_F 1.44269504088896340735992468100189214f
- #define M_LOG10E_F 0.43429448190325182765112891891660508 2f
- #define M_LN2_F 0.69314718055994530941723212145817656 8f
- #define M_LN10_F 2.30258509299404568401799145468436421f
- #define M_PI_F 3.14159265358979323846264338327950288f
- #define M_PI_2_F 1.57079632679489661923132169163975144f
- #define M_PI_4_F 0.78539816339744830961566084581987572 1f
- #define M_1_PI_F 0.31830988618379067153776752674502872 4f
- #define M_2_PI_F 0.63661977236758134307553505349005744 8f
- #define M_2_SQRTPI_F 1.12837916709551257389615890312154517f
- #define M_SQRT2_F 1.41421356237309504880168872420969808f
- #define M_SQRT1_2_F 0.70710678118654752440084436210484903 9f
- #define M_E 0x1.5bf0a8b145769p+1
- #define M_LOG2E 0x1.71547652b82fep+0
- #define M_LOG10E 0x1.bcb7b1526e50ep-2
- #define M_LN2 0x1.62e42fefa39efp-1
- #define M_LN10 0x1.26bb1bbb55516p+1
- #define M_PI 0x1.921fb54442d18p+1
- #define M_PI_2 0x1.921fb54442d18p+0
- #define M_PI_4 0x1.921fb54442d18p-1
- #define M_1_PI 0x1.45f306dc9c883p-2
- #define M_2_PI 0x1.45f306dc9c883p-1
- #define M_2_SQRTPI 0x1.20dd750429b6dp+0
- #define M_SQRT2 0x1.6a09e667f3bcdp+0
- #define M_SQRT1_2 0x1.6a09e667f3bcdp-1
- #define CHAR_BIT 8
- #define SCHAR_MAX 127
- #define SCHAR_MIN (-128)
- #define UCHAR_MAX 255
- #define CHAR_MAX SCHAR_MAX
- #define CHAR_MIN SCHAR_MIN
- #define USHRT_MAX 65535
- #define SHRT_MAX 32767
- #define SHRT_MIN (-32768)
- #define UINT_MAX 0xffffffff
- #define INT_MAX 2147483647
- #define INT_MIN (-2147483647-1)
- #define ULONG_MAX 0xffffffffffffffffUL
- #define LONG_MAX ((long)0x7fffffffffffffffL)
- #define LONG_MIN ((long)(-0x7fffffffffffffffL-1))
- #define DBL_DIG 15
- #define DBL_MANT_DIG 53
- #define DBL_MAX_10_EXP +308

- #define DBL_MAX_EXP +1024
- #define DBL_MIN_10_EXP -307
- #define DBL_MIN_EXP -1021
- #define DBL_MAX 0x1.fffffffffffffp1023
- #define DBL_MIN 0x1.0p-1022
- #define DBL_EPSILON 0x1.0p-52
- #define CLK_LOCAL_MEM_FENCE 0x1

  Queue a memory fence to ensure correct ordering of memory operations to local memory.
- #define CLK_GLOBAL_MEM_FENCE 0x2

  Queue a memory fence to ensure correct ordering of memory operations to global memory.
- #define __kernel_exec(X, typen)
- #define kernel_exec(X, typen)
- #define CLK_ADDRESS_NONE 0

  Use the coordinate (x, y) to do an element lookup in the 2D image object specified by image.
- #define CLK_ADDRESS_CLAMP_TO_EDGE 2
- #define CLK_ADDRESS_CLAMP 4
- #define CLK_ADDRESS_REPEAT 6
- #define CLK_ADDRESS_MIRRORED_REPEAT 8
- #define CLK_NORMALIZED_COORDS_FALSE 0
- #define CLK_NORMALIZED_COORDS_TRUE 1
- #define CLK_FILTER_NEAREST 0x10
- #define CLK_FILTER_LINEAR 0x20
- #define CLK_R 0x10B0

  Return the channel data type.
- #define CLK_A 0x10B1
- #define CLK_RG 0x10B2
- #define CLK_RA 0x10B3
- #define CLK_RGB 0x10B4
- #define CLK_RGBA 0x10B5
- #define CLK_BGRA 0x10B6
- #define CLK_ARGB 0x10B7
- #define CLK_INTENSITY 0x10B8
- #define CLK_LUMINANCE 0x10B9
- #define CLK_Rx 0x10BA
- #define CLK_RGx 0x10BB
- #define CLK_RGBx 0x10BC
- #define CLK_DEPTH 0x10BD
- #define CLK_DEPTH_STENCIL 0x10BE
- #define CLK_SNORM_INT8 0x10D0
- #define CLK_SNORM_INT16 0x10D1
- #define CLK_UNORM_INT8 0x10D2
- #define CLK_UNORM_INT16 0x10D3
- #define CLK_UNORM_SHORT_565 0x10D4
- #define CLK_UNORM_SHORT_555 0x10D5
- #define CLK_UNORM_INT_101010 0x10D6
- #define CLK_SIGNED_INT8 0x10D7
- #define CLK_SIGNED_INT16 0x10D8
- #define CLK_SIGNED_INT32 0x10D9
- #define CLK_UNSIGNED_INT8 0x10DA
- #define CLK_UNSIGNED_INT16 0x10DB
- #define CLK_UNSIGNED_INT32 0x10DC
- #define CLK_HALF_FLOAT 0x10DD
- #define CLK_FLOAT 0x10DE
- #define CLK_UNORM_INT24 0x10DF

- #define as_char(x) __builtin_astype((x), char)

  OpenCL as_typen operators Reinterprets a data type as another data type of the same size.
- #define as_char2(x) __builtin_astype((x), char2)
- #define as_char3(x) __builtin_astype((x), char3)
- #define as_char4(x) __builtin_astype((x), char4)
- #define as_char8(x) __builtin_astype((x), char8)
- #define as_char16(x) __builtin_astype((x), char16)
- #define as_uchar(x) __builtin_astype((x), uchar)
- #define as_uchar2(x) __builtin_astype((x), uchar2)
- #define as_uchar3(x) __builtin_astype((x), uchar3)
- #define as_uchar4(x) __builtin_astype((x), uchar4)
- #define as_uchar8(x) __builtin_astype((x), uchar8)
- #define as_uchar16(x) __builtin_astype((x), uchar16)
- #define as_short(x) __builtin_astype((x), short)
- #define as_short2(x) __builtin_astype((x), short2)
- #define as_short3(x) __builtin_astype((x), short3)
- #define as_short4(x) __builtin_astype((x), short4)
- #define as_short8(x) __builtin_astype((x), short8)
- #define as_short16(x) __builtin_astype((x), short16)
- #define as_ushort(x) __builtin_astype((x), ushort)
- #define as_ushort2(x) __builtin_astype((x), ushort2)
- #define as_ushort3(x) __builtin_astype((x), ushort3)
- #define as_ushort4(x) __builtin_astype((x), ushort4)
- #define as_ushort8(x) __builtin_astype((x), ushort8)
- #define as_ushort16(x) __builtin_astype((x), ushort16)
- #define as_int(x) __builtin_astype((x), int)
- #define as_int2(x) __builtin_astype((x), int2)
- #define as_int3(x) __builtin_astype((x), int3)
- #define as_int4(x) __builtin_astype((x), int4)
- #define as_int8(x) __builtin_astype((x), int8)
- #define as_int16(x) __builtin_astype((x), int16)
- #define as_uint(x) __builtin_astype((x), uint)
- #define as_uint2(x) __builtin_astype((x), uint2)
- #define as_uint3(x) __builtin_astype((x), uint3)
- #define as_uint4(x) __builtin_astype((x), uint4)
- #define as_uint8(x) __builtin_astype((x), uint8)
- #define as_uint16(x) __builtin_astype((x), uint16)
- #define as_long(x) __builtin_astype((x), long)
- #define as_long2(x) __builtin_astype((x), long2)
- #define as_long3(x) __builtin_astype((x), long3)
- #define as_long4(x) __builtin_astype((x), long4)
- #define as_long8(x) __builtin_astype((x), long8)
- #define as_long16(x) __builtin_astype((x), long16)
- #define as_ulong(x) __builtin_astype((x), ulong)
- #define as_ulong2(x) __builtin_astype((x), ulong2)
- #define as_ulong3(x) __builtin_astype((x), ulong3)
- #define as_ulong4(x) __builtin_astype((x), ulong4)
- #define as_ulong8(x) __builtin_astype((x), ulong8)
- #define as_ulong16(x) __builtin_astype((x), ulong16)
- #define as_half(x) __builtin_astype((x), half)
- #define as_half2(x) __builtin_astype((x), half2)
- #define as_half3(x) __builtin_astype((x), half3)
- #define as_half4(x) __builtin_astype((x), half4)
- #define as_half8(x) __builtin_astype((x), half8)

- #define as_half16(x) __builtin_astype((x), half16)
- #define as_float(x) __builtin_astype((x), float)
- #define as_float2(x) __builtin_astype((x), float2)
- #define as_float3(x) __builtin_astype((x), float3)
- #define as_float4(x) __builtin_astype((x), float4)
- #define as_float8(x) __builtin_astype((x), float8)
- #define as_float16(x) __builtin_astype((x), float16)
- #define as_double(x) __builtin_astype((x), double)
- #define as_double2(x) __builtin_astype((x), double2)
- #define as_double3(x) __builtin_astype((x), double3)
- #define as_double4(x) __builtin_astype((x), double4)
- #define as_double8(x) __builtin_astype((x), double8)
- #define as_double16(x) __builtin_astype((x), double16)

Typedefs

- typedef unsigned char uchar

  An unsigned 8-bit integer.
- typedef unsigned short ushort

  An unsigned 16-bit integer.
- typedef unsigned int uint

  An unsigned 32-bit integer.
- typedef unsigned long ulong

  An unsigned 64-bit integer.
- typedef ptrdiff_t intptr_t

  The unsigned integer type of the result of the sizeof operator.
- typedef size_t uintptr_t
- typedef char char2 __attribute__((ext_vector_type(2)))

  Returns the number of global work-items specified for dimension identified by dimindx.
- typedef uint cl_mem_fence_flags

  All work-items in a work-group executing the kernel on a processor must execute this function before any are allowed to continue execution beyond the barrier.

Functions

- uint const_func __attribute__ ((overloadable)) get_work_dim(void)

  Returns the number of dimensions in use.
- int printf (__constant const char *st,...)

Variables

- float const_func float x
- float const_func float y
- float const_func float b
- float const_func float float c
- float __global float * iptr
- float __global int * exp
- float const_func int n
- float __global int * signp
- float float __global int * quo

- float __global float * cosval
- char const_func char minval
- char const_func char char maxval
- char const_func char i
- short const_func uchar lo
- int const_func int int z
- float const_func float float a
- float const_func float edge1
- float4 const_func float4 p1
- char2 const __global char * p
- void size_t offset
- event_t const __global char * src
- event_t const __global char size_t num_elements
- event_t const __global char size_t event_t event
- event_t const __global char size_t size_t src_stride
- event_t const __local char size_t size_t dst_stride
- void event_t * event_list
- int int val
- int int cmp
- char2 const_func uchar2 mask
- float4 sampler_t sampler
- float4 sampler_t int2 coord
- void int2 float4 color

13.83.1 Macro Definition Documentation

13.83.1.1 __kernel_exec

```
#define __kernel_exec(
              X,
              typen )
```

Value:

```
    __kernel \
__attribute__((work_group_size_hint(X, 1, 1))) \
    __attribute__((vec_type_hint(typen)))
```

Definition at line 6483 of file opencl_spir.h.

13.83.1.2 as_char

```
#define as_char(
              x ) __builtin_astype((x), char)
```

OpenCL as_typen operators Reinterprets a data type as another data type of the same size.

Definition at line 7497 of file opencl_spir.h.

13.83.1.3 as_char16

```
#define as_char16(
              x ) __builtin_astype((x), char16)
```

Definition at line 7502 of file opencl_spir.h.

13.83.1.4 as_char2

```
#define as_char2(
              x ) __builtin_astype((x), char2)
```

Definition at line 7498 of file opencl_spir.h.

13.83.1.5 as_char3

```
#define as_char3(
              x ) __builtin_astype((x), char3)
```

Definition at line 7499 of file opencl_spir.h.

13.83.1.6 as_char4

```
#define as_char4(
              x ) __builtin_astype((x), char4)
```

Definition at line 7500 of file opencl_spir.h.

13.83.1.7 as_char8

```
#define as_char8(
              x ) __builtin_astype((x), char8)
```

Definition at line 7501 of file opencl_spir.h.

13.83.1.8 as_double

```
#define as_double(
              x ) __builtin_astype((x), double)
```

Definition at line 7567 of file opencl_spir.h.

13.83.1.9 as_double16

```
#define as_double16(
              x ) __builtin_astype((x), double16)
```

Definition at line 7572 of file opencl_spir.h.

13.83.1.10 as_double2

```
#define as_double2(
              x ) __builtin_astype((x), double2)
```

Definition at line 7568 of file opencl_spir.h.

13.83.1.11 as_double3

```
#define as_double3(
              x ) __builtin_astype((x), double3)
```

Definition at line 7569 of file opencl_spir.h.

13.83.1.12 as_double4

```
#define as_double4(
              x ) __builtin_astype((x), double4)
```

Definition at line 7570 of file opencl_spir.h.

13.83.1.13 as_double8

```
#define as_double8(
              x ) __builtin_astype((x), double8)
```

Definition at line 7571 of file opencl_spir.h.

13.83.1.14 as_float

```
#define as_float(
              x ) __builtin_astype((x), float)
```

Definition at line 7560 of file opencl_spir.h.

13.83.1.15 as_float16

```
#define as_float16(
              x ) __builtin_astype((x), float16)
```

Definition at line 7565 of file opencl_spir.h.

13.83.1.16 as_float2

```
#define as_float2(
              x ) __builtin_astype((x), float2)
```

Definition at line 7561 of file opencl_spir.h.

13.83.1.17 as_float3

```
#define as_float3(
              x ) __builtin_astype((x), float3)
```

Definition at line 7562 of file opencl_spir.h.

13.83.1.18 as_float4

```
#define as_float4(
              x ) __builtin_astype((x), float4)
```

Definition at line 7563 of file opencl_spir.h.

13.83.1.19 as_float8

```
#define as_float8(
              x ) __builtin_astype((x), float8)
```

Definition at line 7564 of file opencl_spir.h.

13.83.1.20 as_half

```
#define as_half(
              x ) __builtin_astype((x), half)
```

Definition at line 7553 of file opencl_spir.h.

13.83.1.21 as_half16

```
#define as_half16(
              x ) __builtin_astype((x), half16)
```

Definition at line 7558 of file opencl_spir.h.

13.83.1.22 as_half2

```
#define as_half2(
              x ) __builtin_astype((x), half2)
```

Definition at line 7554 of file opencl_spir.h.

13.83.1.23 as_half3

```
#define as_half3(
              x ) __builtin_astype((x), half3)
```

Definition at line 7555 of file opencl_spir.h.

13.83.1.24 as_half4

```
#define as_half4(
              x ) __builtin_astype((x), half4)
```

Definition at line 7556 of file opencl_spir.h.

13.83.1.25 as_half8

```
#define as_half8(
              x ) __builtin_astype((x), half8)
```

Definition at line 7557 of file opencl_spir.h.

13.83.1.26 as_int

```
#define as_int(
              x ) __builtin_astype((x), int)
```

Definition at line 7525 of file opencl_spir.h.

13.83.1.27 as_int16

```
#define as_int16(
              x ) __builtin_astype((x), int16)
```

Definition at line 7530 of file opencl_spir.h.

13.83.1.28 as_int2

```
#define as_int2(
              x ) __builtin_astype((x), int2)
```

Definition at line 7526 of file opencl_spir.h.

13.83.1.29 as_int3

```
#define as_int3(
              x ) __builtin_astype((x), int3)
```

Definition at line 7527 of file opencl_spir.h.

13.83.1.30 as_int4

```
#define as_int4(
              x ) __builtin_astype((x), int4)
```

Definition at line 7528 of file opencl_spir.h.

13.83.1.31 as_int8

```
#define as_int8(
              x ) __builtin_astype((x), int8)
```

Definition at line 7529 of file opencl_spir.h.

13.83.1.32 as_long

```
#define as_long(
              x ) __builtin_astype((x), long)
```

Definition at line 7539 of file opencl_spir.h.

13.83.1.33 as_long16

```
#define as_long16(
              x ) __builtin_astype((x), long16)
```

Definition at line 7544 of file opencl_spir.h.

13.83.1.34 as_long2

```
#define as_long2(
              x ) __builtin_astype((x), long2)
```

Definition at line 7540 of file opencl_spir.h.

13.83.1.35 as_long3

```
#define as_long3(
              x ) __builtin_astype((x), long3)
```

Definition at line 7541 of file opencl_spir.h.

13.83.1.36 as_long4

```
#define as_long4(
              x ) __builtin_astype((x), long4)
```

Definition at line 7542 of file opencl_spir.h.

13.83.1.37 as_long8

```
#define as_long8(
              x ) __builtin_astype((x), long8)
```

Definition at line 7543 of file opencl_spir.h.

13.83.1.38 as_short

```
#define as_short(
              x ) __builtin_astype((x), short)
```

Definition at line 7511 of file opencl_spir.h.

13.83.1.39 as_short16

```
#define as_short16(
              x ) __builtin_astype((x), short16)
```

Definition at line 7516 of file opencl_spir.h.

13.83.1.40 as_short2

```
#define as_short2(
              x ) __builtin_astype((x), short2)
```

Definition at line 7512 of file opencl_spir.h.

13.83.1.41 as_short3

```
#define as_short3(
              x ) __builtin_astype((x), short3)
```

Definition at line 7513 of file opencl_spir.h.

13.83.1.42 as_short4

```
#define as_short4(
              x ) __builtin_astype((x), short4)
```

Definition at line 7514 of file opencl_spir.h.

13.83.1.43 as_short8

```
#define as_short8(
              x ) __builtin_astype((x), short8)
```

Definition at line 7515 of file opencl_spir.h.

13.83.1.44 as_uchar

```
#define as_uchar(
              x ) __builtin_astype((x), uchar)
```

Definition at line 7504 of file opencl_spir.h.

13.83.1.45 as_uchar16

```
#define as_uchar16(
              x ) __builtin_astype((x), uchar16)
```

Definition at line 7509 of file opencl_spir.h.

13.83.1.46 as_uchar2

```
#define as_uchar2(
              x ) __builtin_astype((x), uchar2)
```

Definition at line 7505 of file opencl_spir.h.

13.83.1.47 as_uchar3

```
#define as_uchar3(
              x ) __builtin_astype((x), uchar3)
```

Definition at line 7506 of file opencl_spir.h.

13.83.1.48 as_uchar4

```
#define as_uchar4(
              x ) __builtin_astype((x), uchar4)
```

Definition at line 7507 of file opencl_spir.h.

13.83.1.49 as_uchar8

```
#define as_uchar8(
              x ) __builtin_astype((x), uchar8)
```

Definition at line 7508 of file opencl_spir.h.

13.83.1.50 as_uint

```
#define as_uint(
              x ) __builtin_astype((x), uint)
```

Definition at line 7532 of file opencl_spir.h.

13.83.1.51 as_uint16

```
#define as_uint16(
              x ) __builtin_astype((x), uint16)
```

Definition at line 7537 of file opencl_spir.h.

13.83.1.52 as_uint2

```
#define as_uint2(
              x ) __builtin_astype((x), uint2)
```

Definition at line 7533 of file opencl_spir.h.

13.83.1.53 as_uint3

```
#define as_uint3(
              x ) __builtin_astype((x), uint3)
```

Definition at line 7534 of file opencl_spir.h.

13.83.1.54 as_uint4

```
#define as_uint4(
              x ) __builtin_astype((x), uint4)
```

Definition at line 7535 of file opencl_spir.h.

13.83.1.55 as_uint8

```
#define as_uint8(
              x ) __builtin_astype((x), uint8)
```

Definition at line 7536 of file opencl_spir.h.

13.83.1.56 as_ulong

```
#define as_ulong(
              x ) __builtin_astype((x), ulong)
```

Definition at line 7546 of file opencl_spir.h.

13.83.1.57 as_ulong16

```
#define as_ulong16(
              x ) __builtin_astype((x), ulong16)
```

Definition at line 7551 of file opencl_spir.h.

13.83.1.58 as_ulong2

```
#define as_ulong2(
              x ) __builtin_astype((x), ulong2)
```

Definition at line 7547 of file opencl_spir.h.

13.83.1.59 as_ulong3

```
#define as_ulong3(
              x ) __builtin_astype((x), ulong3)
```

Definition at line 7548 of file opencl_spir.h.

13.83.1.60 as_ulong4

```
#define as_ulong4(
              x ) __builtin_astype((x), ulong4)
```

Definition at line 7549 of file opencl_spir.h.

13.83.1.61 as_ulong8

```
#define as_ulong8(
              x ) __builtin_astype((x), ulong8)
```

Definition at line 7550 of file opencl_spir.h.

13.83.1.62 as_ushort

```
#define as_ushort(
              x ) __builtin_astype((x), ushort)
```

Definition at line 7518 of file opencl_spir.h.

13.83.1.63 as_ushort16

```
#define as_ushort16(
              x ) __builtin_astype((x), ushort16)
```

Definition at line 7523 of file opencl_spir.h.

13.83.1.64 as_ushort2

```
#define as_ushort2(
              x ) __builtin_astype((x), ushort2)
```

Definition at line 7519 of file opencl_spir.h.

13.83.1.65 as_ushort3

```
#define as_ushort3(
              x ) __builtin_astype((x), ushort3)
```

Definition at line 7520 of file opencl_spir.h.

13.83.1.66 as_ushort4

```
#define as_ushort4(
              x ) __builtin_astype((x), ushort4)
```

Definition at line 7521 of file opencl_spir.h.

13.83.1.67 as_ushort8

```
#define as_ushort8(
              x ) __builtin_astype((x), ushort8)
```

Definition at line 7522 of file opencl_spir.h.

13.83.1.68 CHAR_BIT

```
#define CHAR_BIT 8
```

Definition at line 218 of file opencl_spir.h.

13.83.1.69 CHAR_MAX

```
#define CHAR_MAX SCHAR_MAX
```

Definition at line 222 of file opencl_spir.h.

13.83.1.70 CHAR_MIN

```
#define CHAR_MIN SCHAR_MIN
```

Definition at line 223 of file opencl_spir.h.

13.83.1.71 CLK_A

```
#define CLK_A 0x10B1
```

Definition at line 7356 of file opencl_spir.h.

13.83.1.72 CLK_ADDRESS_CLAMP

```
#define CLK_ADDRESS_CLAMP 4
```

Definition at line 7011 of file opencl_spir.h.

13.83.1.73 CLK_ADDRESS_CLAMP_TO_EDGE

```
#define CLK_ADDRESS_CLAMP_TO_EDGE 2
```

Definition at line 7010 of file opencl_spir.h.

13.83.1.74 CLK_ADDRESS_MIRRORED_REPEAT

```
#define CLK_ADDRESS_MIRRORED_REPEAT 8
```

Definition at line 7013 of file opencl_spir.h.

13.83.1.75 CLK_ADDRESS_NONE

```
#define CLK_ADDRESS_NONE 0
```

Use the coordinate (x, y) to do an element lookup in the 2D image object specified by image.

read_imagef returns floating-point values in the range [0.0 ... 1.0] for image objects created with image_channel↩ _data_type set to one of the predefined packed formats or CL_UNORM_INT8, or CL_UNORM_INT16. read_imagef returns floating-point values in the range [-1.0 ... 1.0] for image objects created with image_channel_data_type set to CL_SNORM_INT8, or CL_SNORM_INT16. read_imagef returns floating-point values for image objects created with image_channel_data_type set to CL_HALF_FLOAT or CL_FLOAT. The read_imagef calls that take integer coordinates must use a sampler with filter mode set to CLK_FILTER_NEAREST, normalized coordinates set to CLK_NORMALIZED_COORDS_FALSE and addressing mode set to CLK_ADDRESS_CLAMP_TO_EDGE, CLK↩ _ADDRESS_CLAMP or CLK_ADDRESS_NONE; otherwise the values returned are undefined. Values returned by read_imagef for image objects with image_channel_data_type values not specified in the description above are undefined.

Definition at line 7009 of file opencl_spir.h.

13.83.1.76 CLK_ADDRESS_REPEAT

```
#define CLK_ADDRESS_REPEAT 6
```

Definition at line 7012 of file opencl_spir.h.

13.83.1.77 CLK_ARGB

```
#define CLK_ARGB 0x10B7
```

Definition at line 7362 of file opencl_spir.h.

13.83.1.78 CLK_BGRA

```
#define CLK_BGRA 0x10B6
```

Definition at line 7361 of file opencl_spir.h.

13.83.1.79 CLK_DEPTH

```
#define CLK_DEPTH 0x10BD
```

Definition at line 7368 of file opencl_spir.h.

13.83.1.80 CLK_DEPTH_STENCIL

```
#define CLK_DEPTH_STENCIL 0x10BE
```

Definition at line 7369 of file opencl_spir.h.

13.83.1.81 CLK_FILTER_LINEAR

```
#define CLK_FILTER_LINEAR 0x20
```

Definition at line 7025 of file opencl_spir.h.

13.83.1.82 CLK_FILTER_NEAREST

```
#define CLK_FILTER_NEAREST 0x10
```

Definition at line 7024 of file opencl_spir.h.

13.83.1.83 CLK_FLOAT

```
#define CLK_FLOAT 0x10DE
```

Definition at line 7388 of file opencl_spir.h.

13.83.1.84 CLK_GLOBAL_MEM_FENCE

```
#define CLK_GLOBAL_MEM_FENCE 0x2
```

Queue a memory fence to ensure correct ordering of memory operations to global memory.

Definition at line 5831 of file opencl_spir.h.

13.83.1.85 CLK_HALF_FLOAT

```
#define CLK_HALF_FLOAT 0x10DD
```

Definition at line 7387 of file opencl_spir.h.

13.83.1.86 CLK_INTENSITY

```
#define CLK_INTENSITY 0x10B8
```

Definition at line 7363 of file opencl_spir.h.

13.83.1.87 CLK_LOCAL_MEM_FENCE

```
#define CLK_LOCAL_MEM_FENCE 0x1
```

Queue a memory fence to ensure correct ordering of memory operations to local memory.

Definition at line 5825 of file opencl_spir.h.

13.83.1.88 CLK_LUMINANCE

```
#define CLK_LUMINANCE 0x10B9
```

Definition at line 7364 of file opencl_spir.h.

13.83.1.89 CLK_NORMALIZED_COORDS_FALSE

```
#define CLK_NORMALIZED_COORDS_FALSE 0
```

Definition at line 7018 of file opencl_spir.h.

13.83.1.90 CLK_NORMALIZED_COORDS_TRUE

```
#define CLK_NORMALIZED_COORDS_TRUE 1
```

Definition at line 7019 of file opencl_spir.h.

13.83.1.91 CLK_R

```
#define CLK_R 0x10B0
```

Return the channel data type.

Valid values are: CLK_SNORM_INT8 CLK_SNORM_INT16 CLK_UNORM_INT8 CLK_UNORM_INT16 CLK_↩ UNORM_SHORT_565 CLK_UNORM_SHORT_555 CLK_UNORM_SHORT_101010 CLK_SIGNED_INT8 CLK_↩ SIGNED_INT16 CLK_SIGNED_INT32 CLK_UNSIGNED_INT8 CLK_UNSIGNED_INT16 CLK_UNSIGNED_INT32 CLK_HALF_FLOAT CLK_FLOAT

Definition at line 7355 of file opencl_spir.h.

13.83.1.92 CLK_RA

```
#define CLK_RA 0x10B3
```

Definition at line 7358 of file opencl_spir.h.

13.83.1.93 CLK_RG

```
#define CLK_RG 0x10B2
```

Definition at line 7357 of file opencl_spir.h.

13.83.1.94 CLK_RGB

```
#define CLK_RGB 0x10B4
```

Definition at line 7359 of file opencl_spir.h.

13.83.1.95 CLK_RGBA

```
#define CLK_RGBA 0x10B5
```

Definition at line 7360 of file opencl_spir.h.

13.83.1.96 CLK_RGBx

```
#define CLK_RGBx 0x10BC
```

Definition at line 7367 of file opencl_spir.h.

13.83.1.97 CLK_RGx

```
#define CLK_RGx 0x10BB
```

Definition at line 7366 of file opencl_spir.h.

13.83.1.98 CLK_Rx

```
#define CLK_Rx 0x10BA
```

Definition at line 7365 of file opencl_spir.h.

13.83.1.99 CLK_SIGNED_INT16

```
#define CLK_SIGNED_INT16 0x10D8
```

Definition at line 7382 of file opencl_spir.h.

13.83.1.100 CLK_SIGNED_INT32

```
#define CLK_SIGNED_INT32 0x10D9
```

Definition at line 7383 of file opencl_spir.h.

13.83.1.101 CLK_SIGNED_INT8

```
#define CLK_SIGNED_INT8 0x10D7
```

Definition at line 7381 of file opencl_spir.h.

13.83.1.102 CLK_SNORM_INT16

```
#define CLK_SNORM_INT16 0x10D1
```

Definition at line 7375 of file opencl_spir.h.

13.83.1.103 CLK_SNORM_INT8

```
#define CLK_SNORM_INT8 0x10D0
```

Definition at line 7374 of file opencl_spir.h.

13.83.1.104 CLK_UNORM_INT16

```
#define CLK_UNORM_INT16 0x10D3
```

Definition at line 7377 of file opencl_spir.h.

13.83.1.105 CLK_UNORM_INT24

```
#define CLK_UNORM_INT24 0x10DF
```

Definition at line 7389 of file opencl_spir.h.

13.83.1.106 CLK_UNORM_INT8

```
#define CLK_UNORM_INT8 0x10D2
```

Definition at line 7376 of file opencl_spir.h.

13.83.1.107 CLK_UNORM_INT_101010

```
#define CLK_UNORM_INT_101010 0x10D6
```

Definition at line 7380 of file opencl_spir.h.

13.83.1.108 CLK_UNORM_SHORT_555

```
#define CLK_UNORM_SHORT_555 0x10D5
```

Definition at line 7379 of file opencl_spir.h.

13.83.1.109 CLK_UNORM_SHORT_565

```
#define CLK_UNORM_SHORT_565 0x10D4
```

Definition at line 7378 of file opencl_spir.h.

13.83.1.110 CLK_UNSIGNED_INT16

```
#define CLK_UNSIGNED_INT16 0x10DB
```

Definition at line 7385 of file opencl_spir.h.

13.83.1.111 CLK_UNSIGNED_INT32

```
#define CLK_UNSIGNED_INT32 0x10DC
```

Definition at line 7386 of file opencl_spir.h.

13.83.1.112 CLK_UNSIGNED_INT8

```
#define CLK_UNSIGNED_INT8 0x10DA
```

Definition at line 7384 of file opencl_spir.h.

13.83.1.113 const_func

```
#define const_func __attribute__((const))
```

Added according to pocl/examples/example1-spir64/generate_spir.sh.

Todo avoid this hack

Definition at line 21 of file opencl_spir.h.

13.83.1.114 DBL_DIG [1/2]

```
#define DBL_DIG 15
```

Definition at line 234 of file opencl_spir.h.

13.83.1.115 DBL_DIG [2/2]

```
#define DBL_DIG 15
```

Definition at line 234 of file opencl_spir.h.

13.83.1.116 DBL_EPSILON [1/2]

```
#define DBL_EPSILON 0x1.0p-52
```

Definition at line 242 of file opencl_spir.h.

13.83.1.117 DBL_EPSILON [2/2]

```
#define DBL_EPSILON 0x1.0p-52
```

Definition at line 242 of file opencl_spir.h.

13.83.1.118 DBL_MANT_DIG [1/2]

```
#define DBL_MANT_DIG 53
```

Definition at line 235 of file opencl_spir.h.

13.83.1.119 DBL_MANT_DIG [2/2]

```
#define DBL_MANT_DIG 53
```

Definition at line 235 of file opencl_spir.h.

13.83.1.120 DBL_MAX [1/2]

```
#define DBL_MAX 0x1.fffffffffffffp1023
```

Definition at line 240 of file opencl_spir.h.

13.83.1.121 DBL_MAX [2/2]

```
#define DBL_MAX 0x1.fffffffffffffp1023
```

Definition at line 240 of file opencl_spir.h.

13.83.1.122 DBL_MAX_10_EXP [1/2]

```
#define DBL_MAX_10_EXP +308
```

Definition at line 236 of file opencl_spir.h.

13.83.1.123 DBL_MAX_10_EXP [2/2]

```
#define DBL_MAX_10_EXP +308
```

Definition at line 236 of file opencl_spir.h.

13.83.1.124 DBL_MAX_EXP [1/2]

```
#define DBL_MAX_EXP +1024
```

Definition at line 237 of file opencl_spir.h.

13.83.1.125 DBL_MAX_EXP [2/2]

```
#define DBL_MAX_EXP +1024
```

Definition at line 237 of file opencl_spir.h.

13.83.1.126 DBL_MIN [1/2]

```
#define DBL_MIN 0x1.0p-1022
```

Definition at line 241 of file opencl_spir.h.

13.83.1.127 DBL_MIN [2/2]

```
#define DBL_MIN 0x1.0p-1022
```

Definition at line 241 of file opencl_spir.h.

13.83.1.128 DBL_MIN_10_EXP [1/2]

```
#define DBL_MIN_10_EXP -307
```

Definition at line 238 of file opencl_spir.h.

13.83.1.129 DBL_MIN_10_EXP [2/2]

```
#define DBL_MIN_10_EXP -307
```

Definition at line 238 of file opencl_spir.h.

13.83.1.130 DBL_MIN_EXP [1/2]

```
#define DBL_MIN_EXP -1021
```

Definition at line 239 of file opencl_spir.h.

13.83.1.131 DBL_MIN_EXP [2/2]

```
#define DBL_MIN_EXP -1021
```

Definition at line 239 of file opencl_spir.h.

13.83.1.132 DBL_RADIX

```
#define DBL_RADIX 2
```

Definition at line 184 of file opencl_spir.h.

13.83.1.133 FLT_DIG

```
#define FLT_DIG 6
```

Definition at line 167 of file opencl_spir.h.

13.83.1.134 FLT_EPSILON

```
#define FLT_EPSILON 0x1.0p-23f
```

Definition at line 176 of file opencl_spir.h.

13.83.1.135 FLT_MANT_DIG

```
#define FLT_MANT_DIG 24
```

Definition at line 168 of file opencl_spir.h.

13.83.1.136 FLT_MAX

```
#define FLT_MAX 0x1.fffffep127f
```

Definition at line 174 of file opencl_spir.h.

13.83.1.137 FLT_MAX_10_EXP

```
#define FLT_MAX_10_EXP +38
```

Definition at line 169 of file opencl_spir.h.

13.83.1.138 FLT_MAX_EXP

```
#define FLT_MAX_EXP +128
```

Definition at line 170 of file opencl_spir.h.

13.83.1.139 FLT_MIN

```
#define FLT_MIN 0x1.0p-126f
```

Definition at line 175 of file opencl_spir.h.

13.83.1.140 FLT_MIN_10_EXP

```
#define FLT_MIN_10_EXP -37
```

Definition at line 171 of file opencl_spir.h.

13.83.1.141 FLT_MIN_EXP

```
#define FLT_MIN_EXP -125
```

Definition at line 172 of file opencl_spir.h.

13.83.1.142 FLT_RADIX

```
#define FLT_RADIX 2
```

Definition at line 173 of file opencl_spir.h.

13.83.1.143 FP_ILOGB0

```
#define FP_ILOGB0 INT_MIN
```

Definition at line 164 of file opencl_spir.h.

13.83.1.144 FP_ILOGBNAN

```
#define FP_ILOGBNAN INT_MAX
```

Definition at line 165 of file opencl_spir.h.

13.83.1.145 HUGE_VAL

```
#define HUGE_VAL (__builtin_huge_val())
```

A positive double constant expression.

HUGE_VAL evaluates to +infinity. Used as an error value returned by the built-in math functions.

Definition at line 151 of file opencl_spir.h.

13.83.1.146 HUGE_VALF

```
#define HUGE_VALF (__builtin_huge_valf())
```

A positive float constant expression.

HUGE_VALF evaluates to +infinity. Used as an error value returned by the built-in math functions.

Definition at line 144 of file opencl_spir.h.

13.83.1.147 INFINITY

```
#define INFINITY (__builtin_inff())
```

A constant expression of type float representing positive or unsigned infinity.

Definition at line 157 of file opencl_spir.h.

13.83.1.148 INT_MAX

```
#define INT_MAX 2147483647
```

Definition at line 228 of file opencl_spir.h.

13.83.1.149 INT_MIN

```
#define INT_MIN (-2147483647-1)
```

Definition at line 229 of file opencl_spir.h.

13.83.1.150 kernel_exec

```
#define kernel_exec(
              X,
              typen )
```

Value:

```
    __kernel \
__attribute__((work_group_size_hint(X, 1, 1))) \
    __attribute__((vec_type_hint(typen)))
```

Definition at line 6487 of file opencl_spir.h.

13.83.1.151 LONG_MAX

```
#define LONG_MAX ((long)0x7fffffffffffffffL)
```

Definition at line 231 of file opencl_spir.h.

13.83.1.152 LONG_MIN

```
#define LONG_MIN ((long)(-0x7fffffffffffffffL-1))
```

Definition at line 232 of file opencl_spir.h.

13.83.1.153 M_1_PI

```
#define M_1_PI 0x1.45f306dc9c883p-2
```

Definition at line 212 of file opencl_spir.h.

13.83.1.154 M_1_PI_F

```
#define M_1_PI_F 0.31830988618379067153776752674502872 4f
```

Definition at line 198 of file opencl_spir.h.

13.83.1.155 M_2_PI

```
#define M_2_PI 0x1.45f306dc9c883p-1
```

Definition at line 213 of file opencl_spir.h.

13.83.1.156 M_2_PI_F

```
#define M_2_PI_F 0.636619772367581343075535053490057448f
```

Definition at line 199 of file opencl_spir.h.

13.83.1.157 M_2_SQRTPI

```
#define M_2_SQRTPI 0x1.20dd750429b6dp+0
```

Definition at line 214 of file opencl_spir.h.

13.83.1.158 M_2_SQRTPI_F

```
#define M_2_SQRTPI_F 1.12837916709551257389615890312154517f
```

Definition at line 200 of file opencl_spir.h.

13.83.1.159 M_E

```
#define M_E 0x1.5bf0a8b145769p+1
```

Definition at line 204 of file opencl_spir.h.

13.83.1.160 M_E_F

```
#define M_E_F 2.71828182845904523536028747135266250f
```

Definition at line 190 of file opencl_spir.h.

13.83.1.161 M_LN10

```
#define M_LN10 0x1.26bb1bbb55516p+1
```

Definition at line 208 of file opencl_spir.h.

13.83.1.162 M_LN10_F

```
#define M_LN10_F 2.3025850929940456840179914546843642 1f
```

Definition at line 194 of file opencl_spir.h.

13.83.1.163 M_LN2

```
#define M_LN2 0x1.62e42fefa39efp-1
```

Definition at line 207 of file opencl_spir.h.

13.83.1.164 M_LN2_F

```
#define M_LN2_F 0.693147180559945309417232121458176568f
```

Definition at line 193 of file opencl_spir.h.

13.83.1.165 M_LOG10E

```
#define M_LOG10E 0x1.bcb7b1526e50ep-2
```

Definition at line 206 of file opencl_spir.h.

13.83.1.166 M_LOG10E_F

```
#define M_LOG10E_F 0.43429448190325182765112891891660508 2f
```

Definition at line 192 of file opencl_spir.h.

13.83.1.167 M_LOG2E

```
#define M_LOG2E 0x1.71547652b82fep+0
```

Definition at line 205 of file opencl_spir.h.

13.83.1.168 M_LOG2E_F

```
#define M_LOG2E_F 1.44269504088896340735992468100189214f
```

Definition at line 191 of file opencl_spir.h.

13.83.1.169 M_PI

```
#define M_PI 0x1.921fb54442d18p+1
```

Definition at line 209 of file opencl_spir.h.

13.83.1.170 M_PI_2

```
#define M_PI_2 0x1.921fb54442d18p+0
```

Definition at line 210 of file opencl_spir.h.

13.83.1.171 M_PI_2_F

```
#define M_PI_2_F 1.57079632679489661923132169163975144f
```

Definition at line 196 of file opencl_spir.h.

13.83.1.172 M_PI_4

```
#define M_PI_4 0x1.921fb54442d18p-1
```

Definition at line 211 of file opencl_spir.h.

13.83.1.173 M_PI_4_F

```
#define M_PI_4_F 0.78539816339744830961566084581987572l
```

Definition at line 197 of file opencl_spir.h.

13.83.1.174 M_PI_F

```
#define M_PI_F 3.14159265358979323846264338327950288f
```

Definition at line 195 of file opencl_spir.h.

13.83.1.175 M_SQRT1_2

```
#define M_SQRT1_2 0x1.6a09e667f3bcdp-1
```

Definition at line 216 of file opencl_spir.h.

13.83.1.176 M_SQRT1_2_F

```
#define M_SQRT1_2_F 0.70710678118654752440084436210484903f
```

Definition at line 202 of file opencl_spir.h.

13.83.1.177 M_SQRT2

```
#define M_SQRT2 0x1.6a09e667f3bcdp+0
```

Definition at line 215 of file opencl_spir.h.

13.83.1.178 M_SQRT2_F

```
#define M_SQRT2_F 1.41421356237309504880168872420969808f
```

Definition at line 201 of file opencl_spir.h.

13.83.1.179 MAXFLOAT

```
#define MAXFLOAT 0x1.fffffep127f
```

Value of maximum non-infinite single-precision floating-point number.

Definition at line 137 of file opencl_spir.h.

13.83.1.180 NAN

```
#define NAN as_float(__builtin_nanf((char const *)""))
```

A constant expression of type float representing a quiet NaN.

Definition at line 162 of file opencl_spir.h.

13.83.1.181 readonly

```
#define readonly __attribute__((pure))
```

Definition at line 22 of file opencl_spir.h.

13.83.1.182 SCHAR_MAX

```
#define SCHAR_MAX 127
```

Definition at line 219 of file opencl_spir.h.

13.83.1.183 SCHAR_MIN

```
#define SCHAR_MIN (-128)
```

Definition at line 220 of file opencl_spir.h.

13.83.1.184 SHRT_MAX

```
#define SHRT_MAX 32767
```

Definition at line 225 of file opencl_spir.h.

13.83.1.185 SHRT_MIN

```
#define SHRT_MIN (-32768)
```

Definition at line 226 of file opencl_spir.h.

13.83.1.186 UCHAR_MAX

```
#define UCHAR_MAX 255
```

Definition at line 221 of file opencl_spir.h.

13.83.1.187 UINT_MAX

```
#define UINT_MAX 0xffffffff
```

Definition at line 227 of file opencl_spir.h.

13.83.1.188 ULONG_MAX

```
#define ULONG_MAX 0xffffffffffffffffUL
```

Definition at line 230 of file opencl_spir.h.

13.83.1.189 USHRT_MAX

```
#define USHRT_MAX 65535
```

Definition at line 224 of file opencl_spir.h.

13.83.2 Typedef Documentation

13.83.2.1 __attribute__

```
std::size_t const_func __attribute__
```

Returns the number of global work-items specified for dimension identified by dimindx.

Write color value to location specified by coord.xy in the 2D image layer identified by index coord.z in the 2D image array.

Use coord.xy to do an element lookup in the 2D image layer identified by index coord.z in the 2D image array.

Return the number of images in the 1D image array.

Return the number of images in the 2D image array.

Return the 3D image width, height, and depth as an int4 type.

Return the 2D image width and height as an int2 type.

Return the image channel order.

Return the image depth in pixels.

Return the image height in pixels.

Return the image width in pixels.

Write color value to location specified by coordinate (x, y, z) in the 3D image object specified by image.

Use the coordinate (coord.x, coord.y, coord.z) to do an element lookup in the 3D image object specified by image.

Write color value to location specified by coordinate (x, y) in the 2D image object specified by image.

Use the coordinate (x, y) to do an element lookup in the 2D image object specified by image.

The shuffle and shuffle2 built-in functions construct a permutation of elements from one or two input vectors respectively that are of the same type, returning a vector with the same element type as the input and length that is the same as the shuffle mask.

Swaps the old value stored at location p with new value given by val.

unsigned int atomic_sub (volatile __global unsigned int *p, unsigned int val) int atomic_sub (volatile __local int *p, int val) unsigned int atomic_sub (volatile __local unsigned int *p, unsigned int val) Read the 32-bit value (referred to as old) stored at location pointed by p.

Read the 32-bit value (referred to as old) stored at location pointed by p.

Prefetch num_elements * sizeof(gentype) bytes into the global cache.

Wait for events that identify the async_work_group_copy operations to complete.

Perform an async gather of num_elements gentype elements from src to dst.

event_t async_work_group_copy (__global gentype *dst, const __local gentype *src, size_t num_elements, event_t event) Perform an async copy of num_elements gentype elements from src to dst.

Write memory barrier that orders only stores.

Read memory barrier that orders only loads.

Orders loads and stores of a work-item executing a kernel.

For n = 1, 2, 4, 8 and 16 read sizeof (halfn) bytes of data from address (p + (offset $*$ n)).

The floatn value given by data is converted to a halfn value using the appropriate rounding mode.

The float value given by data is first converted to a half value using the appropriate rounding mode.

Read sizeof (halfn) bytes of data from address (p + (offset $*$ n)).

Read sizeof (half) bytes of data from address (p + offset).

Write sizeof (gentypen) bytes given by data to address (p + (offset $*$ n)).

Return sizeof (gentypen) bytes of data read from address (p + (offset $*$ n)).

For each component of a vector type, result[i] = if MSB of c[i] is set ? b[i] : a[i].

Each bit of the result is the corresponding bit of a if the corresponding bit of c is 0.

Returns 1 if the most significant bit in all components of x is set; otherwise returns 0.

Returns 1 if the most significant bit in any component of x is set; otherwise returns 0.

Test for sign bit.

Test if arguments are unordered.

Test if arguments are ordered.

Test for a normal value.

Test for a NaN.

Test for infinity value (+ve or –ve) .

Test for finite value.

Returns the component-wise compare of (x $<$ y) $||$ (x $>$ y) .

Returns the component-wise compare of x $<$= y.

Returns the component-wise compare of x $<$ y.

Returns the component-wise compare of x $>$= y.

Returns the component-wise compare of x $>$ y.

Returns the component-wise compare of x != y.

intn isequal (floatn x, floatn y) Returns the component-wise compare of x == y.

Returns the length of vector p computed as: half_sqrt(p.x2 + p.y2 + ...)

Returns fast_length(p0 – p1).

Returns a vector in the same direction as p but with a length of 1.

Return the length of vector p, i.e., sqrt(p.x2 + p.y 2 + ...)

Returns the distance between p0 and p1.

Compute dot product.

Returns the cross product of p0.xyz and p1.xyz.

Returns 1.0 if x $>$ 0, -0.0 if x = -0.0, +0.0 if x = +0.0, or –1.0 if x $<$ 0.

Returns 0.0 if x $<=$ edge0 and 1.0 if x $>=$ edge1 and performs smooth Hermite interpolation between 0 and 1when edge0 $<$ x $<$ edge1.

Returns 0.0 if x $<$ edge, otherwise it returns 1.0.

Converts degrees to radians, i.e.

Returns the linear blend of x & y implemented as: x + (y – x) $*$ a a must be a value in the range 0.0 . . . 1.0.

Converts radians to degrees, i.e.

Returns fmin(fmax(x, minval), maxval).

Multiply two 24-bit integer values x and y.

Multiply two 24-bit integer values x and y and add the 32-bit integer result to the 32-bit integer z.

result[i] = ((long)hi[i] $<<$ 32) | lo[i] result[i] = ((ulong)hi[i] $<<$ 32) | lo[i]

result[i] = ((int)hi[i] $<<$ 16) | lo[i] result[i] = ((uint)hi[i] $<<$ 16) | lo[i]

result[i] = ((short)hi[i] $<<$ 8) | lo[i] result[i] = ((ushort)hi[i] $<<$ 8) | lo[i]

Returns x - y and saturates the result.

For each element in v, the bits are shifted left by the number of bits given by the corresponding element in i (subject to usual shift modulo rules described in section 6.3).

Computes x $*$ y and returns the high half of the product of x and y.

Returns a $*$ b + c and saturates the result.

Returns mul_hi(a, b) + c.

Returns the number of leading 0-bits in x, starting at the most significant bit position.

Returns min(max(x, minval), maxval).

Returns (x + y + 1) $>>$ 1.

Returns (x + y) $>>$ 1.

Returns x + y and saturates the result.

Returns | x – y | without modulo overflow.

Returns | x |.

Compute tangent over an implementation-defined range.

Compute square root over an implementation-defined range.

Compute sine over an implementation-defined range.

Compute inverse square root over an implementationdefined range.

Compute reciprocal over an implementation-defined range.

Compute a base 10 logarithm over an implementationdefined range.

Compute a base 2 logarithm over an implementationdefined range.

Compute natural logarithm over an implementationdefined range.

Compute the base- 10 exponential of x over an implementation-defined range.

Compute the base- 2 exponential of x over an implementation-defined range.

Compute the base- e exponential of x over an implementation-defined range.

Compute x / y over an implementation-defined range.

Compute cosine over an implementation-defined range.

Compute reciprocal.

Compute the base- 10 exponential of x.

Compute the base- 2 exponential of x.

Compute x / y.

Round to integral value using the round to zero rounding mode.

Compute the gamma function.

Compute tan (PI $*$ x).

Compute hyperbolic tangent.

Compute tangent.

Compute square root.

Compute sin (PI $*$ x).

Compute hyperbolic sine.

Compute sine and cosine of x.

Compute sine.

Compute inverse square root.

Return the integral value nearest to x rounding halfway cases away from zero, regardless of the current rounding direction.

Compute x to the power 1/y.

Round to integral value (using round to nearest even rounding mode) in floating-point format.

The remquo function computes the value r such that r = x - n∗y, where n is the integer nearest the exact value of x/y.

Compute the value r such that r = x - n∗y, where n is the integer nearest the exact value of x/y.

Compute x to the power y, where x is >= 0.

Compute x to the power y, where y is an integer.

Compute x to the power y.

Computes the next representable single-precision floating-point value following x in the direction of y.

Returns a quiet NaN.

Decompose a floating-point number.

Returns x if | x | < | y |, y if | y | < | x |, otherwise fmin(x, y).

Returns x if | x | > | y |, y if | y | > | x |, otherwise fmax(x, y).

mad approximates a ∗ b + c.

Compute the exponent of x, which is the integral part of logr | x |.

Compute a base e logarithm of (1.0 + x).

Compute a base 10 logarithm.

Compute a base 2 logarithm.

Compute natural logarithm.

Log gamma function.

Multiply x by 2 to the power n.

Return the exponent as an integer value.

Compute the value of the square root of x^2+ y^2 without undue overflow or underflow.

Extract mantissa and exponent from x.

Returns fmin(x – floor (x), 0x1.fffffep-1f).

Modulus.

Returns y if y < x, otherwise it returns x.

Returns y if x < y, otherwise it returns x.

Returns the correctly rounded floating-point representation of the sum of c with the infinitely precise product of a and b.

Round to integral value using the round to –ve infinity rounding mode.

x - y if x > y, +0 if x is less than or equal to y.

Compute absolute value of a floating-point number.

Compute e^x- 1.0.

Exponential base 10 function.

Exponential base 2 function.

Compute the base- e exponential of x.

Error function encountered in integrating the normal distribution.

Complementary error function.

Compute cos (PI $*$ x).

Compute hyperbolic consine.

Compute cosine.

Returns x with its sign changed to match the sign of y.

Round to integral value using the round to +ve infinity rounding mode.

Compute cube-root.

Compute atan2 (y, x) / PI.

Compute atan (x) / PI.

Hyperbolic arc tangent.

Arc tangent of y / x.

Arc tangent function.

Compute asin (x) / PI.

Inverse hyperbolic sine.

Arc sine function.

Compute acos (x) / PI.

Inverse hyperbolic cosine.

Arc cosine function.

get_global_offset returns the offset values specified in global_work_offset argument to clEnqueueNDRangeKernel.

get_group_id returns the work-group ID which is a number from 0 .

Returns the number of work-groups that will execute a kernel for dimension identified by dimindx.

Returns the unique local work-item ID i.e.

Returns the number of local work-items specified in dimension identified by dimindx.

Returns the unique global work-item ID value for dimension identified by dimindx.

This value is given by the global_work_size argument to clEnqueueNDRangeKernel. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_global_size() returns 1. For clEnqueueTask, this always returns 1.

The global work-item ID specifies the work-item ID based on the number of global work-items specified to execute the kernel. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_global_id() returns 0. For clEnqueueTask, this returns 0.

This value is given by the local_work_size argument to clEnqueueNDRangeKernel if local_work_size is not NULL; otherwise the OpenCL implementation chooses an appropriate local_work_size value which is returned by this function. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_local_size() returns 1. For clEnqueueTask, this always returns 1.

a work-item within a specific work-group for dimension identified by dimindx. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_local_id() returns 0. For clEnqueueTask, this returns 0.

Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_num_groups () returns 1. For clEnqueueTask, this always returns 1.

. get_num_groups(dimindx) – 1. Valid values of dimindx are 0 to get_work_dim() – 1. For other values, get_group↩ _id() returns 0. For clEnqueueTask, this returns 0.

Valid values of dimindx are 0 to get_work_dim() – 1. For other values, get_global_offset() returns 0. For cl↩ EnqueueTask, this returns 0.

Rounding of intermediate products shall not occur. Edge case behavior is per the IEEE 754-2008 standard.

If one argument is a NaN, fmax() returns the other argument. If both arguments are NaNs, fmax() returns a NaN.

If one argument is a NaN, fmin() returns the other argument. If both arguments are NaNs, fmin() returns a NaN.

Returns x – y ∗ trunc (x/y).

floor(x) is returned in iptr.

For each component the mantissa returned is a float with magnitude in the interval [1/2, 1) or 0. Each component of x equals mantissa returned ∗ 2^exp.

Returns the natural logarithm of the absolute value of the gamma function. The sign of the gamma function is returned in the signp argument of lgamma_r.

Whether or how the product of a ∗ b is rounded and how supernormal or subnormal intermediate products are handled is not defined. mad is intended to be used where speed is preferred over accuracy.

The modf function breaks the argument x into integral and fractional parts, each of which has the same sign as the argument. It stores the integral part in the object pointed to by iptr.

The nancode may be placed in the significand of the resulting NaN.

Thus, if y is less than x, nextafter() returns the largest representable floating-point number less than x.

If there are two integers closest to x/y, n shall be the even one. If r is zero, it is given the same sign as x.

If there are two integers closest to x/y, n shall be the even one. If r is zero, it is given the same sign as x. This is the same value that is returned by the remainder function. remquo also calculates the lower seven bits of the integral quotient x/y, and gives that value the same sign as x/y. It stores this signed value in the object pointed to by quo.

Refer to section 7.1 for description of rounding modes.

The computed sine is the return value and computed cosine is returned in cosval.

x must be in the range -2^16 ... +2^16.

x must be in the range -2^16 ... +2^16.

x must be in the range -216 ... +216.

The maximum error is implementation-defined.

The maximum error is implementation defined.

The maximum error is implementationdefined.

The range of x and y are implementation-defined. The maximum error is implementation-defined.

The intermediate sum does not modulo overflow.

Results are undefined if minval > maxval.

Bits shifted off the left side of the element are shifted back in from the right.

Refer to definition of mul24 to see how the 24-bit integer multiplication is performed.

x and y are 32-bit integers but only the low 24-bits are used to perform the multiplication. mul24 should only be used when values in x and y are in the range [- 2^23, 2^23-1] if x and y are signed integers and in the range [0, 2^24-1] if x and y are unsigned integers. If x and y are not in this range, the multiplication result is implementation-defined.

(180 / PI) ∗ radians.

If x and y are infinite or NaN, the return values are undefined.

If a is not in the range 0.0 ... 1.0, the return values are undefined.

(PI / 180) ∗ degrees.

This is useful in cases where you would want a threshold function with a smooth transition. This is equivalent to: gentype t; t = clamp ((x – edge0) / (edge1 – edge0), 0, 1); return t ∗ t ∗ (3 – 2 ∗ t); Results are undefined if edge0 >= edge1 or if x, edge0 or edge1 is a NaN.

Returns 0.0 if x is a NaN.

The w component of float4 result returned will be 0.0.

This is calculated as length(p0 – p1).

fast_normalize is computed as: p ∗ half_rsqrt (p.x^2 + p.y^2 + ...) The result shall be within 8192 ulps error from the infinitely precise result of if (all(p == 0.0f)) result = p; else result = p / sqrt (p.x^2 + p.y^2 + ...); with the following exceptions: 1) If the sum of squares is greater than FLT_MAX then the value of the floating-point values in the result vector are undefined. 2) If the sum of squares is less than FLT_MIN then the implementation may return back p. 3) If the device is in “denorms are flushed to zero” mode, individual operand elements with magnitude less than sqrt(FLT_MIN) may be flushed to zero before proceeding with the calculation.

isordered() takes arguments x and y, and returns the result isequal(x, x) && isequal(y, y).

isunordered() takes arguments x and y, returning non-zero if x or y is NaN, and zero otherwise.

The scalar version of the function returns a 1 if the sign bit in the float is set else returns 0. The vector version of the function returns the following for each component in floatn: a -1 if the sign bit in the float is set else returns 0.

Otherwise it is the corresponding bit of b.

For a scalar type, result = c ? b : a.

The address computed as (p + (offset ∗ n)) must be 8-bit aligned if gentype is char, uchar; 16-bit aligned if gentype is short, ushort; 32-bit aligned if gentype is int, uint, float; 64-bit aligned if gentype is long, ulong.

The data read is interpreted as a half value. The half value is converted to a float value and the float value is returned. The read address computed as (p + offset) must be 16-bit aligned.

The data read is interpreted as a halfn value. The halfn value read is converted to a floatn value and the floatn value is returned. The read address computed as (p + (offset ∗ n)) must be 16-bit aligned.

The half value is then written to address computed as (p + offset). The address computed as (p + offset) must be 16- bit aligned. vstore_half use the current rounding mode. The default current rounding mode is round to nearest even.

The halfn value is then written to address computed as (p + (offset ∗ n)). The address computed as (p + (offset ∗ n)) must be 16-bit aligned. vstore_halfn uses the current rounding mode. The default current rounding mode is round to nearest even.

The data read is interpreted as a halfn value. The halfn value read is converted to a floatn value and the floatn value is returned. The address computed as (p + (offset ∗ n)) must be aligned to sizeof (halfn) bytes. For n = 3, vloada_half3 reads a half3 from address (p + (offset ∗ 4)) and returns a float3. The address computed as (p + (offset ∗ 4)) must be aligned to sizeof (half) ∗ 4 bytes.

For n = 1, 2, 4, 8 and 16, the halfn value is written to the address computed as (p + (offset

- n)). The address computed as (p + (offset ∗ n)) must be aligned to sizeof (halfn) bytes. For n = 3, the half3 value is written to the address computed as (p + (offset ∗ 4)). The address computed as (p + (offset ∗ 4)) must be aligned to sizeof (half) ∗ 4 bytes. vstorea_halfn uses the current rounding mode. The default current rounding mode is round to nearest even.

  This means that loads and stores preceding the mem_fence will be committed to memory before any loads and stores following the mem_fence. The flags argument specifies the memory address space and can be set to a combination of the following literal values: CLK_LOCAL_MEM_FENCE CLK_GLOBAL_MEM_FENCE.

The flags argument specifies the memory address space and can be set to to a combination of the following literal values: CLK_LOCAL_MEM_FENCE CLK_GLOBAL_MEM_FENCE.

The async copy is performed by all work-items in a workgroup and this built-in function must therefore be encountered by all work-items in a workgroup executing the kernel with the same argument values; otherwise the results are undefined. Returns an event object that can be used by wait_group_events to wait for the async copy to finish. The event argument can also be used to associate the async_work_group_copy with a previous async copy allowing an event to be shared by multiple async copies; otherwise event should be zero. If event argument is non-zero, the event object supplied in event argument will be returned. This function does not perform any implicit synchronization of source data such as using a barrier before performing the copy.

The src_stride is the stride in elements for each gentype element read from src. The dst_stride is the stride in elements for each gentype element written to dst. The async gather is performed by all work-items in a work-group. This built-in function must therefore be encountered by all work-items in a work-group executing the kernel with the same argument values; otherwise the results are undefined. Returns an event object that can be used by wait_group_events to wait for the async copy to finish. The event argument can also be used to associate the async_work_group_strided_copy with a previous async copy allowing an event to be shared by multiple async copies; otherwise event should be zero. If event argument is non-zero, the event object supplied in event argument will be returned. This function does not perform any implicit synchronization of source data such as using a barrier before performing the copy.

The event objects specified in event_list will be released after the wait is performed. This function must be encountered by all workitems in a work-group executing the kernel with the same num_events and event objects specified in event_list; otherwise the results are undefined.

The prefetch instruction is applied to a work-item in a workgroup and does not affect the functional behavior of the kernel.

Compute (old + val) and store result at location pointed by p. The function returns old.

Compute (old - val) and store result at location pointed by p. The function returns old.

Returns old value.

Compute (old + 1) and store result at location pointed by p. The function returns old.

Compute (old - 1) and store result at location pointed by p. The function returns old.

Compute (old == cmp) ? val : old and store result at location pointed by p. The function returns old.

Compute min(old, val) and store minimum value at location pointed by p. The function returns old.

Compute max(old, val) and store maximum value at location pointed by p. The function returns old.

Compute (old & val) and store result at location pointed by p. The function returns old.

Compute (old | val) and store result at location pointed by p. The function returns old.

Compute (old ^ val) and store result at location pointed by p. The function returns old.

The size of each element in the mask must match the size of each element in the result. For shuffle, only the ilogb(2m-1) least significant bits of each mask element are considered. For shuffle2, only the ilogb(2m-1)+1 least significant bits of each mask element are considered. Other bits in the mask shall be ignored. The elements of the input vectors are numbered from left to right across one or both of the vectors. For this purpose, the number of elements in a vector is given by vec_step(gentypem). The shuffle mask operand specifies, for each element of the result vector, which element of the one or two input vectors the result element gets. Examples: uint4 mask = (uint4)(3, 2, 1, 0); float4 a; float4 r = shuffle(a, mask); // r.s0123 = a.wzyx uint8 mask = (uint8)(0, 1, 2, 3, 4, 5, 6, 7); float4 a, b; float8 r = shuffle2(a, b, mask); // r.s0123 = a.xyzw // r.s4567 = b.xyzw uint4 mask; float8 a; float4 b; b = shuffle(a, mask); Examples that are not valid are: uint8 mask; short16 a; short8 b; b = shuffle(a, mask); <- not valid

read_imagei and read_imageui return unnormalized signed integer and unsigned integer values respectively. Each channel will be stored in a 32-bit integer. read_imagei can only be used with image objects created with image_channel_data_type set to one of the following values: CL_SIGNED_INT8, CL_SIGNED_INT16 and CL_SIGNED_INT32. If the image_channel_data_type is not one of the above values, the values returned by read_imagei are undefined. read_imageui can only be used with image objects created with image_channel_data_type set to one of the following values: CL_UNSIGNED_INT8, CL_UNSIGNED_INT16 and CL_UNSIGNED_INT32. If the image_channel_data_type is not one of the above values, the values returned by read_imageui are undefined. The read_image{i|ui} calls support a nearest filter only. The filter_mode specified in sampler must be set to CLK_FILTER_NEAREST; otherwise the values returned are undefined. Furthermore, the read_image{i|ui} calls that take integer coordinates must use a sampler with normalized coordinates set to CLK_NORMALIZED_COORDS_FALSE and addressing mode set to CLK_ADDRESS_CLAMP_TO_EDGE, CLK_ADDRESS_CLAMP or CLK_ADDRESS_NONE; otherwise the values returned are undefined.

Appropriate data format conversion to the specified image format is done before writing the color value. x & y are considered to be unnormalized coordinates and must be in the range 0 ... image width - 1, and 0 ... image height - 1. write_imagef can only be used with image objects created with image_channel_data_type set to one of the pre-defined packed formats or set to CL_SNORM_INT8, CL_UNORM_INT8, CL_SNORM_INT16, CL_UNORM_INT16, CL_HALF_FLOAT or CL_FLOAT. Appropriate data format conversion will be done to convert channel data from a floating-point value to actual data format in which the channels are stored. write_imagei can only be used with image objects created with image_channel_data_type set to one of the following values: CL_SIGNED_INT8,

CL_SIGNED_INT16 and CL_SIGNED_INT32. write_imageui can only be used with image objects created with image_channel_data_type set to one of the following values: CL_UNSIGNED_INT8, CL_UNSIGNED_INT16 and CL_UNSIGNED_INT32. The behavior of write_imagef, write_imagei and write_imageui for image objects created with image_channel_data_type values not specified in the description above or with (x, y) coordinate values that are not in the range (0 ... image width - 1, 0 ... image height - 1), respectively, is undefined.

coord.w is ignored. read_imagef returns floating-point values in the range [0.0 ... 1.0] for image objects created with image_channel_data_type set to one of the predefined packed formats or CL_UNORM_INT8, or CL_UNORM_↩ INT16. read_imagef returns floating-point values in the range [-1.0 ... 1.0] for image objects created with image↩ _channel_data_type set to CL_SNORM_INT8, or CL_SNORM_INT16. read_imagef returns floating-point values for image objects created with image_channel_data_type set to CL_HALF_FLOAT or CL_FLOAT. The read_imagef calls that take integer coordinates must use a sampler with filter mode set to CLK_FILTER_NEAREST, normalized coordinates set to CLK_NORMALIZED_COORDS_FALSE and addressing mode set to CLK_ADDRESS_CLAMP↩ _TO_EDGE, CLK_ADDRESS_CLAMP or CLK_ADDRESS_NONE; otherwise the values returned are undefined. Values returned by read_imagef for image objects with image_channel_data_type values not specified in the description are undefined.

coord.w is ignored. read_imagei and read_imageui return unnormalized signed integer and unsigned integer values respectively. Each channel will be stored in a 32-bit integer. read_imagei can only be used with image objects created with image_channel_data_type set to one of the following values: CL_SIGNED_INT8, CL_SIGNED_INT16 and CL_SIGNED_INT32. If the image_channel_data_type is not one of the above values, the values returned by read_imagei are undefined. read_imageui can only be used with image objects created with image_channel_↩ data_type set to one of the following values: CL_UNSIGNED_INT8, CL_UNSIGNED_INT16 and CL_UNSIGNED↩ _INT32. If the image_channel_data_type is not one of the above values, the values returned by read_imageui are undefined. The read_image{i|ui} calls support a nearest filter only. The filter_mode specified in sampler must be set to CLK_FILTER_NEAREST; otherwise the values returned are undefined. Furthermore, the read_image{i|ui} calls that take integer coordinates must use a sampler with normalized coordinates set to CLK_NORMALIZED_↩ COORDS_FALSE and addressing mode set to CLK_ADDRESS_CLAMP_TO_EDGE, CLK_ADDRESS_CLAMP or CLK_ADDRESS_NONE; otherwise the values returned are undefined.

Valid values are: CLK_A CLK_R CLK_Rx CLK_RG CLK_RGx CLK_RA CLK_RGB CLK_RGBx CLK_RGBA CLK↩ _ARGB CLK_BGRA CLK_INTENSITY CLK_LUMINANCE

The width is returned in the x component, and the height in the y component.

The width is returned in the x component, height in the y component, depth in the z component and the w component is 0.

Definition at line 77 of file opencl_spir.h.

13.83.2.2 cl_mem_fence_flags

```
typedef uint cl_mem_fence_flags
```

All work-items in a work-group executing the kernel on a processor must execute this function before any are allowed to continue execution beyond the barrier.

This function must be encountered by all work-items in a work-group executing the kernel. If barrier is inside a conditional statement, then all work-items must enter the conditional if any work-item enters the conditional statement and executes the barrier. If barrer is inside a loop, all work-items must execute the barrier for each iteration of the loop before any are allowed to continue execution beyond the barrier. The barrier function also queues a memory fence (reads and writes) to ensure correct ordering of memory operations to local or global memory. The flags argument specifies the memory address space and can be set to a combination of the following literal values. CLK_LOCAL_MEM_FENCE - The barrier function will either flush any variables stored in local memory or queue a memory fence to ensure correct ordering of memory operations to local memory. CLK_GLOBAL_MEM_FENCE – The barrier function will queue a memory fence to ensure correct ordering of memory operations to global memory. This can be useful when work-items, for example, write to buffer or image objects and then want to read the updated data.

Definition at line 5774 of file opencl_spir.h.

13.83.2.3 intptr_t

```
typedef ptrdiff_t intptr_t
```

The unsigned integer type of the result of the sizeof operator.

This is a 32-bit unsigned integer if CL_DEVICE_ADDRESS_BITS defined in table 4.3 is 32-bits and is a 64-bit unsigned integer if CL_DEVICE_ADDRESS_BITS is 64-bits. A signed integer type that is the result of subtracting two pointers. This is a 32-bit signed integer if CL_DEVICE_ADDRESS_BITS defined in table 4.3 is 32-bits and is a 64-bit signed integer if CL_DEVICE_ADDRESS_BITS is 64-bits.

Definition at line 70 of file opencl_spir.h.

13.83.2.4 uchar

```
typedef unsigned char uchar
```

An unsigned 8-bit integer.

Definition at line 29 of file opencl_spir.h.

13.83.2.5 uint

```
typedef unsigned int uint
```

An unsigned 32-bit integer.

Definition at line 39 of file opencl_spir.h.

13.83.2.6 uintptr_t

```
typedef size_t uintptr_t
```

Definition at line 72 of file opencl_spir.h.

13.83.2.7 ulong

```
typedef unsigned long ulong
```

An unsigned 64-bit integer.

Definition at line 44 of file opencl_spir.h.

13.83.2.8 ushort

```
typedef unsigned short ushort
```

An unsigned 16-bit integer.

Definition at line 34 of file opencl_spir.h.

13.83.3 Function Documentation

13.83.3.1 __attribute__()

```
uint const_func __attribute__ (
             (overloadable)  )
```

Returns the number of dimensions in use.

Write color value to location specified by coord.xy in the 2D image layer identified by index coord.z in the 2D image array.

Use coord.xy to do an element lookup in the 2D image layer identified by index coord.z in the 2D image array.

Return the number of images in the 1D image array.

Return the number of images in the 2D image array.

Return the 3D image width, height, and depth as an int4 type.

Return the 2D image width and height as an int2 type.

Return the image channel order.

Return the image depth in pixels.

Return the image height in pixels.

Return the image width in pixels.

Write color value to location specified by coordinate (x, y, z) in the 3D image object specified by image.

Use the coordinate (coord.x, coord.y, coord.z) to do an element lookup in the 3D image object specified by image.

Write color value to location specified by coordinate (x, y) in the 2D image object specified by image.

Use the coordinate (x, y) to do an element lookup in the 2D image object specified by image.

The shuffle and shuffle2 built-in functions construct a permutation of elements from one or two input vectors respectively that are of the same type, returning a vector with the same element type as the input and length that is the same as the shuffle mask.

Swaps the old value stored at location p with new value given by val.

unsigned int atomic_sub (volatile __global unsigned int *p, unsigned int val) int atomic_sub (volatile __local int *p, int val) unsigned int atomic_sub (volatile __local unsigned int *p, unsigned int val) Read the 32-bit value (referred to as old) stored at location pointed by p.

Read the 32-bit value (referred to as old) stored at location pointed by p.

Prefetch num_elements * sizeof(gentype) bytes into the global cache.

Wait for events that identify the async_work_group_copy operations to complete.

Perform an async gather of num_elements gentype elements from src to dst.

event_t async_work_group_copy (__global gentype *dst, const __local gentype *src, size_t num_elements, event_t event) Perform an async copy of num_elements gentype elements from src to dst.

Write memory barrier that orders only stores.

Read memory barrier that orders only loads.

Orders loads and stores of a work-item executing a kernel.

For n = 1, 2, 4, 8 and 16 read sizeof (halfn) bytes of data from address (p + (offset * n)).

The floatn value given by data is converted to a halfn value using the appropriate rounding mode.

The float value given by data is first converted to a half value using the appropriate rounding mode.

Read sizeof (halfn) bytes of data from address (p + (offset * n)).

Read sizeof (half) bytes of data from address (p + offset).

Write sizeof (gentypen) bytes given by data to address (p + (offset * n)).

Return sizeof (gentypen) bytes of data read from address (p + (offset * n)).

For each component of a vector type, result[i] = if MSB of c[i] is set ? b[i] : a[i].

Each bit of the result is the corresponding bit of a if the corresponding bit of c is 0.

Returns 1 if the most significant bit in all components of x is set; otherwise returns 0.

Returns 1 if the most significant bit in any component of x is set; otherwise returns 0.

Test for sign bit.

Test if arguments are unordered.

Test if arguments are ordered.

Test for a normal value.

Test for a NaN.

Test for infinity value (+ve or –ve) .

Test for finite value.

Returns the component-wise compare of (x < y) || (x > y) .

Returns the component-wise compare of x <= y.

Returns the component-wise compare of x $<$ y.

Returns the component-wise compare of x $>=$ y.

Returns the component-wise compare of x $>$ y.

Returns the component-wise compare of x != y.

intn isequal (floatn x, floatn y) Returns the component-wise compare of x == y.

Returns the length of vector p computed as: half_sqrt(p.x2 + p.y2 + ...)

Returns fast_length(p0 – p1).

Returns a vector in the same direction as p but with a length of 1.

Return the length of vector p, i.e., sqrt(p.x2 + p.y 2 + ...)

Returns the distance between p0 and p1.

Compute dot product.

Returns the cross product of p0.xyz and p1.xyz.

Returns 1.0 if x $>$ 0, -0.0 if x = -0.0, +0.0 if x = +0.0, or –1.0 if x $<$ 0.

Returns 0.0 if x $<=$ edge0 and 1.0 if x $>=$ edge1 and performs smooth Hermite interpolation between 0 and 1when edge0 $<$ x $<$ edge1.

Returns 0.0 if x $<$ edge, otherwise it returns 1.0.

Converts degrees to radians, i.e.

Returns the linear blend of x & y implemented as: x + (y – x) $*$ a a must be a value in the range 0.0 . . . 1.0.

Converts radians to degrees, i.e.

Returns fmin(fmax(x, minval), maxval).

Multiply two 24-bit integer values x and y.

Multiply two 24-bit integer values x and y and add the 32-bit integer result to the 32-bit integer z.

result[i] = ((long)hi[i] $<<$ 32) | lo[i] result[i] = ((ulong)hi[i] $<<$ 32) | lo[i]

result[i] = ((int)hi[i] $<<$ 16) | lo[i] result[i] = ((uint)hi[i] $<<$ 16) | lo[i]

result[i] = ((short)hi[i] $<<$ 8) | lo[i] result[i] = ((ushort)hi[i] $<<$ 8) | lo[i]

Returns x - y and saturates the result.

For each element in v, the bits are shifted left by the number of bits given by the corresponding element in i (subject to usual shift modulo rules described in section 6.3).

Computes x $*$ y and returns the high half of the product of x and y.

Returns a $*$ b + c and saturates the result.

Returns mul_hi(a, b) + c.

Returns the number of leading 0-bits in x, starting at the most significant bit position.

Returns min(max(x, minval), maxval).

Returns (x + y + 1) $>>$ 1.

Returns (x + y) $>>$ 1.

Returns x + y and saturates the result.

Returns $|$ x – y $|$ without modulo overflow.

Returns $|$ x $|$.

Compute tangent over an implementation-defined range.

Compute square root over an implementation-defined range.

Compute sine over an implementation-defined range.

Compute inverse square root over an implementationdefined range.

Compute reciprocal over an implementation-defined range.

Compute a base 10 logarithm over an implementationdefined range.

Compute a base 2 logarithm over an implementationdefined range.

Compute natural logarithm over an implementationdefined range.

Compute the base- 10 exponential of x over an implementation-defined range.

Compute the base- 2 exponential of x over an implementation-defined range.

Compute the base- e exponential of x over an implementation-defined range.

Compute x / y over an implementation-defined range.

Compute cosine over an implementation-defined range.

Compute reciprocal.

Compute the base- 10 exponential of x.

Compute the base- 2 exponential of x.

Compute x / y.

Round to integral value using the round to zero rounding mode.

Compute the gamma function.

Compute tan (PI $*$ x).

Compute hyperbolic tangent.

Compute tangent.

Compute square root.

Compute sin (PI $*$ x).

Compute hyperbolic sine.

Compute sine and cosine of x.

Compute sine.

Compute inverse square root.

Return the integral value nearest to x rounding halfway cases away from zero, regardless of the current rounding direction.

Compute x to the power 1/y.

Round to integral value (using round to nearest even rounding mode) in floating-point format.

The remquo function computes the value r such that r = x - n$*$y, where n is the integer nearest the exact value of x/y.

Compute the value r such that r = x - n$*$y, where n is the integer nearest the exact value of x/y.

Compute x to the power y, where x is $>=$ 0.

Compute x to the power y, where y is an integer.

Compute x to the power y.

Computes the next representable single-precision floating-point value following x in the direction of y.

Returns a quiet NaN.

Decompose a floating-point number.

Returns x if $\mid$ x $\mid$ $<$ $\mid$ y $\mid$, y if $\mid$ y $\mid$ $<$ $\mid$ x $\mid$, otherwise fmin(x, y).

Returns x if $\mid$ x $\mid$ $>$ $\mid$ y $\mid$, y if $\mid$ y $\mid$ $>$ $\mid$ x $\mid$, otherwise fmax(x, y).

mad approximates a $*$ b + c.

Compute the exponent of x, which is the integral part of logr $\mid$ x $\mid$.

Compute a base e logarithm of (1.0 + x).

Compute a base 10 logarithm.

Compute a base 2 logarithm.

Compute natural logarithm.

Log gamma function.

Multiply x by 2 to the power n.

Return the exponent as an integer value.

Compute the value of the square root of x^2+ y^2 without undue overflow or underflow.

Extract mantissa and exponent from x.

Returns fmin(x – floor (x), 0x1.fffffep-1f).

Modulus.

Returns y if y $<$ x, otherwise it returns x.

Returns y if x $<$ y, otherwise it returns x.

Returns the correctly rounded floating-point representation of the sum of c with the infinitely precise product of a and b.

Round to integral value using the round to –ve infinity rounding mode.

x - y if x $>$ y, +0 if x is less than or equal to y.

Compute absolute value of a floating-point number.

Compute e^x- 1.0.

Exponential base 10 function.

Exponential base 2 function.

Compute the base- e exponential of x.

Error function encountered in integrating the normal distribution.

Complementary error function.

Compute cos (PI $*$ x).

Compute hyperbolic consine.

Compute cosine.

Returns x with its sign changed to match the sign of y.

Round to integral value using the round to +ve infinity rounding mode.

Compute cube-root.

Compute atan2 (y, x) / PI.

Compute atan (x) / PI.

Hyperbolic arc tangent.

Arc tangent of y / x.

Arc tangent function.

Compute asin (x) / PI.

Inverse hyperbolic sine.

Arc sine function.

Compute acos (x) / PI.

Inverse hyperbolic cosine.

Arc cosine function.

get_global_offset returns the offset values specified in global_work_offset argument to clEnqueueNDRangeKernel.

get_group_id returns the work-group ID which is a number from 0 .

Returns the number of work-groups that will execute a kernel for dimension identified by dimindx.

Returns the unique local work-item ID i.e.

Returns the number of local work-items specified in dimension identified by dimindx.

Returns the unique global work-item ID value for dimension identified by dimindx.

Returns the number of global work-items specified for dimension identified by dimindx.

This is the value given to the work_dim argument specified in clEnqueueNDRangeKernel. For clEnqueueTask, this returns 1.

This value is given by the global_work_size argument to clEnqueueNDRangeKernel. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_global_size() returns 1. For clEnqueueTask, this always returns 1.

The global work-item ID specifies the work-item ID based on the number of global work-items specified to execute the kernel. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_global_id() returns 0. For clEnqueueTask, this returns 0.

This value is given by the local_work_size argument to clEnqueueNDRangeKernel if local_work_size is not NULL; otherwise the OpenCL implementation chooses an appropriate local_work_size value which is returned by this function. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_local_size() returns 1. For clEnqueueTask, this always returns 1.

a work-item within a specific work-group for dimension identified by dimindx. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_local_id() returns 0. For clEnqueueTask, this returns 0.

Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_num_groups () returns 1. For clEnqueueTask, this always returns 1.

. get_num_groups(dimindx) – 1. Valid values of dimindx are 0 to get_work_dim() – 1. For other values, get_group↩ _id() returns 0. For clEnqueueTask, this returns 0.

Valid values of dimindx are 0 to get_work_dim() – 1. For other values, get_global_offset() returns 0. For cl↩ EnqueueTask, this returns 0.

Rounding of intermediate products shall not occur. Edge case behavior is per the IEEE 754-2008 standard.

If one argument is a NaN, fmax() returns the other argument. If both arguments are NaNs, fmax() returns a NaN.

If one argument is a NaN, fmin() returns the other argument. If both arguments are NaNs, fmin() returns a NaN.

Returns x – y ∗ trunc (x/y).

floor(x) is returned in iptr.

For each component the mantissa returned is a float with magnitude in the interval [1/2, 1) or 0. Each component of x equals mantissa returned ∗ 2^exp.

Returns the natural logarithm of the absolute value of the gamma function. The sign of the gamma function is returned in the signp argument of lgamma_r.

Whether or how the product of a $*$ b is rounded and how supernormal or subnormal intermediate products are handled is not defined. mad is intended to be used where speed is preferred over accuracy.

The modf function breaks the argument x into integral and fractional parts, each of which has the same sign as the argument. It stores the integral part in the object pointed to by iptr.

The nancode may be placed in the significand of the resulting NaN.

Thus, if y is less than x, nextafter() returns the largest representable floating-point number less than x.

If there are two integers closest to x/y, n shall be the even one. If r is zero, it is given the same sign as x.

If there are two integers closest to x/y, n shall be the even one. If r is zero, it is given the same sign as x. This is the same value that is returned by the remainder function. remquo also calculates the lower seven bits of the integral quotient x/y, and gives that value the same sign as x/y. It stores this signed value in the object pointed to by quo.

Refer to section 7.1 for description of rounding modes.

The computed sine is the return value and computed cosine is returned in cosval.

x must be in the range -2^16 ... +2^16.

x must be in the range -2^16 ... +2^16.

x must be in the range -216 ... +216.

The maximum error is implementation-defined.

The maximum error is implementation defined.

The maximum error is implementationdefined.

The range of x and y are implementation-defined. The maximum error is implementation-defined.

The intermediate sum does not modulo overflow.

Results are undefined if minval $>$ maxval.

Bits shifted off the left side of the element are shifted back in from the right.

Refer to definition of mul24 to see how the 24-bit integer multiplication is performed.

x and y are 32-bit integers but only the low 24-bits are used to perform the multiplication. mul24 should only be used when values in x and y are in the range [- 2^23, 2^23-1] if x and y are signed integers and in the range [0, 2^24-1] if x and y are unsigned integers. If x and y are not in this range, the multiplication result is implementation-defined.

(180 / PI) $*$ radians.

If x and y are infinite or NaN, the return values are undefined.

If a is not in the range 0.0 ... 1.0, the return values are undefined.

(PI / 180) $*$ degrees.

This is useful in cases where you would want a threshold function with a smooth transition. This is equivalent to: gentype t; t = clamp ((x – edge0) / (edge1 – edge0), 0, 1); return t $*$ t $*$ (3 – 2 $*$ t); Results are undefined if edge0 $>=$ edge1 or if x, edge0 or edge1 is a NaN.

Returns 0.0 if x is a NaN.

The w component of float4 result returned will be 0.0.

This is calculated as length(p0 – p1).

fast_normalize is computed as: p ∗ half_rsqrt (p.x^2 + p.y^2 + . . .) The result shall be within 8192 ulps error from the infinitely precise result of if (all(p == 0.0f)) result = p; else result = p / sqrt (p.x^2 + p.y^2 + ...); with the following exceptions: 1) If the sum of squares is greater than FLT_MAX then the value of the floating-point values in the result vector are undefined. 2) If the sum of squares is less than FLT_MIN then the implementation may return back p. 3) If the device is in "denorms are flushed to zero" mode, individual operand elements with magnitude less than sqrt(FLT_MIN) may be flushed to zero before proceeding with the calculation.

isordered() takes arguments x and y, and returns the result isequal(x, x) && isequal(y, y).

isunordered() takes arguments x and y, returning non-zero if x or y is NaN, and zero otherwise.

The scalar version of the function returns a 1 if the sign bit in the float is set else returns 0. The vector version of the function returns the following for each component in floatn: a -1 if the sign bit in the float is set else returns 0.

Otherwise it is the corresponding bit of b.

For a scalar type, result = c ? b : a.

The address computed as (p + (offset ∗ n)) must be 8-bit aligned if gentype is char, uchar; 16-bit aligned if gentype is short, ushort; 32-bit aligned if gentype is int, uint, float; 64-bit aligned if gentype is long, ulong.

The data read is interpreted as a half value. The half value is converted to a float value and the float value is returned. The read address computed as (p + offset) must be 16-bit aligned.

The data read is interpreted as a halfn value. The halfn value read is converted to a floatn value and the floatn value is returned. The read address computed as (p + (offset ∗ n)) must be 16-bit aligned.

The half value is then written to address computed as (p + offset). The address computed as (p + offset) must be 16- bit aligned. vstore_half use the current rounding mode. The default current rounding mode is round to nearest even.

The halfn value is then written to address computed as (p + (offset ∗ n)). The address computed as (p + (offset ∗ n)) must be 16-bit aligned. vstore_halfn uses the current rounding mode. The default current rounding mode is round to nearest even.

The data read is interpreted as a halfn value. The halfn value read is converted to a floatn value and the floatn value is returned. The address computed as (p + (offset ∗ n)) must be aligned to sizeof (halfn) bytes. For n = 3, vloada_half3 reads a half3 from address (p + (offset ∗ 4)) and returns a float3. The address computed as (p + (offset ∗ 4)) must be aligned to sizeof (half) ∗ 4 bytes.

For n = 1, 2, 4, 8 and 16, the halfn value is written to the address computed as (p + (offset

- n)). The address computed as (p + (offset ∗ n)) must be aligned to sizeof (halfn) bytes. For n = 3, the half3 value is written to the address computed as (p + (offset ∗ 4)). The address computed as (p + (offset ∗ 4)) must be aligned to sizeof (half) ∗ 4 bytes. vstorea_halfn uses the current rounding mode. The default current rounding mode is round to nearest even.

  This means that loads and stores preceding the mem_fence will be committed to memory before any loads and stores following the mem_fence. The flags argument specifies the memory address space and can be set to a combination of the following literal values: CLK_LOCAL_MEM_FENCE CLK_GLOBAL_MEM_FENCE.

The flags argument specifies the memory address space and can be set to to a combination of the following literal values: CLK_LOCAL_MEM_FENCE CLK_GLOBAL_MEM_FENCE.

The async copy is performed by all work-items in a workgroup and this built-in function must therefore be encountered by all work-items in a workgroup executing the kernel with the same argument values; otherwise the results are undefined. Returns an event object that can be used by wait_group_events to wait for the async copy to finish. The event argument can also be used to associate the async_work_group_copy with a previous async copy allowing an event to be shared by multiple async copies; otherwise event should be zero. If event argument is non-zero, the event object supplied in event argument will be returned. This function does not perform any implicit synchronization of source data such as using a barrier before performing the copy.

The src_stride is the stride in elements for each gentype element read from src. The dst_stride is the stride in elements for each gentype element written to dst. The async gather is performed by all work-items in a work-group. This built-in function must therefore be encountered by all work-items in a work-group executing the kernel with the same argument values; otherwise the results are undefined. Returns an event object that can be used by wait_group_events to wait for the async copy to finish. The event argument can also be used to associate the async_work_group_strided_copy with a previous async copy allowing an event to be shared by multiple async copies; otherwise event should be zero. If event argument is non-zero, the event object supplied in event argument will be returned. This function does not perform any implicit synchronization of source data such as using a barrier before performing the copy.

The event objects specified in event_list will be released after the wait is performed. This function must be encountered by all workitems in a work-group executing the kernel with the same num_events and event objects specified in event_list; otherwise the results are undefined.

The prefetch instruction is applied to a work-item in a workgroup and does not affect the functional behavior of the kernel.

Compute (old + val) and store result at location pointed by p. The function returns old.

Compute (old - val) and store result at location pointed by p. The function returns old.

Returns old value.

Compute (old + 1) and store result at location pointed by p. The function returns old.

Compute (old - 1) and store result at location pointed by p. The function returns old.

Compute (old == cmp) ? val : old and store result at location pointed by p. The function returns old.

Compute min(old, val) and store minimum value at location pointed by p. The function returns old.

Compute max(old, val) and store maximum value at location pointed by p. The function returns old.

Compute (old & val) and store result at location pointed by p. The function returns old.

Compute (old | val) and store result at location pointed by p. The function returns old.

Compute (old ^ val) and store result at location pointed by p. The function returns old.

The size of each element in the mask must match the size of each element in the result. For shuffle, only the ilogb(2m-1) least significant bits of each mask element are considered. For shuffle2, only the ilogb(2m-1)+1 least significant bits of each mask element are considered. Other bits in the mask shall be ignored. The elements of the input vectors are numbered from left to right across one or both of the vectors. For this purpose, the number of elements in a vector is given by vec_step(gentypem). The shuffle mask operand specifies, for each element of the result vector, which element of the one or two input vectors the result element gets. Examples: uint4 mask = (uint4)(3, 2, 1, 0); float4 a; float4 r = shuffle(a, mask); // r.s0123 = a.wzyx uint8 mask = (uint8)(0, 1, 2, 3, 4, 5, 6, 7); float4 a, b; float8 r = shuffle2(a, b, mask); // r.s0123 = a.xyzw // r.s4567 = b.xyzw uint4 mask; float8 a; float4 b; b = shuffle(a, mask); Examples that are not valid are: uint8 mask; short16 a; short8 b; b = shuffle(a, mask); <- not valid

read_imagei and read_imageui return unnormalized signed integer and unsigned integer values respectively. Each channel will be stored in a 32-bit integer. read_imagei can only be used with image objects created with image_channel_data_type set to one of the following values: CL_SIGNED_INT8, CL_SIGNED_INT16 and CL_SIGNED_INT32. If the image_channel_data_type is not one of the above values, the values returned by read_imagei are undefined. read_imageui can only be used with image objects created with image_channel_data_type set to one of the following values: CL_UNSIGNED_INT8, CL_UNSIGNED_INT16 and CL_UNSIGNED_INT32. If the image_channel_data_type is not one of the above values, the values returned by read_imageui are undefined. The read_image{i|ui} calls support a nearest filter only. The filter_mode specified in sampler must be set to CLK_FILTER_NEAREST; otherwise the values returned are undefined. Furthermore, the read_image{i|ui} calls that take integer coordinates must use a sampler with normalized coordinates set to CLK_NORMALIZED_COORDS_FALSE and addressing mode set to CLK_ADDRESS_CLAMP_TO_EDGE, CLK_ADDRESS_CLAMP or CLK_ADDRESS_NONE; otherwise the values returned are undefined.

Appropriate data format conversion to the specified image format is done before writing the color value. x & y are considered to be unnormalized coordinates and must be in the range 0 ... image width - 1, and 0 ... image height - 1. write_imagef can only be used with image objects created with image_channel_data_type set to one of the pre-defined packed formats or set to CL_SNORM_INT8, CL_UNORM_INT8, CL_SNORM_INT16, CL_UNORM_INT16, CL_HALF_FLOAT or CL_FLOAT. Appropriate data format conversion will be done to convert channel data from a floating-point value to actual data format in which the channels are stored. write_imagei can only be used with image objects created with image_channel_data_type set to one of the following values: CL_SIGNED_INT8, CL_SIGNED_INT16 and CL_SIGNED_INT32. write_imageui can only be used with image objects created with image_channel_data_type set to one of the following values: CL_UNSIGNED_INT8, CL_UNSIGNED_INT16 and CL_UNSIGNED_INT32. The behavior of write_imagef, write_imagei and write_imageui for image objects created with image_channel_data_type values not specified in the description above or with (x, y) coordinate values that are not in the range (0 ... image width - 1, 0 ... image height - 1), respectively, is undefined.

coord.w is ignored. read_imagef returns floating-point values in the range [0.0 ... 1.0] for image objects created with image_channel_data_type set to one of the predefined packed formats or CL_UNORM_INT8, or CL_UNORM_INT16. read_imagef returns floating-point values in the range [-1.0 ... 1.0] for image objects created with image_channel_data_type set to CL_SNORM_INT8, or CL_SNORM_INT16. read_imagef returns floating-point values for image objects created with image_channel_data_type set to CL_HALF_FLOAT or CL_FLOAT. The read_imagef calls that take integer coordinates must use a sampler with filter mode set to CLK_FILTER_NEAREST, normalized coordinates set to CLK_NORMALIZED_COORDS_FALSE and addressing mode set to CLK_ADDRESS_CLAMP_TO_EDGE, CLK_ADDRESS_CLAMP or CLK_ADDRESS_NONE; otherwise the values returned are undefined. Values returned by read_imagef for image objects with image_channel_data_type values not specified in the description are undefined.

coord.w is ignored. read_imagei and read_imageui return unnormalized signed integer and unsigned integer values respectively. Each channel will be stored in a 32-bit integer. read_imagei can only be used with image objects created with image_channel_data_type set to one of the following values: CL_SIGNED_INT8, CL_SIGNED_INT16 and CL_SIGNED_INT32. If the image_channel_data_type is not one of the above values, the values returned by read_imagei are undefined. read_imageui can only be used with image objects created with image_channel_data_type set to one of the following values: CL_UNSIGNED_INT8, CL_UNSIGNED_INT16 and CL_UNSIGNED_INT32. If the image_channel_data_type is not one of the above values, the values returned by read_imageui are undefined. The read_image{i|ui} calls support a nearest filter only. The filter_mode specified in sampler must be set to CLK_FILTER_NEAREST; otherwise the values returned are undefined. Furthermore, the read_image{i|ui} calls that take integer coordinates must use a sampler with normalized coordinates set to CLK_NORMALIZED_COORDS_FALSE and addressing mode set to CLK_ADDRESS_CLAMP_TO_EDGE, CLK_ADDRESS_CLAMP or CLK_ADDRESS_NONE; otherwise the values returned are undefined.

Valid values are: CLK_A CLK_R CLK_Rx CLK_RG CLK_RGx CLK_RA CLK_RGB CLK_RGBx CLK_RGBA CLK_ARGB CLK_BGRA CLK_INTENSITY CLK_LUMINANCE

The width is returned in the x component, and the height in the y component.

The width is returned in the x component, height in the y component, depth in the z component and the w component is 0.

13.83.3.2 printf()

```
int printf (
             __constant const char * st,
              ... )
```

13.83.4 Variable Documentation

13.83.4.1 a

```
double16 const_func double16 double a
```

Definition at line 3613 of file opencl_spir.h.

Referenced by trisycl::drt::accessor< Accessor >::accessor(), trisycl::get_pipe_detail(), trisycl::detail::task::register_accessor(), trisycl::mixin::accessor< T, Dimensions >::set_access(), and trisycl::vendor::xilinx::graphics::application::start().

13.83.4.2 b

```
double16 const_func double16 b
```

Definition at line 915 of file opencl_spir.h.

Referenced by trisycl::detail::add_buffer_to_task(), std::hash< trisycl::buffer< T, Dimensions, Allocator > >::operator()(), trisycl::detail::task::release_buffers(), trisycl::detail::accessor< T, Dimensions, Mode, Target >::set_buffer(), and trisycl::detail::waiter().

13.83.4.3 c

```
double16 const_func double16 ulong16 c
```

Definition at line 915 of file opencl_spir.h.

Referenced by trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::display(), trisycl::vendor::xilinx::acap::aie::detail::tile trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::display(), trisycl::vendor::xilinx::acap::aie::axi_stream_ std::hash< trisycl::context >::operator()(), and trisycl::detail::opencl_context::~opencl_context().

13.83.4.4 cmp

```
unsigned int unsigned int cmp
```

Definition at line 6359 of file opencl_spir.h.

13.83.4.5 color

```
void int4 uint4 color
```

Definition at line 7138 of file opencl_spir.h.

Referenced by trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::display_routing_configuration(), and trisycl::vendor::xilinx::graphics::palette::rainbow_palette().

13.83.4.6 coord

```
void int4 coord
```

Definition at line 7027 of file opencl_spir.h.

Referenced by trisycl::vendor::xilinx::acap::aie::geography< Layout >::generate_tiles().

13.83.4.7 cosval

```
double16 __private double16 * cosval
```

Definition at line 1892 of file opencl_spir.h.

13.83.4.8 dst_stride

```
event_t const __local half16 size_t size_t dst_stride
```

Definition at line 6074 of file opencl_spir.h.

13.83.4.9 edge1

```
double16 const_func double16 edge1
```

Definition at line 3722 of file opencl_spir.h.

13.83.4.10 event

```
event_t const __local half16 size_t size_t event_t event
```

Definition at line 5863 of file opencl_spir.h.

13.83.4.11 event_list

```
void event_t* event_list
```

Definition at line 6162 of file opencl_spir.h.

13.83.4.12 exp

```
double16 __private int16 * exp
```

Definition at line 1102 of file opencl_spir.h.

13.83.4.13 i

```
ulong16 const_func ulong16 i
```

Definition at line 3222 of file opencl_spir.h.

Referenced by trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::device(), trisycl::vendor::xilinx::acap::aie::detail::tile_infrastru
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::display(), trisycl::vendor::xilinx::acap::aie::axi_stream_
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::display_routing_configuration(),
trisycl::detail::fiber_pool::fiber_pool(), trisycl::detail::vec< DataType, NumElements >::flatten(), trisycl::detail::vec< DataType, NumEl
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_neighborhood(), trisycl::detail::cache< Key, Value >::get_or_
trisycl::mixin::accessor< T, Dimensions >::get_range(), trisycl::vendor::xilinx::graphics::palette::gray_palette(),
trisycl::vendor::xilinx::acap::aie::lock_unit::lock(), trisycl::vendor::xilinx::acap::aie::memory_base::lock(), trisycl::vendor::xilinx::acap::ai
main(), trisycl::detail::spir::make_array_with_func(), trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::operator
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >::parallel_for(), trisycl::detail::parallel_for_
trisycl::vendor::xilinx::graphics::palette::rainbow_palette(), trisycl::detail::sycl_2_2::pipe< T >::reserve_write(),
trisycl::detail::fiber_pool::run(), trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::start(),
trisycl::mixin::accessor< T, Dimensions >::tuple_indexed_mdspan_access(), and trisycl::vendor::xilinx::graphics::image_grid::update

13.83.4.14 iptr

```
double16 __private double16 * iptr
```

Definition at line 1040 of file opencl_spir.h.

13.83.4.15 lo

```
ulong16 const_func uint16 lo
```

Definition at line 3327 of file opencl_spir.h.

13.83.4.16 mask

```
double16 const_func double16 ulong16 mask
```

Definition at line 6535 of file opencl_spir.h.

13.83.4.17 maxval

```
double16 const_func double double maxval
```

Definition at line 2707 of file opencl_spir.h.

13.83.4.18 minval

```
double16 const_func double minval
```

Definition at line 2707 of file opencl_spir.h.

13.83.4.19 n

```
double16 const_func int n
```

Definition at line 1205 of file opencl_spir.h.

Referenced by trisycl::detail::vec< DataType, NumElements >::convert().

13.83.4.20 num_elements

```
void size_t num_elements
```

Definition at line 5863 of file opencl_spir.h.

13.83.4.21 offset

```
void size_t offset
```

Definition at line 5062 of file opencl_spir.h.

Referenced by trisycl::handler::parallel_for(), and trisycl::detail::parallel_for_global_offset().

13.83.4.22 p

```
void size_t __private half * p
```

Definition at line 4827 of file opencl_spir.h.

Referenced by trisycl::detail::address_space_ptr< T, AS >::address_space_ptr(), trisycl::vendor::xilinx::acap::aie::detail::tile_infrastru
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::display_routing_configuration(),
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input(), trisycl::vendor::xilinx::acap::aie::shim_tile< AI
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::input(), trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure<
main(), std::hash< trisycl::platform >::operator()(), trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::ou
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::output(), trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::output(), trisycl::vendor::trisycl::scope::platform< PlatformS
trisycl::detail::task::postlude(), trisycl::detail::task::prelude(), trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >::set_program
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::shim_tile(), trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStream
trisycl::detail::fiber_pool::submit(), trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >::tile_infrastructure(),
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >::try_read(), trisycl::vendor::xilinx::acap::aie::axi_stream_switch
trisycl::vendor::xilinx::acap::aie::fifo_channel::try_read(), trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >::try_read
and trisycl::detail::opencl_platform::~opencl_platform().

13.83.4.23 p1

```
float const_func float4 p1
```

Definition at line 3787 of file opencl_spir.h.

13.83.4.24 quo

```
double16 double16 __private int16 * quo
```

Definition at line 1717 of file opencl_spir.h.

13.83.4.25 sampler

```
uint4 const_func sampler_t sampler
```

Definition at line 7027 of file opencl_spir.h.

13.83.4.26 signp

```
double16 __private int16 * signp
```

Definition at line 1265 of file opencl_spir.h.

13.83.4.27 src

```
event_t const __local half16 * src
```

Definition at line 5863 of file opencl_spir.h.

Referenced by trisycl::vendor::xilinx::acap::aie::device< Layout >::connect(), trisycl::vendor::xilinx::acap::aie::detail::device< Layout >
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >::partition_array(), and trisycl::detail::small_array< BasicType

13.83.4.28 src_stride

```
event_t const __global half16 size_t size_t src_stride
```

Definition at line 6020 of file opencl_spir.h.

13.83.4.29 val

```
unsigned int unsigned int val
```

Definition at line 6245 of file opencl_spir.h.

13.83.4.30 x

```
double16 const_func double16 double16 x
```

Definition at line 489 of file opencl_spir.h.

Referenced by trisycl::detail::vec< DataType, NumElements >::apply_binary_functor(), trisycl::detail::vec< DataType, NumElements
trisycl::vendor::xilinx::acap::aie::geography< Layout >::cascade_linear_id(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::c
trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >::check(), trisycl::cross(), trisycl::vendor::xilinx::acap::aie::detail::devic
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::for_each_memory_base(), trisycl::vendor::xilinx::acap::aie::de
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >::for_each_tile_base(), trisycl::vendor::xilinx::acap::aie::detail::c
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::for_each_tile_neighborhood(), trisycl::vendor::xilinx::acap::aie::detail::devic
trisycl::vendor::xilinx::graphics::frame_grid::frame_grid(), trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::get_casca
trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >::get_cascade_stream_out(), trisycl::vendor::xilinx::graphics::frame_
trisycl::vendor::xilinx::graphics::image_grid::image_grid(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_cascade_end(),
trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_cascade_start(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is
trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_memory_module(), trisycl::vendor::xilinx::acap::aie::layout::full::is_noc_tile(
trisycl::vendor::xilinx::acap::aie::layout::full::is_pl_tile(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_west_column(),
trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_valid(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::is_x_y_v
trisycl::vendor::xilinx::acap::aie::geography< Layout >::linear_id(), main(), trisycl::max(), trisycl::vendor::xilinx::acap::aie::device< Lay
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::mem(), trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memor
trisycl::vendor::xilinx::acap::aie::geography< Layout >::memory_module_linear_id(), trisycl::normalize(), trisycl::vendor::xilinx::acap::
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::shim(), trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >::start(),
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::start(), trisycl::vendor::xilinx::acap::aie::device< Layou
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >::tile(), trisycl::TRISYCL_MATH_WRAP2s(), trisycl::vendor::xilinx::graphics::i
trisycl::vendor::xilinx::graphics::application::update_tile_data_image(), trisycl::vendor::xilinx::acap::aie::geography< Layout >::validat
trisycl::vendor::xilinx::acap::aie::geography< Layout >::validate_x_y(), and trisycl::detail::vec< DataType, NumElements >::zip().

13.83.4.31 y

```
double16 const_func double16 y
```

Definition at line 623 of file opencl_spir.h.

Referenced by main().

13.83.4.32 z

```
uint16 const_func uint16 uint16 z
```

Definition at line 3431 of file opencl_spir.h.

13.84 opencl_spir.h

Go to the documentation of this file.

```
00001 //
00002 //                     SPIR Tools
00003 //
00004 // This file is distributed under the University of Illinois Open Source
00005 // License.  See LICENSE.TXT for details.
00006 //
00007
00008 #ifndef _OPENCL_H_
00009 #define _OPENCL_H_
00010
00011 #if !defined(__SPIR32__) && !defined(__SPIR64__)
00012 #error "This header file should be used with SPIR target only."
00013 #endif
00014
00015 /// Added according to pocl/examples/example1-spir64/generate_spir.sh
00016 /// \todo avoid this hack
00017 #pragma OPENCL EXTENSION cl_khr_fp64 :  enable
00018
00019 // Optimizations
00020
00021 #define const_func __attribute__((const))
00022 #define readonly __attribute__((pure))
00023
00024 // built-in scalar data types:
00025
00026 /**
00027 * An unsigned 8-bit integer.
00028 */
00029 typedef unsigned char uchar;
00030
00031 /**
00032 * An unsigned 16-bit integer.
00033 */
00034 typedef unsigned short ushort;
00035
00036 /**
00037 * An unsigned 32-bit integer.
00038 */
00039 typedef unsigned int uint;
00040
00041 /**
00042 * An unsigned 64-bit integer.
00043 */
00044 typedef unsigned long ulong;
00045
00046 /**
00047 * The unsigned integer type of the result of the sizeof operator.  This
00048 * is a 32-bit unsigned integer if CL_DEVICE_ADDRESS_BITS
00049 * defined in table 4.3 is 32-bits and is a 64-bit unsigned integer if
00050 * CL_DEVICE_ADDRESS_BITS is 64-bits.
00051 */
00052 #if defined(__SPIR32__)
```

```
00053 typedef uint size_t;
00054 #elif defined (__SPIR64__)
00055 typedef ulong size_t;
00056 #endif
00057
00058 /**
00059 * A signed integer type that is the result of subtracting two pointers.
00060 * This is a 32-bit signed integer if CL_DEVICE_ADDRESS_BITS
00061 * defined in table 4.3 is 32-bits and is a 64-bit signed integer if
00062 * CL_DEVICE_ADDRESS_BITS is 64-bits.
00063 */
00064 #if defined(__SPIR32__)
00065 typedef int ptrdiff_t;
00066 #elif defined (__SPIR64__ )
00067 typedef long ptrdiff_t;
00068 #endif
00069
00070 typedef ptrdiff_t intptr_t;
00071
00072 typedef size_t uintptr_t;
00073
00074
00075 // built-in vector data types:
00076
00077 typedef char char2 __attribute__((ext_vector_type(2)));
00078 typedef char char3 __attribute__((ext_vector_type(3)));
00079 typedef char char4 __attribute__((ext_vector_type(4)));
00080 typedef char char8 __attribute__((ext_vector_type(8)));
00081 typedef char char16 __attribute__((ext_vector_type(16)));
00082 typedef uchar uchar2 __attribute__((ext_vector_type(2)));
00083 typedef uchar uchar3 __attribute__((ext_vector_type(3)));
00084 typedef uchar uchar4 __attribute__((ext_vector_type(4)));
00085 typedef uchar uchar8 __attribute__((ext_vector_type(8)));
00086 typedef uchar uchar16 __attribute__((ext_vector_type(16)));
00087 typedef short short2 __attribute__((ext_vector_type(2)));
00088 typedef short short3 __attribute__((ext_vector_type(3)));
00089 typedef short short4 __attribute__((ext_vector_type(4)));
00090 typedef short short8 __attribute__((ext_vector_type(8)));
00091 typedef short short16 __attribute__((ext_vector_type(16)));
00092 typedef ushort ushort2 __attribute__((ext_vector_type(2)));
00093 typedef ushort ushort3 __attribute__((ext_vector_type(3)));
00094 typedef ushort ushort4 __attribute__((ext_vector_type(4)));
00095 typedef ushort ushort8 __attribute__((ext_vector_type(8)));
00096 typedef ushort ushort16 __attribute__((ext_vector_type(16)));
00097 typedef int int2 __attribute__((ext_vector_type(2)));
00098 typedef int int3 __attribute__((ext_vector_type(3)));
00099 typedef int int4 __attribute__((ext_vector_type(4)));
00100 typedef int int8 __attribute__((ext_vector_type(8)));
00101 typedef int int16 __attribute__((ext_vector_type(16)));
00102 typedef uint uint2 __attribute__((ext_vector_type(2)));
00103 typedef uint uint3 __attribute__((ext_vector_type(3)));
00104 typedef uint uint4 __attribute__((ext_vector_type(4)));
00105 typedef uint uint8 __attribute__((ext_vector_type(8)));
00106 typedef uint uint16 __attribute__((ext_vector_type(16)));
00107 typedef long long2 __attribute__((ext_vector_type(2)));
00108 typedef long long3 __attribute__((ext_vector_type(3)));
00109 typedef long long4 __attribute__((ext_vector_type(4)));
00110 typedef long long8 __attribute__((ext_vector_type(8)));
00111 typedef long long16 __attribute__((ext_vector_type(16)));
00112 typedef ulong ulong2 __attribute__((ext_vector_type(2)));
00113 typedef ulong ulong3 __attribute__((ext_vector_type(3)));
00114 typedef ulong ulong4 __attribute__((ext_vector_type(4)));
00115 typedef ulong ulong8 __attribute__((ext_vector_type(8)));
00116 typedef ulong ulong16 __attribute__((ext_vector_type(16)));
00117 typedef float float2 __attribute__((ext_vector_type(2)));
00118 typedef float float3 __attribute__((ext_vector_type(3)));
00119 typedef float float4 __attribute__((ext_vector_type(4)));
00120 typedef float float8 __attribute__((ext_vector_type(8)));
00121 typedef float float16 __attribute__((ext_vector_type(16)));
00122 typedef half half2 __attribute__((ext_vector_type(2)));
00123 typedef half half3 __attribute__((ext_vector_type(3)));
00124 typedef half half4 __attribute__((ext_vector_type(4)));
00125 typedef half half8 __attribute__((ext_vector_type(8)));
00126 typedef half half16 __attribute__((ext_vector_type(16)));
00127 typedef double double2 __attribute__((ext_vector_type(2)));
00128 typedef double double3 __attribute__((ext_vector_type(3)));
00129 typedef double double4 __attribute__((ext_vector_type(4)));
00130 typedef double double8 __attribute__((ext_vector_type(8)));
00131 typedef double double16 __attribute__((ext_vector_type(16)));
00132
00133 /**
00134 * Value of maximum non-infinite single-precision floating-point
00135 * number.
00136 */
00137 #define MAXFLOAT    0x1.fffffep127f
00138
00139 /**
```

```
00140 * A positive float constant expression.  HUGE_VALF evaluates
00141 * to +infinity.  Used as an error value returned by the built-in
00142 * math functions.
00143 */
00144 #define HUGE_VALF (__builtin_huge_valf())
00145
00146 /**
00147 * A positive double constant expression.  HUGE_VAL evaluates
00148 * to +infinity.  Used as an error value returned by the built-in
00149 * math functions.
00150 */
00151 #define HUGE_VAL (__builtin_huge_val())
00152
00153 /**
00154 * A constant expression of type float representing positive or
00155 * unsigned infinity.
00156 */
00157 #define INFINITY (__builtin_inff())
00158
00159 /**
00160 * A constant expression of type float representing a quiet NaN.
00161 */
00162 #define NAN as_float(__builtin_nanf((char const *)""))
00163
00164 #define FP_ILOGB0        INT_MIN
00165 #define FP_ILOGBNAN      INT_MAX
00166
00167 #define FLT_DIG 6
00168 #define FLT_MANT_DIG 24
00169 #define FLT_MAX_10_EXP +38
00170 #define FLT_MAX_EXP +128
00171 #define FLT_MIN_10_EXP -37
00172 #define FLT_MIN_EXP -125
00173 #define FLT_RADIX 2
00174 #define FLT_MAX 0x1.fffffep127f
00175 #define FLT_MIN 0x1.0p-126f
00176 #define FLT_EPSILON 0x1.0p-23f
00177
00178 #define DBL_DIG 15
00179 #define DBL_MANT_DIG 53
00180 #define DBL_MAX_10_EXP +308
00181 #define DBL_MAX_EXP +1024
00182 #define DBL_MIN_10_EXP -307
00183 #define DBL_MIN_EXP -1021
00184 #define DBL_RADIX 2
00185 #define DBL_MAX 0x1.fffffffffffffp1023
00186 #define DBL_MIN 0x1.0p-1022
00187 #define DBL_EPSILON 0x1.0p-52
00188
00189
00190 #define M_E_F         2.71828182845904523536028747135266250f
00191 #define M_LOG2E_F     1.44269504088896340735992468100189214f
00192 #define M_LOG10E_F    0.43429448190325182765112891891660508 2f
00193 #define M_LN2_F       0.69314718055994530941723212145817656 8f
00194 #define M_LN10_F      2.30258509299404568401799145468436421f
00195 #define M_PI_F        3.14159265358979323846264338327950288f
00196 #define M_PI_2_F      1.57079632679489661923132169163975144f
00197 #define M_PI_4_F      0.78539816339744830961566084581987572 1f
00198 #define M_1_PI_F      0.31830988618379067153776752674502872 4f
00199 #define M_2_PI_F      0.63661977236758134307553505349005744 8f
00200 #define M_2_SQRTPI_F  1.12837916709551257389615890312154517f
00201 #define M_SQRT2_F     1.41421356237309504880168872420969808f
00202 #define M_SQRT1_2_F   0.70710678118654752440084436210484903 9f
00203
00204 #define M_E             0x1.5bf0a8b145769p+1
00205 #define M_LOG2E         0x1.71547652b82fep+0
00206 #define M_LOG10E        0x1.bcb7b1526e50ep-2
00207 #define M_LN2           0x1.62e42fefa39efp-1
00208 #define M_LN10          0x1.26bb1bbb55516p+1
00209 #define M_PI            0x1.921fb54442d18p+1
00210 #define M_PI_2          0x1.921fb54442d18p+0
00211 #define M_PI_4          0x1.921fb54442d18p-1
00212 #define M_1_PI          0x1.45f306dc9c883p-2
00213 #define M_2_PI          0x1.45f306dc9c883p-1
00214 #define M_2_SQRTPI      0x1.20dd750429b6dp+0
00215 #define M_SQRT2         0x1.6a09e667f3bcdp+0
00216 #define M_SQRT1_2       0x1.6a09e667f3bcdp-1
00217
00218 #define CHAR_BIT    8
00219 #define SCHAR_MAX   127
00220 #define SCHAR_MIN   (-128)
00221 #define UCHAR_MAX   255
00222 #define CHAR_MAX    SCHAR_MAX
00223 #define CHAR_MIN    SCHAR_MIN
00224 #define USHRT_MAX   65535
00225 #define SHRT_MAX    32767
00226 #define SHRT_MIN    (-32768)
```

```
00227 #define UINT_MAX    0xffffffff
00228 #define INT_MAX     2147483647
00229 #define INT_MIN     (-2147483647-1)
00230 #define ULONG_MAX   0xffffffffffffffffUL
00231 #define LONG_MAX    ((long)0x7fffffffffffffffL)
00232 #define LONG_MIN    ((long)(-0x7fffffffffffffffL-1))
00233
00234 #define DBL_DIG 15
00235 #define DBL_MANT_DIG 53
00236 #define DBL_MAX_10_EXP +308
00237 #define DBL_MAX_EXP +1024
00238 #define DBL_MIN_10_EXP -307
00239 #define DBL_MIN_EXP -1021
00240 #define DBL_MAX 0x1.fffffffffffffp1023
00241 #define DBL_MIN 0x1.0p-1022
00242 #define DBL_EPSILON 0x1.0p-52
00243
00244 // work-item functions
00245
00246 /**
00247 * Returns the number of dimensions in use.  This is the
00248 * value given to the work_dim argument specified in
00249 * clEnqueueNDRangeKernel.
00250 * For clEnqueueTask, this returns 1.
00251 */
00252 uint const_func __attribute__((overloadable)) get_work_dim(void);
00253
00254 /**
00255 * Returns the number of global work-items specified for
00256 * dimension identified by dimindx.  This value is given by
00257 * the global_work_size argument to
00258 * clEnqueueNDRangeKernel.  Valid values of dimindx
00259 * are 0 to get_work_dim() - 1.  For other values of
00260 * dimindx, get_global_size() returns 1.
00261 * For clEnqueueTask, this always returns 1.
00262 */
00263 size_t const_func __attribute__((overloadable)) get_global_size(uint dimindx);
00264
00265 /**
00266 * Returns the unique global work-item ID value for
00267 * dimension identified by dimindx.  The global work-item
00268 * ID specifies the work-item ID based on the number of
00269 * global work-items specified to execute the kernel.  Valid
00270 * values of dimindx are 0 to get_work_dim() - 1.  For
00271 * other values of dimindx, get_global_id() returns 0.
00272 * For clEnqueueTask, this returns 0.
00273 */
00274 size_t const_func __attribute__((overloadable)) get_global_id(uint dimindx);
00275
00276 /**
00277 * Returns the number of local work-items specified in
00278 * dimension identified by dimindx.  This value is given by
00279 * the local_work_size argument to
00280 * clEnqueueNDRangeKernel if local_work_size is not
00281 * NULL; otherwise the OpenCL implementation chooses
00282 * an appropriate local_work_size value which is returned
00283 * by this function.  Valid values of dimindx are 0 to
00284 * get_work_dim() - 1.  For other values of dimindx,
00285 * get_local_size() returns 1.
00286 * For clEnqueueTask, this always returns 1.
00287 */
00288 size_t const_func __attribute__((overloadable)) get_local_size(uint dimindx);
00289
00290 /**
00291 * Returns the unique local work-item ID i.e.  a work-item
00292 * within a specific work-group for dimension identified by
00293 * dimindx.  Valid values of dimindx are 0 to
00294 * get_work_dim() - 1.  For other values of dimindx,
00295 * get_local_id() returns 0.
00296 * For clEnqueueTask, this returns 0.
00297 */
00298 size_t const_func __attribute__((overloadable)) get_local_id(uint dimindx);
00299
00300 /**
00301 * Returns the number of work-groups that will execute a
00302 * kernel for dimension identified by dimindx.
00303 * Valid values of dimindx are 0 to get_work_dim() - 1.
00304 * For other values of dimindx, get_num_groups () returns
00305 * 1.
00306 * For clEnqueueTask, this always returns 1.
00307 */
00308 size_t const_func __attribute__((overloadable)) get_num_groups(uint dimindx);
00309
00310 /**
00311 * get_group_id returns the work-group ID which is a
00312 * number from 0 ..  get_num_groups(dimindx) - 1.
00313 * Valid values of dimindx are 0 to get_work_dim() - 1.
```

```
00314 * For other values, get_group_id() returns 0.
00315 * For clEnqueueTask, this returns 0.
00316 */
00317 size_t const_func __attribute__((overloadable)) get_group_id(uint dimindx);
00318
00319 /**
00320 * get_global_offset returns the offset values specified in
00321 * global_work_offset argument to
00322 * clEnqueueNDRangeKernel.
00323 * Valid values of dimindx are 0 to get_work_dim() - 1.
00324 * For other values, get_global_offset() returns 0.
00325 * For clEnqueueTask, this returns 0.
00326 */
00327 size_t const_func __attribute__((overloadable)) get_global_offset(uint dimindx);
00328
00329 int printf(__constant const char* st, ...);
00330 // Math functions:
00331
00332 /**
00333 * Arc cosine function.
00334 */
00335 float const_func __attribute__((overloadable)) acos(float);
00336 float2 const_func __attribute__((overloadable)) acos(float2);
00337 float3 const_func __attribute__((overloadable)) acos(float3);
00338 float4 const_func __attribute__((overloadable)) acos(float4);
00339 float8 const_func __attribute__((overloadable)) acos(float8);
00340 float16 const_func __attribute__((overloadable)) acos(float16);
00341 double const_func __attribute__((overloadable)) acos(double);
00342 double2 const_func __attribute__((overloadable)) acos(double2);
00343 double3 const_func __attribute__((overloadable)) acos(double3);
00344 double4 const_func __attribute__((overloadable)) acos(double4);
00345 double8 const_func __attribute__((overloadable)) acos(double8);
00346 double16 const_func __attribute__((overloadable)) acos(double16);
00347 //half const_func __attribute__((overloadable)) acos(half);
00348 //half2 const_func __attribute__((overloadable)) acos(half2);
00349 //half3 const_func __attribute__((overloadable)) acos(half3);
00350 //half4 const_func __attribute__((overloadable)) acos(half4);
00351 //half8 const_func __attribute__((overloadable)) acos(half8);
00352 //half16 const_func __attribute__((overloadable)) acos(half16);
00353
00354 /**
00355 * Inverse hyperbolic cosine.
00356 */
00357 float const_func __attribute__((overloadable)) acosh(float);
00358 float2 const_func __attribute__((overloadable)) acosh(float2);
00359 float3 const_func __attribute__((overloadable)) acosh(float3);
00360 float4 const_func __attribute__((overloadable)) acosh(float4);
00361 float8 const_func __attribute__((overloadable)) acosh(float8);
00362 float16 const_func __attribute__((overloadable)) acosh(float16);
00363 double const_func __attribute__((overloadable)) acosh(double);
00364 double2 const_func __attribute__((overloadable)) acosh(double2);
00365 double3 const_func __attribute__((overloadable)) acosh(double3);
00366 double4 const_func __attribute__((overloadable)) acosh(double4);
00367 double8 const_func __attribute__((overloadable)) acosh(double8);
00368 double16 const_func __attribute__((overloadable)) acosh(double16);
00369 //half const_func __attribute__((overloadable)) acosh(half);
00370 //half2 const_func __attribute__((overloadable)) acosh(half2);
00371 //half3 const_func __attribute__((overloadable)) acosh(half3);
00372 //half4 const_func __attribute__((overloadable)) acosh(half4);
00373 //half8 const_func __attribute__((overloadable)) acosh(half8);
00374 //half16 const_func __attribute__((overloadable)) acosh(half16);
00375
00376 /**
00377 * Compute acos (x) / PI.
00378 */
00379 float const_func __attribute__((overloadable)) acospi(float x);
00380 float2 const_func __attribute__((overloadable)) acospi(float2 x);
00381 float3 const_func __attribute__((overloadable)) acospi(float3 x);
00382 float4 const_func __attribute__((overloadable)) acospi(float4 x);
00383 float8 const_func __attribute__((overloadable)) acospi(float8 x);
00384 float16 const_func __attribute__((overloadable)) acospi(float16 x);
00385 double const_func __attribute__((overloadable)) acospi(double x);
00386 double2 const_func __attribute__((overloadable)) acospi(double2 x);
00387 double3 const_func __attribute__((overloadable)) acospi(double3 x);
00388 double4 const_func __attribute__((overloadable)) acospi(double4 x);
00389 double8 const_func __attribute__((overloadable)) acospi(double8 x);
00390 double16 const_func __attribute__((overloadable)) acospi(double16 x);
00391 //half const_func __attribute__((overloadable)) acospi(half x);
00392 //half2 const_func __attribute__((overloadable)) acospi(half2 x);
00393 //half3 const_func __attribute__((overloadable)) acospi(half3 x);
00394 //half4 const_func __attribute__((overloadable)) acospi(half4 x);
00395 //half8 const_func __attribute__((overloadable)) acospi(half8 x);
00396 //half16 const_func __attribute__((overloadable)) acospi(half16 x);
00397
00398 /**
00399 * Arc sine function.
00400 */
```

```
00401 float const_func __attribute__((overloadable)) asin(float);
00402 float2 const_func __attribute__((overloadable)) asin(float2);
00403 float3 const_func __attribute__((overloadable)) asin(float3);
00404 float4 const_func __attribute__((overloadable)) asin(float4);
00405 float8 const_func __attribute__((overloadable)) asin(float8);
00406 float16 const_func __attribute__((overloadable)) asin(float16);
00407 double const_func __attribute__((overloadable)) asin(double);
00408 double2 const_func __attribute__((overloadable)) asin(double2);
00409 double3 const_func __attribute__((overloadable)) asin(double3);
00410 double4 const_func __attribute__((overloadable)) asin(double4);
00411 double8 const_func __attribute__((overloadable)) asin(double8);
00412 double16 const_func __attribute__((overloadable)) asin(double16);
00413 //half const_func __attribute__((overloadable)) asin(half);
00414 //half2 const_func __attribute__((overloadable)) asin(half2);
00415 //half3 const_func __attribute__((overloadable)) asin(half3);
00416 //half4 const_func __attribute__((overloadable)) asin(half4);
00417 //half8 const_func __attribute__((overloadable)) asin(half8);
00418 //half16 const_func __attribute__((overloadable)) asin(half16);
00419
00420 /**
00421 * Inverse hyperbolic sine.
00422 */
00423 float const_func __attribute__((overloadable)) asinh(float);
00424 float2 const_func __attribute__((overloadable)) asinh(float2);
00425 float3 const_func __attribute__((overloadable)) asinh(float3);
00426 float4 const_func __attribute__((overloadable)) asinh(float4);
00427 float8 const_func __attribute__((overloadable)) asinh(float8);
00428 float16 const_func __attribute__((overloadable)) asinh(float16);
00429 double const_func __attribute__((overloadable)) asinh(double);
00430 double2 const_func __attribute__((overloadable)) asinh(double2);
00431 double3 const_func __attribute__((overloadable)) asinh(double3);
00432 double4 const_func __attribute__((overloadable)) asinh(double4);
00433 double8 const_func __attribute__((overloadable)) asinh(double8);
00434 double16 const_func __attribute__((overloadable)) asinh(double16);
00435 //half const_func __attribute__((overloadable)) asinh(half);
00436 //half2 const_func __attribute__((overloadable)) asinh(half2);
00437 //half3 const_func __attribute__((overloadable)) asinh(half3);
00438 //half4 const_func __attribute__((overloadable)) asinh(half4);
00439 //half8 const_func __attribute__((overloadable)) asinh(half8);
00440 //half16 const_func __attribute__((overloadable)) asinh(half16);
00441
00442 /**
00443 * Compute asin (x) / PI.
00444 */
00445 float const_func __attribute__((overloadable)) asinpi(float x);
00446 float2 const_func __attribute__((overloadable)) asinpi(float2 x);
00447 float3 const_func __attribute__((overloadable)) asinpi(float3 x);
00448 float4 const_func __attribute__((overloadable)) asinpi(float4 x);
00449 float8 const_func __attribute__((overloadable)) asinpi(float8 x);
00450 float16 const_func __attribute__((overloadable)) asinpi(float16 x);
00451 double const_func __attribute__((overloadable)) asinpi(double x);
00452 double2 const_func __attribute__((overloadable)) asinpi(double2 x);
00453 double3 const_func __attribute__((overloadable)) asinpi(double3 x);
00454 double4 const_func __attribute__((overloadable)) asinpi(double4 x);
00455 double8 const_func __attribute__((overloadable)) asinpi(double8 x);
00456 double16 const_func __attribute__((overloadable)) asinpi(double16 x);
00457 //half const_func __attribute__((overloadable)) asinpi(half x);
00458 //half2 const_func __attribute__((overloadable)) asinpi(half2 x);
00459 //half3 const_func __attribute__((overloadable)) asinpi(half3 x);
00460 //half4 const_func __attribute__((overloadable)) asinpi(half4 x);
00461 //half8 const_func __attribute__((overloadable)) asinpi(half8 x);
00462 //half16 const_func __attribute__((overloadable)) asinpi(half16 x);
00463
00464 /**
00465 * Arc tangent function.
00466 */
00467 float const_func __attribute__((overloadable)) atan(float y_over_x);
00468 float2 const_func __attribute__((overloadable)) atan(float2 y_over_x);
00469 float3 const_func __attribute__((overloadable)) atan(float3 y_over_x);
00470 float4 const_func __attribute__((overloadable)) atan(float4 y_over_x);
00471 float8 const_func __attribute__((overloadable)) atan(float8 y_over_x);
00472 float16 const_func __attribute__((overloadable)) atan(float16 y_over_x);
00473 double const_func __attribute__((overloadable)) atan(double y_over_x);
00474 double2 const_func __attribute__((overloadable)) atan(double2 y_over_x);
00475 double3 const_func __attribute__((overloadable)) atan(double3 y_over_x);
00476 double4 const_func __attribute__((overloadable)) atan(double4 y_over_x);
00477 double8 const_func __attribute__((overloadable)) atan(double8 y_over_x);
00478 double16 const_func __attribute__((overloadable)) atan(double16 y_over_x);
00479 //half const_func __attribute__((overloadable)) atan(half y_over_x);
00480 //half2 const_func __attribute__((overloadable)) atan(half2 y_over_x);
00481 //half3 const_func __attribute__((overloadable)) atan(half3 y_over_x);
00482 //half4 const_func __attribute__((overloadable)) atan(half4 y_over_x);
00483 //half8 const_func __attribute__((overloadable)) atan(half8 y_over_x);
00484 //half16 const_func __attribute__((overloadable)) atan(half16 y_over_x);
00485
00486 /**
00487 * Arc tangent of y / x.
```

```
00488 */
00489 float const_func __attribute__((overloadable)) atan2(float y, float x);
00490 float2 const_func __attribute__((overloadable)) atan2(float2 y, float2 x);
00491 float3 const_func __attribute__((overloadable)) atan2(float3 y, float3 x);
00492 float4 const_func __attribute__((overloadable)) atan2(float4 y, float4 x);
00493 float8 const_func __attribute__((overloadable)) atan2(float8 y, float8 x);
00494 float16 const_func __attribute__((overloadable)) atan2(float16 y, float16 x);
00495 double const_func __attribute__((overloadable)) atan2(double y, double x);
00496 double2 const_func __attribute__((overloadable)) atan2(double2 y, double2 x);
00497 double3 const_func __attribute__((overloadable)) atan2(double3 y, double3 x);
00498 double4 const_func __attribute__((overloadable)) atan2(double4 y, double4 x);
00499 double8 const_func __attribute__((overloadable)) atan2(double8 y, double8 x);
00500 double16 const_func __attribute__((overloadable)) atan2(double16 y, double16 x);
00501 //half const_func __attribute__((overloadable)) atan2(half y, half x);
00502 //half2 const_func __attribute__((overloadable)) atan2(half2 y, half2 x);
00503 //half3 const_func __attribute__((overloadable)) atan2(half3 y, half3 x);
00504 //half4 const_func __attribute__((overloadable)) atan2(half4 y, half4 x);
00505 //half8 const_func __attribute__((overloadable)) atan2(half8 y, half8 x);
00506 //half16 const_func __attribute__((overloadable)) atan2(half16 y, half16 x);
00507
00508 /**
00509 * Hyperbolic arc tangent.
00510 */
00511 float const_func __attribute__((overloadable)) atanh(float);
00512 float2 const_func __attribute__((overloadable)) atanh(float2);
00513 float3 const_func __attribute__((overloadable)) atanh(float3);
00514 float4 const_func __attribute__((overloadable)) atanh(float4);
00515 float8 const_func __attribute__((overloadable)) atanh(float8);
00516 float16 const_func __attribute__((overloadable)) atanh(float16);
00517 double const_func __attribute__((overloadable)) atanh(double);
00518 double2 const_func __attribute__((overloadable)) atanh(double2);
00519 double3 const_func __attribute__((overloadable)) atanh(double3);
00520 double4 const_func __attribute__((overloadable)) atanh(double4);
00521 double8 const_func __attribute__((overloadable)) atanh(double8);
00522 double16 const_func __attribute__((overloadable)) atanh(double16);
00523 //half const_func __attribute__((overloadable)) atanh(half);
00524 //half2 const_func __attribute__((overloadable)) atanh(half2);
00525 //half3 const_func __attribute__((overloadable)) atanh(half3);
00526 //half4 const_func __attribute__((overloadable)) atanh(half4);
00527 //half8 const_func __attribute__((overloadable)) atanh(half8);
00528 //half16 const_func __attribute__((overloadable)) atanh(half16);
00529
00530 /**
00531 * Compute atan (x) / PI.
00532 */
00533 float const_func __attribute__((overloadable)) atanpi(float x);
00534 float2 const_func __attribute__((overloadable)) atanpi(float2 x);
00535 float3 const_func __attribute__((overloadable)) atanpi(float3 x);
00536 float4 const_func __attribute__((overloadable)) atanpi(float4 x);
00537 float8 const_func __attribute__((overloadable)) atanpi(float8 x);
00538 float16 const_func __attribute__((overloadable)) atanpi(float16 x);
00539 double const_func __attribute__((overloadable)) atanpi(double x);
00540 double2 const_func __attribute__((overloadable)) atanpi(double2 x);
00541 double3 const_func __attribute__((overloadable)) atanpi(double3 x);
00542 double4 const_func __attribute__((overloadable)) atanpi(double4 x);
00543 double8 const_func __attribute__((overloadable)) atanpi(double8 x);
00544 double16 const_func __attribute__((overloadable)) atanpi(double16 x);
00545 //half const_func __attribute__((overloadable)) atanpi(half x);
00546 //half2 const_func __attribute__((overloadable)) atanpi(half2 x);
00547 //half3 const_func __attribute__((overloadable)) atanpi(half3 x);
00548 //half4 const_func __attribute__((overloadable)) atanpi(half4 x);
00549 //half8 const_func __attribute__((overloadable)) atanpi(half8 x);
00550 //half16 const_func __attribute__((overloadable)) atanpi(half16 x);
00551
00552 /**
00553 * Compute atan2 (y, x) / PI.
00554 */
00555 float const_func __attribute__((overloadable)) atan2pi(float y, float x);
00556 float2 const_func __attribute__((overloadable)) atan2pi(float2 y, float2 x);
00557 float3 const_func __attribute__((overloadable)) atan2pi(float3 y, float3 x);
00558 float4 const_func __attribute__((overloadable)) atan2pi(float4 y, float4 x);
00559 float8 const_func __attribute__((overloadable)) atan2pi(float8 y, float8 x);
00560 float16 const_func __attribute__((overloadable)) atan2pi(float16 y, float16 x);
00561 double const_func __attribute__((overloadable)) atan2pi(double y, double x);
00562 double2 const_func __attribute__((overloadable)) atan2pi(double2 y, double2 x);
00563 double3 const_func __attribute__((overloadable)) atan2pi(double3 y, double3 x);
00564 double4 const_func __attribute__((overloadable)) atan2pi(double4 y, double4 x);
00565 double8 const_func __attribute__((overloadable)) atan2pi(double8 y, double8 x);
00566 double16 const_func __attribute__((overloadable)) atan2pi(double16 y, double16 x);
00567 //half const_func __attribute__((overloadable)) atan2pi(half y, half x);
00568 //half2 const_func __attribute__((overloadable)) atan2pi(half2 y, half2 x);
00569 //half3 const_func __attribute__((overloadable)) atan2pi(half3 y, half3 x);
00570 //half4 const_func __attribute__((overloadable)) atan2pi(half4 y, half4 x);
00571 //half8 const_func __attribute__((overloadable)) atan2pi(half8 y, half8 x);
00572 //half16 const_func __attribute__((overloadable)) atan2pi(half16 y, half16 x);
00573
00574 /**
```

```
00575 * Compute cube-root.
00576 */
00577 float const_func __attribute__((overloadable)) cbrt(float);
00578 float2 const_func __attribute__((overloadable)) cbrt(float2);
00579 float3 const_func __attribute__((overloadable)) cbrt(float3);
00580 float4 const_func __attribute__((overloadable)) cbrt(float4);
00581 float8 const_func __attribute__((overloadable)) cbrt(float8);
00582 float16 const_func __attribute__((overloadable)) cbrt(float16);
00583 double const_func __attribute__((overloadable)) cbrt(double);
00584 double2 const_func __attribute__((overloadable)) cbrt(double2);
00585 double3 const_func __attribute__((overloadable)) cbrt(double3);
00586 double4 const_func __attribute__((overloadable)) cbrt(double4);
00587 double8 const_func __attribute__((overloadable)) cbrt(double8);
00588 double16 const_func __attribute__((overloadable)) cbrt(double16);
00589 //half const_func __attribute__((overloadable)) cbrt(half);
00590 //half2 const_func __attribute__((overloadable)) cbrt(half2);
00591 //half3 const_func __attribute__((overloadable)) cbrt(half3);
00592 //half4 const_func __attribute__((overloadable)) cbrt(half4);
00593 //half8 const_func __attribute__((overloadable)) cbrt(half8);
00594 //half16 const_func __attribute__((overloadable)) cbrt(half16);
00595
00596 /**
00597 * Round to integral value using the round to +ve
00598 * infinity rounding mode.
00599 */
00600 float const_func __attribute__((overloadable)) ceil(float);
00601 float2 const_func __attribute__((overloadable)) ceil(float2);
00602 float3 const_func __attribute__((overloadable)) ceil(float3);
00603 float4 const_func __attribute__((overloadable)) ceil(float4);
00604 float8 const_func __attribute__((overloadable)) ceil(float8);
00605 float16 const_func __attribute__((overloadable)) ceil(float16);
00606 double const_func __attribute__((overloadable)) ceil(double);
00607 double2 const_func __attribute__((overloadable)) ceil(double2);
00608 double3 const_func __attribute__((overloadable)) ceil(double3);
00609 double4 const_func __attribute__((overloadable)) ceil(double4);
00610 double8 const_func __attribute__((overloadable)) ceil(double8);
00611 double16 const_func __attribute__((overloadable)) ceil(double16);
00612 //half const_func __attribute__((overloadable)) ceil(half);
00613 //half2 const_func __attribute__((overloadable)) ceil(half2);
00614 //half3 const_func __attribute__((overloadable)) ceil(half3);
00615 //half4 const_func __attribute__((overloadable)) ceil(half4);
00616 //half8 const_func __attribute__((overloadable)) ceil(half8);
00617 //half16 const_func __attribute__((overloadable)) ceil(half16);
00618
00619 /**
00620 * Returns x with its sign changed to match the sign of
00621 * y.
00622 */
00623 float const_func __attribute__((overloadable)) copysign(float x, float y);
00624 float2 const_func __attribute__((overloadable)) copysign(float2 x, float2 y);
00625 float3 const_func __attribute__((overloadable)) copysign(float3 x, float3 y);
00626 float4 const_func __attribute__((overloadable)) copysign(float4 x, float4 y);
00627 float8 const_func __attribute__((overloadable)) copysign(float8 x, float8 y);
00628 float16 const_func __attribute__((overloadable)) copysign(float16 x, float16 y);
00629 double const_func __attribute__((overloadable)) copysign(double x, double y);
00630 double2 const_func __attribute__((overloadable)) copysign(double2 x, double2 y);
00631 double3 const_func __attribute__((overloadable)) copysign(double3 x, double3 y);
00632 double4 const_func __attribute__((overloadable)) copysign(double4 x, double4 y);
00633 double8 const_func __attribute__((overloadable)) copysign(double8 x, double8 y);
00634 double16 const_func __attribute__((overloadable)) copysign(double16 x, double16 y);
00635 //half const_func __attribute__((overloadable)) copysign(half x, half y);
00636 //half2 const_func __attribute__((overloadable)) copysign(half2 x, half2 y);
00637 //half3 const_func __attribute__((overloadable)) copysign(half3 x, half3 y);
00638 //half4 const_func __attribute__((overloadable)) copysign(half4 x, half4 y);
00639 //half8 const_func __attribute__((overloadable)) copysign(half8 x, half8 y);
00640 //half16 const_func __attribute__((overloadable)) copysign(half16 x, half16 y);
00641
00642 /**
00643 * Compute cosine.
00644 */
00645 double const_func __attribute__((overloadable)) cos(double);
00646 double2 const_func __attribute__((overloadable)) cos(double2);
00647 double3 const_func __attribute__((overloadable)) cos(double3);
00648 double4 const_func __attribute__((overloadable)) cos(double4);
00649 double8 const_func __attribute__((overloadable)) cos(double8);
00650 double16 const_func __attribute__((overloadable)) cos(double16);
00651 float const_func __attribute__((overloadable)) cos(float);
00652 float2 const_func __attribute__((overloadable)) cos(float2);
00653 float3 const_func __attribute__((overloadable)) cos(float3);
00654 float4 const_func __attribute__((overloadable)) cos(float4);
00655 float8 const_func __attribute__((overloadable)) cos(float8);
00656 float16 const_func __attribute__((overloadable)) cos(float16);
00657 //half const_func __attribute__((overloadable)) cos(half);
00658 //half2 const_func __attribute__((overloadable)) cos(half2);
00659 //half3 const_func __attribute__((overloadable)) cos(half3);
00660 //half4 const_func __attribute__((overloadable)) cos(half4);
00661 //half8 const_func __attribute__((overloadable)) cos(half8);
```

```
00662 //half16 const_func __attribute__((overloadable)) cos(half16);
00663
00664 /**
00665 * Compute hyperbolic consine.
00666 */
00667 float const_func __attribute__((overloadable)) cosh(float);
00668 float2 const_func __attribute__((overloadable)) cosh(float2);
00669 float3 const_func __attribute__((overloadable)) cosh(float3);
00670 float4 const_func __attribute__((overloadable)) cosh(float4);
00671 float8 const_func __attribute__((overloadable)) cosh(float8);
00672 float16 const_func __attribute__((overloadable)) cosh(float16);
00673 double const_func __attribute__((overloadable)) cosh(double);
00674 double2 const_func __attribute__((overloadable)) cosh(double2);
00675 double3 const_func __attribute__((overloadable)) cosh(double3);
00676 double4 const_func __attribute__((overloadable)) cosh(double4);
00677 double8 const_func __attribute__((overloadable)) cosh(double8);
00678 double16 const_func __attribute__((overloadable)) cosh(double16);
00679 //half const_func __attribute__((overloadable)) cosh(half);
00680 //half2 const_func __attribute__((overloadable)) cosh(half2);
00681 //half3 const_func __attribute__((overloadable)) cosh(half3);
00682 //half4 const_func __attribute__((overloadable)) cosh(half4);
00683 //half8 const_func __attribute__((overloadable)) cosh(half8);
00684 //half16 const_func __attribute__((overloadable)) cosh(half16);
00685
00686 /**
00687 * Compute cos (PI * x).
00688 */
00689 float const_func __attribute__((overloadable)) cospi(float x);
00690 float2 const_func __attribute__((overloadable)) cospi(float2 x);
00691 float3 const_func __attribute__((overloadable)) cospi(float3 x);
00692 float4 const_func __attribute__((overloadable)) cospi(float4 x);
00693 float8 const_func __attribute__((overloadable)) cospi(float8 x);
00694 float16 const_func __attribute__((overloadable)) cospi(float16 x);
00695 double const_func __attribute__((overloadable)) cospi(double x);
00696 double2 const_func __attribute__((overloadable)) cospi(double2 x);
00697 double3 const_func __attribute__((overloadable)) cospi(double3 x);
00698 double4 const_func __attribute__((overloadable)) cospi(double4 x);
00699 double8 const_func __attribute__((overloadable)) cospi(double8 x);
00700 double16 const_func __attribute__((overloadable)) cospi(double16 x);
00701 //half const_func __attribute__((overloadable)) cospi(half x);
00702 //half2 const_func __attribute__((overloadable)) cospi(half2 x);
00703 //half3 const_func __attribute__((overloadable)) cospi(half3 x);
00704 //half4 const_func __attribute__((overloadable)) cospi(half4 x);
00705 //half8 const_func __attribute__((overloadable)) cospi(half8 x);
00706 //half16 const_func __attribute__((overloadable)) cospi(half16 x);
00707
00708 /**
00709 * Complementary error function.
00710 */
00711 float const_func __attribute__((overloadable)) erfc(float);
00712 float2 const_func __attribute__((overloadable)) erfc(float2);
00713 float3 const_func __attribute__((overloadable)) erfc(float3);
00714 float4 const_func __attribute__((overloadable)) erfc(float4);
00715 float8 const_func __attribute__((overloadable)) erfc(float8);
00716 float16 const_func __attribute__((overloadable)) erfc(float16);
00717 double const_func __attribute__((overloadable)) erfc(double);
00718 double2 const_func __attribute__((overloadable)) erfc(double2);
00719 double3 const_func __attribute__((overloadable)) erfc(double3);
00720 double4 const_func __attribute__((overloadable)) erfc(double4);
00721 double8 const_func __attribute__((overloadable)) erfc(double8);
00722 double16 const_func __attribute__((overloadable)) erfc(double16);
00723 //half const_func __attribute__((overloadable)) erfc(half);
00724 //half2 const_func __attribute__((overloadable)) erfc(half2);
00725 //half3 const_func __attribute__((overloadable)) erfc(half3);
00726 //half4 const_func __attribute__((overloadable)) erfc(half4);
00727 //half8 const_func __attribute__((overloadable)) erfc(half8);
00728 //half16 const_func __attribute__((overloadable)) erfc(half16);
00729
00730 /**
00731 * Error function encountered in integrating the
00732 * normal distribution.
00733 */
00734 float const_func __attribute__((overloadable)) erf(float);
00735 float2 const_func __attribute__((overloadable)) erf(float2);
00736 float3 const_func __attribute__((overloadable)) erf(float3);
00737 float4 const_func __attribute__((overloadable)) erf(float4);
00738 float8 const_func __attribute__((overloadable)) erf(float8);
00739 float16 const_func __attribute__((overloadable)) erf(float16);
00740 double const_func __attribute__((overloadable)) erf(double);
00741 double2 const_func __attribute__((overloadable)) erf(double2);
00742 double3 const_func __attribute__((overloadable)) erf(double3);
00743 double4 const_func __attribute__((overloadable)) erf(double4);
00744 double8 const_func __attribute__((overloadable)) erf(double8);
00745 double16 const_func __attribute__((overloadable)) erf(double16);
00746 //half const_func __attribute__((overloadable)) erf(half);
00747 //half2 const_func __attribute__((overloadable)) erf(half2);
00748 //half3 const_func __attribute__((overloadable)) erf(half3);
```

```
00749 //half4 const_func __attribute__((overloadable)) erf(half4);
00750 //half8 const_func __attribute__((overloadable)) erf(half8);
00751 //half16 const_func __attribute__((overloadable)) erf(half16);
00752
00753 /**
00754 * Compute the base- e exponential of x.
00755 */
00756 float const_func __attribute__((overloadable)) exp(float x);
00757 float2 const_func __attribute__((overloadable)) exp(float2 x);
00758 float3 const_func __attribute__((overloadable)) exp(float3 x);
00759 float4 const_func __attribute__((overloadable)) exp(float4 x);
00760 float8 const_func __attribute__((overloadable)) exp(float8 x);
00761 float16 const_func __attribute__((overloadable)) exp(float16 x);
00762 double const_func __attribute__((overloadable)) exp(double x);
00763 double2 const_func __attribute__((overloadable)) exp(double2 x);
00764 double3 const_func __attribute__((overloadable)) exp(double3 x);
00765 double4 const_func __attribute__((overloadable)) exp(double4 x);
00766 double8 const_func __attribute__((overloadable)) exp(double8 x);
00767 double16 const_func __attribute__((overloadable)) exp(double16 x);
00768 //half const_func __attribute__((overloadable)) exp(half x);
00769 //half2 const_func __attribute__((overloadable)) exp(half2 x);
00770 //half3 const_func __attribute__((overloadable)) exp(half3 x);
00771 //half4 const_func __attribute__((overloadable)) exp(half4 x);
00772 //half8 const_func __attribute__((overloadable)) exp(half8 x);
00773 //half16 const_func __attribute__((overloadable)) exp(half16 x);
00774
00775 /**
00776 * Exponential base 2 function.
00777 */
00778 float const_func __attribute__((overloadable)) exp2(float);
00779 float2 const_func __attribute__((overloadable)) exp2(float2);
00780 float3 const_func __attribute__((overloadable)) exp2(float3);
00781 float4 const_func __attribute__((overloadable)) exp2(float4);
00782 float8 const_func __attribute__((overloadable)) exp2(float8);
00783 float16 const_func __attribute__((overloadable)) exp2(float16);
00784 double const_func __attribute__((overloadable)) exp2(double);
00785 double2 const_func __attribute__((overloadable)) exp2(double2);
00786 double3 const_func __attribute__((overloadable)) exp2(double3);
00787 double4 const_func __attribute__((overloadable)) exp2(double4);
00788 double8 const_func __attribute__((overloadable)) exp2(double8);
00789 double16 const_func __attribute__((overloadable)) exp2(double16);
00790 //half const_func __attribute__((overloadable)) exp2(half);
00791 //half2 const_func __attribute__((overloadable)) exp2(half2);
00792 //half3 const_func __attribute__((overloadable)) exp2(half3);
00793 //half4 const_func __attribute__((overloadable)) exp2(half4);
00794 //half8 const_func __attribute__((overloadable)) exp2(half8);
00795 //half16 const_func __attribute__((overloadable)) exp2(half16);
00796
00797 /**
00798 * Exponential base 10 function.
00799 */
00800 float const_func __attribute__((overloadable)) exp10(float);
00801 float2 const_func __attribute__((overloadable)) exp10(float2);
00802 float3 const_func __attribute__((overloadable)) exp10(float3);
00803 float4 const_func __attribute__((overloadable)) exp10(float4);
00804 float8 const_func __attribute__((overloadable)) exp10(float8);
00805 float16 const_func __attribute__((overloadable)) exp10(float16);
00806 double const_func __attribute__((overloadable)) exp10(double);
00807 double2 const_func __attribute__((overloadable)) exp10(double2);
00808 double3 const_func __attribute__((overloadable)) exp10(double3);
00809 double4 const_func __attribute__((overloadable)) exp10(double4);
00810 double8 const_func __attribute__((overloadable)) exp10(double8);
00811 double16 const_func __attribute__((overloadable)) exp10(double16);
00812 //half const_func __attribute__((overloadable)) exp10(half);
00813 //half2 const_func __attribute__((overloadable)) exp10(half2);
00814 //half3 const_func __attribute__((overloadable)) exp10(half3);
00815 //half4 const_func __attribute__((overloadable)) exp10(half4);
00816 //half8 const_func __attribute__((overloadable)) exp10(half8);
00817 //half16 const_func __attribute__((overloadable)) exp10(half16);
00818
00819 /**
00820 * Compute e^x- 1.0.
00821 */
00822 float const_func __attribute__((overloadable)) expm1(float x);
00823 float2 const_func __attribute__((overloadable)) expm1(float2 x);
00824 float3 const_func __attribute__((overloadable)) expm1(float3 x);
00825 float4 const_func __attribute__((overloadable)) expm1(float4 x);
00826 float8 const_func __attribute__((overloadable)) expm1(float8 x);
00827 float16 const_func __attribute__((overloadable)) expm1(float16 x);
00828 double const_func __attribute__((overloadable)) expm1(double x);
00829 double2 const_func __attribute__((overloadable)) expm1(double2 x);
00830 double3 const_func __attribute__((overloadable)) expm1(double3 x);
00831 double4 const_func __attribute__((overloadable)) expm1(double4 x);
00832 double8 const_func __attribute__((overloadable)) expm1(double8 x);
00833 double16 const_func __attribute__((overloadable)) expm1(double16 x);
00834 //half const_func __attribute__((overloadable)) expm1(half x);
00835 //half2 const_func __attribute__((overloadable)) expm1(half2 x);
```

```
00836 //half3 const_func __attribute__((overloadable)) expm1(half3 x);
00837 //half4 const_func __attribute__((overloadable)) expm1(half4 x);
00838 //half8 const_func __attribute__((overloadable)) expm1(half8 x);
00839 //half16 const_func __attribute__((overloadable)) expm1(half16 x);
00840
00841 /**
00842 * Compute absolute value of a floating-point number.
00843 */
00844 float const_func __attribute__((overloadable)) fabs(float);
00845 float2 const_func __attribute__((overloadable)) fabs(float2);
00846 float3 const_func __attribute__((overloadable)) fabs(float3);
00847 float4 const_func __attribute__((overloadable)) fabs(float4);
00848 float8 const_func __attribute__((overloadable)) fabs(float8);
00849 float16 const_func __attribute__((overloadable)) fabs(float16);
00850 double const_func __attribute__((overloadable)) fabs(double);
00851 double2 const_func __attribute__((overloadable)) fabs(double2);
00852 double3 const_func __attribute__((overloadable)) fabs(double3);
00853 double4 const_func __attribute__((overloadable)) fabs(double4);
00854 double8 const_func __attribute__((overloadable)) fabs(double8);
00855 double16 const_func __attribute__((overloadable)) fabs(double16);
00856 //half const_func __attribute__((overloadable)) fabs(half);
00857 //half2 const_func __attribute__((overloadable)) fabs(half2);
00858 //half3 const_func __attribute__((overloadable)) fabs(half3);
00859 //half4 const_func __attribute__((overloadable)) fabs(half4);
00860 //half8 const_func __attribute__((overloadable)) fabs(half8);
00861 //half16 const_func __attribute__((overloadable)) fabs(half16);
00862
00863 /**
00864 * x - y if x > y, +0 if x is less than or equal to y.
00865 */
00866 float const_func __attribute__((overloadable)) fdim(float x, float y);
00867 float2 const_func __attribute__((overloadable)) fdim(float2 x, float2 y);
00868 float3 const_func __attribute__((overloadable)) fdim(float3 x, float3 y);
00869 float4 const_func __attribute__((overloadable)) fdim(float4 x, float4 y);
00870 float8 const_func __attribute__((overloadable)) fdim(float8 x, float8 y);
00871 float16 const_func __attribute__((overloadable)) fdim(float16 x, float16 y);
00872 double const_func __attribute__((overloadable)) fdim(double x, double y);
00873 double2 const_func __attribute__((overloadable)) fdim(double2 x, double2 y);
00874 double3 const_func __attribute__((overloadable)) fdim(double3 x, double3 y);
00875 double4 const_func __attribute__((overloadable)) fdim(double4 x, double4 y);
00876 double8 const_func __attribute__((overloadable)) fdim(double8 x, double8 y);
00877 double16 const_func __attribute__((overloadable)) fdim(double16 x, double16 y);
00878 //half const_func __attribute__((overloadable)) fdim(half x, half y);
00879 //half2 const_func __attribute__((overloadable)) fdim(half2 x, half2 y);
00880 //half3 const_func __attribute__((overloadable)) fdim(half3 x, half3 y);
00881 //half4 const_func __attribute__((overloadable)) fdim(half4 x, half4 y);
00882 //half8 const_func __attribute__((overloadable)) fdim(half8 x, half8 y);
00883 //half16 const_func __attribute__((overloadable)) fdim(half16 x, half16 y);
00884
00885 /**
00886 * Round to integral value using the round to -ve
00887 * infinity rounding mode.
00888 */
00889 float const_func __attribute__((overloadable)) floor(float);
00890 float2 const_func __attribute__((overloadable)) floor(float2);
00891 float3 const_func __attribute__((overloadable)) floor(float3);
00892 float4 const_func __attribute__((overloadable)) floor(float4);
00893 float8 const_func __attribute__((overloadable)) floor(float8);
00894 float16 const_func __attribute__((overloadable)) floor(float16);
00895 double const_func __attribute__((overloadable)) floor(double);
00896 double2 const_func __attribute__((overloadable)) floor(double2);
00897 double3 const_func __attribute__((overloadable)) floor(double3);
00898 double4 const_func __attribute__((overloadable)) floor(double4);
00899 double8 const_func __attribute__((overloadable)) floor(double8);
00900 double16 const_func __attribute__((overloadable)) floor(double16);
00901 //half const_func __attribute__((overloadable)) floor(half);
00902 //half2 const_func __attribute__((overloadable)) floor(half2);
00903 //half3 const_func __attribute__((overloadable)) floor(half3);
00904 //half4 const_func __attribute__((overloadable)) floor(half4);
00905 //half8 const_func __attribute__((overloadable)) floor(half8);
00906 //half16 const_func __attribute__((overloadable)) floor(half16);
00907
00908 /**
00909 * Returns the correctly rounded floating-point
00910 * representation of the sum of c with the infinitely
00911 * precise product of a and b.  Rounding of
00912 * intermediate products shall not occur.  Edge case
00913 * behavior is per the IEEE 754-2008 standard.
00914 */
00915 float const_func __attribute__((overloadable)) fma(float a, float b, float c);
00916 float2 const_func __attribute__((overloadable)) fma(float2 a, float2 b, float2 c);
00917 float3 const_func __attribute__((overloadable)) fma(float3 a, float3 b, float3 c);
00918 float4 const_func __attribute__((overloadable)) fma(float4 a, float4 b, float4 c);
00919 float8 const_func __attribute__((overloadable)) fma(float8 a, float8 b, float8 c);
00920 float16 const_func __attribute__((overloadable)) fma(float16 a, float16 b, float16 c);
00921 double const_func __attribute__((overloadable)) fma(double a, double b, double c);
00922 double2 const_func __attribute__((overloadable)) fma(double2 a, double2 b, double2 c);
```

```
00923 double3 const_func __attribute__((overloadable)) fma(double3 a, double3 b, double3 c);
00924 double4 const_func __attribute__((overloadable)) fma(double4 a, double4 b, double4 c);
00925 double8 const_func __attribute__((overloadable)) fma(double8 a, double8 b, double8 c);
00926 double16 const_func __attribute__((overloadable)) fma(double16 a, double16 b, double16 c);
00927 //half const_func __attribute__((overloadable)) fma(half a, half b, half c);
00928 //half2 const_func __attribute__((overloadable)) fma(half2 a, half2 b, half2 c);
00929 //half3 const_func __attribute__((overloadable)) fma(half3 a, half3 b, half3 c);
00930 //half4 const_func __attribute__((overloadable)) fma(half4 a, half4 b, half4 c);
00931 //half8 const_func __attribute__((overloadable)) fma(half8 a, half8 b, half8 c);
00932 //half16 const_func __attribute__((overloadable)) fma(half16 a, half16 b, half16 c);
00933
00934 /**
00935 * Returns y if x < y, otherwise it returns x.  If one
00936 * argument is a NaN, fmax() returns the other
00937 * argument.  If both arguments are NaNs, fmax()
00938 * returns a NaN.
00939 */
00940 float const_func __attribute__((overloadable)) fmax(float x, float y);
00941 float2 const_func __attribute__((overloadable)) fmax(float2 x, float2 y);
00942 float3 const_func __attribute__((overloadable)) fmax(float3 x, float3 y);
00943 float4 const_func __attribute__((overloadable)) fmax(float4 x, float4 y);
00944 float8 const_func __attribute__((overloadable)) fmax(float8 x, float8 y);
00945 float16 const_func __attribute__((overloadable)) fmax(float16 x, float16 y);
00946 float2 const_func __attribute__((overloadable)) fmax(float2 x, float y);
00947 float3 const_func __attribute__((overloadable)) fmax(float3 x, float y);
00948 float4 const_func __attribute__((overloadable)) fmax(float4 x, float y);
00949 float8 const_func __attribute__((overloadable)) fmax(float8 x, float y);
00950 float16 const_func __attribute__((overloadable)) fmax(float16 x, float y);
00951 double const_func __attribute__((overloadable)) fmax(double x, double y);
00952 double2 const_func __attribute__((overloadable)) fmax(double2 x, double2 y);
00953 double3 const_func __attribute__((overloadable)) fmax(double3 x, double3 y);
00954 double4 const_func __attribute__((overloadable)) fmax(double4 x, double4 y);
00955 double8 const_func __attribute__((overloadable)) fmax(double8 x, double8 y);
00956 double16 const_func __attribute__((overloadable)) fmax(double16 x, double16 y);
00957 double2 const_func __attribute__((overloadable)) fmax(double2 x, double y);
00958 double3 const_func __attribute__((overloadable)) fmax(double3 x, double y);
00959 double4 const_func __attribute__((overloadable)) fmax(double4 x, double y);
00960 double8 const_func __attribute__((overloadable)) fmax(double8 x, double y);
00961 double16 const_func __attribute__((overloadable)) fmax(double16 x, double y);
00962 //half const_func __attribute__((overloadable)) fmax(half x, half y);
00963 //half2 const_func __attribute__((overloadable)) fmax(half2 x, half2 y);
00964 //half3 const_func __attribute__((overloadable)) fmax(half3 x, half3 y);
00965 //half4 const_func __attribute__((overloadable)) fmax(half4 x, half4 y);
00966 //half8 const_func __attribute__((overloadable)) fmax(half8 x, half8 y);
00967 //half16 const_func __attribute__((overloadable)) fmax(half16 x, half16 y);
00968 //half2 const_func __attribute__((overloadable)) fmax(half2 x, half y);
00969 //half3 const_func __attribute__((overloadable)) fmax(half3 x, half y);
00970 //half4 const_func __attribute__((overloadable)) fmax(half4 x, half y);
00971 //half8 const_func __attribute__((overloadable)) fmax(half8 x, half y);
00972 //half16 const_func __attribute__((overloadable)) fmax(half16 x, half y);
00973
00974 /**
00975 * Returns y if y < x, otherwise it returns x.  If one
00976 * argument is a NaN, fmin() returns the other
00977 * argument.  If both arguments are NaNs, fmin()
00978 * returns a NaN.
00979 */
00980 float const_func __attribute__((overloadable)) fmin(float x, float y);
00981 float2 const_func __attribute__((overloadable)) fmin(float2 x, float2 y);
00982 float3 const_func __attribute__((overloadable)) fmin(float3 x, float3 y);
00983 float4 const_func __attribute__((overloadable)) fmin(float4 x, float4 y);
00984 float8 const_func __attribute__((overloadable)) fmin(float8 x, float8 y);
00985 float16 const_func __attribute__((overloadable)) fmin(float16 x, float16 y);
00986 float2 const_func __attribute__((overloadable)) fmin(float2 x, float y);
00987 float3 const_func __attribute__((overloadable)) fmin(float3 x, float y);
00988 float4 const_func __attribute__((overloadable)) fmin(float4 x, float y);
00989 float8 const_func __attribute__((overloadable)) fmin(float8 x, float y);
00990 float16 const_func __attribute__((overloadable)) fmin(float16 x, float y);
00991 double const_func __attribute__((overloadable)) fmin(double x, double y);
00992 double2 const_func __attribute__((overloadable)) fmin(double2 x, double2 y);
00993 double3 const_func __attribute__((overloadable)) fmin(double3 x, double3 y);
00994 double4 const_func __attribute__((overloadable)) fmin(double4 x, double4 y);
00995 double8 const_func __attribute__((overloadable)) fmin(double8 x, double8 y);
00996 double16 const_func __attribute__((overloadable)) fmin(double16 x, double16 y);
00997 double2 const_func __attribute__((overloadable)) fmin(double2 x, double y);
00998 double3 const_func __attribute__((overloadable)) fmin(double3 x, double y);
00999 double4 const_func __attribute__((overloadable)) fmin(double4 x, double y);
01000 double8 const_func __attribute__((overloadable)) fmin(double8 x, double y);
01001 double16 const_func __attribute__((overloadable)) fmin(double16 x, double y);
01002 //half const_func __attribute__((overloadable)) fmin(half x, half y);
01003 //half2 const_func __attribute__((overloadable)) fmin(half2 x, half2 y);
01004 //half3 const_func __attribute__((overloadable)) fmin(half3 x, half3 y);
01005 //half4 const_func __attribute__((overloadable)) fmin(half4 x, half4 y);
01006 //half8 const_func __attribute__((overloadable)) fmin(half8 x, half8 y);
01007 //half16 const_func __attribute__((overloadable)) fmin(half16 x, half16 y);
01008 //half2 const_func __attribute__((overloadable)) fmin(half2 x, half y);
01009 //half3 const_func __attribute__((overloadable)) fmin(half3 x, half y);
```

```
01010 //half4 const_func __attribute__((overloadable)) fmin(half4 x, half y);
01011 //half8 const_func __attribute__((overloadable)) fmin(half8 x, half y);
01012 //half16 const_func __attribute__((overloadable)) fmin(half16 x, half y);
01013
01014 /**
01015 * Modulus.  Returns x - y * trunc (x/y).
01016 */
01017 float const_func __attribute__((overloadable)) fmod(float x, float y);
01018 float2 const_func __attribute__((overloadable)) fmod(float2 x, float2 y);
01019 float3 const_func __attribute__((overloadable)) fmod(float3 x, float3 y);
01020 float4 const_func __attribute__((overloadable)) fmod(float4 x, float4 y);
01021 float8 const_func __attribute__((overloadable)) fmod(float8 x, float8 y);
01022 float16 const_func __attribute__((overloadable)) fmod(float16 x, float16 y);
01023 double const_func __attribute__((overloadable)) fmod(double x, double y);
01024 double2 const_func __attribute__((overloadable)) fmod(double2 x, double2 y);
01025 double3 const_func __attribute__((overloadable)) fmod(double3 x, double3 y);
01026 double4 const_func __attribute__((overloadable)) fmod(double4 x, double4 y);
01027 double8 const_func __attribute__((overloadable)) fmod(double8 x, double8 y);
01028 double16 const_func __attribute__((overloadable)) fmod(double16 x, double16 y);
01029 //half const_func __attribute__((overloadable)) fmod(half x, half y);
01030 //half2 const_func __attribute__((overloadable)) fmod(half2 x, half2 y);
01031 //half3 const_func __attribute__((overloadable)) fmod(half3 x, half3 y);
01032 //half4 const_func __attribute__((overloadable)) fmod(half4 x, half4 y);
01033 //half8 const_func __attribute__((overloadable)) fmod(half8 x, half8 y);
01034 //half16 const_func __attribute__((overloadable)) fmod(half16 x, half16 y);
01035
01036 /**
01037 * Returns fmin( x - floor (x), 0x1.fffffep-1f ).
01038 * floor(x) is returned in iptr.
01039 */
01040 float __attribute__((overloadable)) fract(float x, __global float *iptr);
01041 float2 __attribute__((overloadable)) fract(float2 x, __global float2 *iptr);
01042 float3 __attribute__((overloadable)) fract(float3 x, __global float3 *iptr);
01043 float4 __attribute__((overloadable)) fract(float4 x, __global float4 *iptr);
01044 float8 __attribute__((overloadable)) fract(float8 x, __global float8 *iptr);
01045 float16 __attribute__((overloadable)) fract(float16 x, __global float16 *iptr);
01046 float __attribute__((overloadable)) fract(float x, __local float *iptr);
01047 float2 __attribute__((overloadable)) fract(float2 x, __local float2 *iptr);
01048 float3 __attribute__((overloadable)) fract(float3 x, __local float3 *iptr);
01049 float4 __attribute__((overloadable)) fract(float4 x, __local float4 *iptr);
01050 float8 __attribute__((overloadable)) fract(float8 x, __local float8 *iptr);
01051 float16 __attribute__((overloadable)) fract(float16 x, __local float16 *iptr);
01052 float __attribute__((overloadable)) fract(float x, __private float *iptr);
01053 float2 __attribute__((overloadable)) fract(float2 x, __private float2 *iptr);
01054 float3 __attribute__((overloadable)) fract(float3 x, __private float3 *iptr);
01055 float4 __attribute__((overloadable)) fract(float4 x, __private float4 *iptr);
01056 float8 __attribute__((overloadable)) fract(float8 x, __private float8 *iptr);
01057 float16 __attribute__((overloadable)) fract(float16 x, __private float16 *iptr);
01058 double __attribute__((overloadable)) fract(double x, __global double *iptr);
01059 double2 __attribute__((overloadable)) fract(double2 x, __global double2 *iptr);
01060 double3 __attribute__((overloadable)) fract(double3 x, __global double3 *iptr);
01061 double4 __attribute__((overloadable)) fract(double4 x, __global double4 *iptr);
01062 double8 __attribute__((overloadable)) fract(double8 x, __global double8 *iptr);
01063 double16 __attribute__((overloadable)) fract(double16 x, __global double16 *iptr);
01064 double __attribute__((overloadable)) fract(double x, __local double *iptr);
01065 double2 __attribute__((overloadable)) fract(double2 x, __local double2 *iptr);
01066 double3 __attribute__((overloadable)) fract(double3 x, __local double3 *iptr);
01067 double4 __attribute__((overloadable)) fract(double4 x, __local double4 *iptr);
01068 double8 __attribute__((overloadable)) fract(double8 x, __local double8 *iptr);
01069 double16 __attribute__((overloadable)) fract(double16 x, __local double16 *iptr);
01070 double __attribute__((overloadable)) fract(double x, __private double *iptr);
01071 double2 __attribute__((overloadable)) fract(double2 x, __private double2 *iptr);
01072 double3 __attribute__((overloadable)) fract(double3 x, __private double3 *iptr);
01073 double4 __attribute__((overloadable)) fract(double4 x, __private double4 *iptr);
01074 double8 __attribute__((overloadable)) fract(double8 x, __private double8 *iptr);
01075 double16 __attribute__((overloadable)) fract(double16 x, __private double16 *iptr);
01076 //half __attribute__((overloadable)) fract(half x, __global half *iptr);
01077 //half2 __attribute__((overloadable)) fract(half2 x, __global half2 *iptr);
01078 //half3 __attribute__((overloadable)) fract(half3 x, __global half3 *iptr);
01079 //half4 __attribute__((overloadable)) fract(half4 x, __global half4 *iptr);
01080 //half8 __attribute__((overloadable)) fract(half8 x, __global half8 *iptr);
01081 //half16 __attribute__((overloadable)) fract(half16 x, __global half16 *iptr);
01082 //half __attribute__((overloadable)) fract(half x, __local half *iptr);
01083 //half2 __attribute__((overloadable)) fract(half2 x, __local half2 *iptr);
01084 //half3 __attribute__((overloadable)) fract(half3 x, __local half3 *iptr);
01085 //half4 __attribute__((overloadable)) fract(half4 x, __local half4 *iptr);
01086 //half8 __attribute__((overloadable)) fract(half8 x, __local half8 *iptr);
01087 //half16 __attribute__((overloadable)) fract(half16 x, __local half16 *iptr);
01088 //half __attribute__((overloadable)) fract(half x, __private half *iptr);
01089 //half2 __attribute__((overloadable)) fract(half2 x, __private half2 *iptr);
01090 //half3 __attribute__((overloadable)) fract(half3 x, __private half3 *iptr);
01091 //half4 __attribute__((overloadable)) fract(half4 x, __private half4 *iptr);
01092 //half8 __attribute__((overloadable)) fract(half8 x, __private half8 *iptr);
01093 //half16 __attribute__((overloadable)) fract(half16 x, __private half16 *iptr);
01094
01095
01096 /**
```

```
01097 * Extract mantissa and exponent from x.  For each
01098 * component the mantissa returned is a float with
01099 * magnitude in the interval [1/2, 1) or 0.  Each
01100 * component of x equals mantissa returned * 2^exp.
01101 */
01102 float __attribute__((overloadable)) frexp(float x, __global int *exp);
01103 float2 __attribute__((overloadable)) frexp(float2 x, __global int2 *exp);
01104 float3 __attribute__((overloadable)) frexp(float3 x, __global int3 *exp);
01105 float4 __attribute__((overloadable)) frexp(float4 x, __global int4 *exp);
01106 float8 __attribute__((overloadable)) frexp(float8 x, __global int8 *exp);
01107 float16 __attribute__((overloadable)) frexp(float16 x, __global int16 *exp);
01108 float __attribute__((overloadable)) frexp(float x, __local int *exp);
01109 float2 __attribute__((overloadable)) frexp(float2 x, __local int2 *exp);
01110 float3 __attribute__((overloadable)) frexp(float3 x, __local int3 *exp);
01111 float4 __attribute__((overloadable)) frexp(float4 x, __local int4 *exp);
01112 float8 __attribute__((overloadable)) frexp(float8 x, __local int8 *exp);
01113 float16 __attribute__((overloadable)) frexp(float16 x, __local int16 *exp);
01114 float __attribute__((overloadable)) frexp(float x, __private int *exp);
01115 float2 __attribute__((overloadable)) frexp(float2 x, __private int2 *exp);
01116 float3 __attribute__((overloadable)) frexp(float3 x, __private int3 *exp);
01117 float4 __attribute__((overloadable)) frexp(float4 x, __private int4 *exp);
01118 float8 __attribute__((overloadable)) frexp(float8 x, __private int8 *exp);
01119 float16 __attribute__((overloadable)) frexp(float16 x, __private int16 *exp);
01120 double __attribute__((overloadable)) frexp(double x, __global int *exp);
01121 double2 __attribute__((overloadable)) frexp(double2 x, __global int2 *exp);
01122 double3 __attribute__((overloadable)) frexp(double3 x, __global int3 *exp);
01123 double4 __attribute__((overloadable)) frexp(double4 x, __global int4 *exp);
01124 double8 __attribute__((overloadable)) frexp(double8 x, __global int8 *exp);
01125 double16 __attribute__((overloadable)) frexp(double16 x, __global int16 *exp);
01126 double __attribute__((overloadable)) frexp(double x, __local int *exp);
01127 double2 __attribute__((overloadable)) frexp(double2 x, __local int2 *exp);
01128 double3 __attribute__((overloadable)) frexp(double3 x, __local int3 *exp);
01129 double4 __attribute__((overloadable)) frexp(double4 x, __local int4 *exp);
01130 double8 __attribute__((overloadable)) frexp(double8 x, __local int8 *exp);
01131 double16 __attribute__((overloadable)) frexp(double16 x, __local int16 *exp);
01132 double __attribute__((overloadable)) frexp(double x, __private int *exp);
01133 double2 __attribute__((overloadable)) frexp(double2 x, __private int2 *exp);
01134 double3 __attribute__((overloadable)) frexp(double3 x, __private int3 *exp);
01135 double4 __attribute__((overloadable)) frexp(double4 x, __private int4 *exp);
01136 double8 __attribute__((overloadable)) frexp(double8 x, __private int8 *exp);
01137 double16 __attribute__((overloadable)) frexp(double16 x, __private int16 *exp);
01138 //half __attribute__((overloadable)) frexp(half x, __global int *exp);
01139 //half2 __attribute__((overloadable)) frexp(half2 x, __global int2 *exp);
01140 //half3 __attribute__((overloadable)) frexp(half3 x, __global int3 *exp);
01141 //half4 __attribute__((overloadable)) frexp(half4 x, __global int4 *exp);
01142 //half8 __attribute__((overloadable)) frexp(half8 x, __global int8 *exp);
01143 //half16 __attribute__((overloadable)) frexp(half16 x, __global int16 *exp);
01144 //half __attribute__((overloadable)) frexp(half x, __local int *exp);
01145 //half2 __attribute__((overloadable)) frexp(half2 x, __local int2 *exp);
01146 //half3 __attribute__((overloadable)) frexp(half3 x, __local int3 *exp);
01147 //half4 __attribute__((overloadable)) frexp(half4 x, __local int4 *exp);
01148 //half8 __attribute__((overloadable)) frexp(half8 x, __local int8 *exp);
01149 //half16 __attribute__((overloadable)) frexp(half16 x, __local int16 *exp);
01150 //half __attribute__((overloadable)) frexp(half x, __private int *exp);
01151 //half2 __attribute__((overloadable)) frexp(half2 x, __private int2 *exp);
01152 //half3 __attribute__((overloadable)) frexp(half3 x, __private int3 *exp);
01153 //half4 __attribute__((overloadable)) frexp(half4 x, __private int4 *exp);
01154 //half8 __attribute__((overloadable)) frexp(half8 x, __private int8 *exp);
01155 //half16 __attribute__((overloadable)) frexp(half16 x, __private int16 *exp);
01156
01157 /**
01158 * Compute the value of the square root of x^2+ y^2
01159 * without undue overflow or underflow.
01160 */
01161 float const_func __attribute__((overloadable)) hypot(float x, float y);
01162 float2 const_func __attribute__((overloadable)) hypot(float2 x, float2 y);
01163 float3 const_func __attribute__((overloadable)) hypot(float3 x, float3 y);
01164 float4 const_func __attribute__((overloadable)) hypot(float4 x, float4 y);
01165 float8 const_func __attribute__((overloadable)) hypot(float8 x, float8 y);
01166 float16 const_func __attribute__((overloadable)) hypot(float16 x, float16 y);
01167 double const_func __attribute__((overloadable)) hypot(double x, double y);
01168 double2 const_func __attribute__((overloadable)) hypot(double2 x, double2 y);
01169 double3 const_func __attribute__((overloadable)) hypot(double3 x, double3 y);
01170 double4 const_func __attribute__((overloadable)) hypot(double4 x, double4 y);
01171 double8 const_func __attribute__((overloadable)) hypot(double8 x, double8 y);
01172 double16 const_func __attribute__((overloadable)) hypot(double16 x, double16 y);
01173 //half const_func __attribute__((overloadable)) hypot(half x, half y);
01174 //half2 const_func __attribute__((overloadable)) hypot(half2 x, half2 y);
01175 //half3 const_func __attribute__((overloadable)) hypot(half3 x, half3 y);
01176 //half4 const_func __attribute__((overloadable)) hypot(half4 x, half4 y);
01177 //half8 const_func __attribute__((overloadable)) hypot(half8 x, half8 y);
01178 //half16 const_func __attribute__((overloadable)) hypot(half16 x, half16 y);
01179
01180 /**
01181 * Return the exponent as an integer value.
01182 */
01183 int const_func __attribute__((overloadable)) ilogb(float x);
```

```
01184 int2 const_func __attribute__((overloadable)) ilogb(float2 x);
01185 int3 const_func __attribute__((overloadable)) ilogb(float3 x);
01186 int4 const_func __attribute__((overloadable)) ilogb(float4 x);
01187 int8 const_func __attribute__((overloadable)) ilogb(float8 x);
01188 int16 const_func __attribute__((overloadable)) ilogb(float16 x);
01189 int const_func __attribute__((overloadable)) ilogb(double x);
01190 int2 const_func __attribute__((overloadable)) ilogb(double2 x);
01191 int3 const_func __attribute__((overloadable)) ilogb(double3 x);
01192 int4 const_func __attribute__((overloadable)) ilogb(double4 x);
01193 int8 const_func __attribute__((overloadable)) ilogb(double8 x);
01194 int16 const_func __attribute__((overloadable)) ilogb(double16 x);
01195 //int const_func __attribute__((overloadable)) ilogb(half x);
01196 //int2 const_func __attribute__((overloadable)) ilogb(half2 x);
01197 //int3 const_func __attribute__((overloadable)) ilogb(half3 x);
01198 //int4 const_func __attribute__((overloadable)) ilogb(half4 x);
01199 //int8 const_func __attribute__((overloadable)) ilogb(half8 x);
01200 //int16 const_func __attribute__((overloadable)) ilogb(half16 x);
01201
01202 /**
01203 * Multiply x by 2 to the power n.
01204 */
01205 float const_func __attribute__((overloadable)) ldexp(float x, int n);
01206 float2 const_func __attribute__((overloadable)) ldexp(float2 x, int2 n);
01207 float3 const_func __attribute__((overloadable)) ldexp(float3 x, int3 n);
01208 float4 const_func __attribute__((overloadable)) ldexp(float4 x, int4 n);
01209 float8 const_func __attribute__((overloadable)) ldexp(float8 x, int8 n);
01210 float16 const_func __attribute__((overloadable)) ldexp(float16 x, int16 n);
01211 float2 const_func __attribute__((overloadable)) ldexp(float2 x, int n);
01212 float3 const_func __attribute__((overloadable)) ldexp(float3 x, int n);
01213 float4 const_func __attribute__((overloadable)) ldexp(float4 x, int n);
01214 float8 const_func __attribute__((overloadable)) ldexp(float8 x, int n);
01215 float16 const_func __attribute__((overloadable)) ldexp(float16 x, int n);
01216 double const_func __attribute__((overloadable)) ldexp(double x, int n);
01217 double2 const_func __attribute__((overloadable)) ldexp(double2 x, int2 n);
01218 double3 const_func __attribute__((overloadable)) ldexp(double3 x, int3 n);
01219 double4 const_func __attribute__((overloadable)) ldexp(double4 x, int4 n);
01220 double8 const_func __attribute__((overloadable)) ldexp(double8 x, int8 n);
01221 double16 const_func __attribute__((overloadable)) ldexp(double16 x, int16 n);
01222 double2 const_func __attribute__((overloadable)) ldexp(double2 x, int n);
01223 double3 const_func __attribute__((overloadable)) ldexp(double3 x, int n);
01224 double4 const_func __attribute__((overloadable)) ldexp(double4 x, int n);
01225 double8 const_func __attribute__((overloadable)) ldexp(double8 x, int n);
01226 double16 const_func __attribute__((overloadable)) ldexp(double16 x, int n);
01227 //half const_func __attribute__((overloadable)) ldexp(half x, int n);
01228 //half2 const_func __attribute__((overloadable)) ldexp(half2 x, int2 n);
01229 //half3 const_func __attribute__((overloadable)) ldexp(half3 x, int3 n);
01230 //half4 const_func __attribute__((overloadable)) ldexp(half4 x, int4 n);
01231 //half8 const_func __attribute__((overloadable)) ldexp(half8 x, int8 n);
01232 //half16 const_func __attribute__((overloadable)) ldexp(half16 x, int16 n);
01233 //half const_func __attribute__((overloadable)) ldexp(half x, int n);
01234 //half2 const_func __attribute__((overloadable)) ldexp(half2 x, int n);
01235 //half3 const_func __attribute__((overloadable)) ldexp(half3 x, int n);
01236 //half4 const_func __attribute__((overloadable)) ldexp(half4 x, int n);
01237 //half8 const_func __attribute__((overloadable)) ldexp(half8 x, int n);
01238 //half16 const_func __attribute__((overloadable)) ldexp(half16 x, int n);
01239
01240 /**
01241 * Log gamma function.  Returns the natural
01242 * logarithm of the absolute value of the gamma
01243 * function.  The sign of the gamma function is
01244 * returned in the signp argument of lgamma_r.
01245 */
01246 float const_func __attribute__((overloadable)) lgamma(float x);
01247 float2 const_func __attribute__((overloadable)) lgamma(float2 x);
01248 float3 const_func __attribute__((overloadable)) lgamma(float3 x);
01249 float4 const_func __attribute__((overloadable)) lgamma(float4 x);
01250 float8 const_func __attribute__((overloadable)) lgamma(float8 x);
01251 float16 const_func __attribute__((overloadable)) lgamma(float16 x);
01252 double const_func __attribute__((overloadable)) lgamma(double x);
01253 double2 const_func __attribute__((overloadable)) lgamma(double2 x);
01254 double3 const_func __attribute__((overloadable)) lgamma(double3 x);
01255 double4 const_func __attribute__((overloadable)) lgamma(double4 x);
01256 double8 const_func __attribute__((overloadable)) lgamma(double8 x);
01257 double16 const_func __attribute__((overloadable)) lgamma(double16 x);
01258 //half const_func __attribute__((overloadable)) lgamma(half x);
01259 //half2 const_func __attribute__((overloadable)) lgamma(half2 x);
01260 //half3 const_func __attribute__((overloadable)) lgamma(half3 x);
01261 //half4 const_func __attribute__((overloadable)) lgamma(half4 x);
01262 //half8 const_func __attribute__((overloadable)) lgamma(half8 x);
01263 //half16 const_func __attribute__((overloadable)) lgamma(half16 x);
01264
01265 float __attribute__((overloadable)) lgamma_r(float x, __global int *signp);
01266 float2 __attribute__((overloadable)) lgamma_r(float2 x, __global int2 *signp);
01267 float3 __attribute__((overloadable)) lgamma_r(float3 x, __global int3 *signp);
01268 float4 __attribute__((overloadable)) lgamma_r(float4 x, __global int4 *signp);
01269 float8 __attribute__((overloadable)) lgamma_r(float8 x, __global int8 *signp);
01270 float16 __attribute__((overloadable)) lgamma_r(float16 x, __global int16 *signp);
```

```
01271 float __attribute__((overloadable)) lgamma_r(float x, __local int *signp);
01272 float2 __attribute__((overloadable)) lgamma_r(float2 x, __local int2 *signp);
01273 float3 __attribute__((overloadable)) lgamma_r(float3 x, __local int3 *signp);
01274 float4 __attribute__((overloadable)) lgamma_r(float4 x, __local int4 *signp);
01275 float8 __attribute__((overloadable)) lgamma_r(float8 x, __local int8 *signp);
01276 float16 __attribute__((overloadable)) lgamma_r(float16 x, __local int16 *signp);
01277 float __attribute__((overloadable)) lgamma_r(float x, __private int *signp);
01278 float2 __attribute__((overloadable)) lgamma_r(float2 x, __private int2 *signp);
01279 float3 __attribute__((overloadable)) lgamma_r(float3 x, __private int3 *signp);
01280 float4 __attribute__((overloadable)) lgamma_r(float4 x, __private int4 *signp);
01281 float8 __attribute__((overloadable)) lgamma_r(float8 x, __private int8 *signp);
01282 float16 __attribute__((overloadable)) lgamma_r(float16 x, __private int16 *signp);
01283 double __attribute__((overloadable)) lgamma_r(double x, __global int *signp);
01284 double2 __attribute__((overloadable)) lgamma_r(double2 x, __global int2 *signp);
01285 double3 __attribute__((overloadable)) lgamma_r(double3 x, __global int3 *signp);
01286 double4 __attribute__((overloadable)) lgamma_r(double4 x, __global int4 *signp);
01287 double8 __attribute__((overloadable)) lgamma_r(double8 x, __global int8 *signp);
01288 double16 __attribute__((overloadable)) lgamma_r(double16 x, __global int16 *signp);
01289 double __attribute__((overloadable)) lgamma_r(double x, __local int *signp);
01290 double2 __attribute__((overloadable)) lgamma_r(double2 x, __local int2 *signp);
01291 double3 __attribute__((overloadable)) lgamma_r(double3 x, __local int3 *signp);
01292 double4 __attribute__((overloadable)) lgamma_r(double4 x, __local int4 *signp);
01293 double8 __attribute__((overloadable)) lgamma_r(double8 x, __local int8 *signp);
01294 double16 __attribute__((overloadable)) lgamma_r(double16 x, __local int16 *signp);
01295 double __attribute__((overloadable)) lgamma_r(double x, __private int *signp);
01296 double2 __attribute__((overloadable)) lgamma_r(double2 x, __private int2 *signp);
01297 double3 __attribute__((overloadable)) lgamma_r(double3 x, __private int3 *signp);
01298 double4 __attribute__((overloadable)) lgamma_r(double4 x, __private int4 *signp);
01299 double8 __attribute__((overloadable)) lgamma_r(double8 x, __private int8 *signp);
01300 double16 __attribute__((overloadable)) lgamma_r(double16 x, __private int16 *signp);
01301 //half __attribute__((overloadable)) lgamma_r(half x, __global int *signp);
01302 //half2 __attribute__((overloadable)) lgamma_r(half2 x, __global int2 *signp);
01303 //half3 __attribute__((overloadable)) lgamma_r(half3 x, __global int3 *signp);
01304 //half4 __attribute__((overloadable)) lgamma_r(half4 x, __global int4 *signp);
01305 //half8 __attribute__((overloadable)) lgamma_r(half8 x, __global int8 *signp);
01306 //half16 __attribute__((overloadable)) lgamma_r(half16 x, __global int16 *signp);
01307 //half __attribute__((overloadable)) lgamma_r(half x, __local int *signp);
01308 //half2 __attribute__((overloadable)) lgamma_r(half2 x, __local int2 *signp);
01309 //half3 __attribute__((overloadable)) lgamma_r(half3 x, __local int3 *signp);
01310 //half4 __attribute__((overloadable)) lgamma_r(half4 x, __local int4 *signp);
01311 //half8 __attribute__((overloadable)) lgamma_r(half8 x, __local int8 *signp);
01312 //half16 __attribute__((overloadable)) lgamma_r(half16 x, __local int16 *signp);
01313 //half __attribute__((overloadable)) lgamma_r(half x, __private int *signp);
01314 //half2 __attribute__((overloadable)) lgamma_r(half2 x, __private int2 *signp);
01315 //half3 __attribute__((overloadable)) lgamma_r(half3 x, __private int3 *signp);
01316 //half4 __attribute__((overloadable)) lgamma_r(half4 x, __private int4 *signp);
01317 //half8 __attribute__((overloadable)) lgamma_r(half8 x, __private int8 *signp);
01318 //half16 __attribute__((overloadable)) lgamma_r(half16 x, __private int16 *signp);
01319
01320 /**
01321 * Compute natural logarithm.
01322 */
01323 float const_func __attribute__((overloadable)) log(float);
01324 float2 const_func __attribute__((overloadable)) log(float2);
01325 float3 const_func __attribute__((overloadable)) log(float3);
01326 float4 const_func __attribute__((overloadable)) log(float4);
01327 float8 const_func __attribute__((overloadable)) log(float8);
01328 float16 const_func __attribute__((overloadable)) log(float16);
01329 double const_func __attribute__((overloadable)) log(double);
01330 double2 const_func __attribute__((overloadable)) log(double2);
01331 double3 const_func __attribute__((overloadable)) log(double3);
01332 double4 const_func __attribute__((overloadable)) log(double4);
01333 double8 const_func __attribute__((overloadable)) log(double8);
01334 double16 const_func __attribute__((overloadable)) log(double16);
01335 //half const_func __attribute__((overloadable)) log(half);
01336 //half2 const_func __attribute__((overloadable)) log(half2);
01337 //half3 const_func __attribute__((overloadable)) log(half3);
01338 //half4 const_func __attribute__((overloadable)) log(half4);
01339 //half8 const_func __attribute__((overloadable)) log(half8);
01340 //half16 const_func __attribute__((overloadable)) log(half16);
01341
01342 /**
01343 * Compute a base 2 logarithm.
01344 */
01345 float const_func __attribute__((overloadable)) log2(float);
01346 float2 const_func __attribute__((overloadable)) log2(float2);
01347 float3 const_func __attribute__((overloadable)) log2(float3);
01348 float4 const_func __attribute__((overloadable)) log2(float4);
01349 float8 const_func __attribute__((overloadable)) log2(float8);
01350 float16 const_func __attribute__((overloadable)) log2(float16);
01351 double const_func __attribute__((overloadable)) log2(double);
01352 double2 const_func __attribute__((overloadable)) log2(double2);
01353 double3 const_func __attribute__((overloadable)) log2(double3);
01354 double4 const_func __attribute__((overloadable)) log2(double4);
01355 double8 const_func __attribute__((overloadable)) log2(double8);
01356 double16 const_func __attribute__((overloadable)) log2(double16);
01357 //half const_func __attribute__((overloadable)) log2(half);
```

```
01358 //half2 const_func __attribute__((overloadable)) log2(half2);
01359 //half3 const_func __attribute__((overloadable)) log2(half3);
01360 //half4 const_func __attribute__((overloadable)) log2(half4);
01361 //half8 const_func __attribute__((overloadable)) log2(half8);
01362 //half16 const_func __attribute__((overloadable)) log2(half16);
01363
01364 /**
01365 * Compute a base 10 logarithm.
01366 */
01367 float const_func __attribute__((overloadable)) log10(float);
01368 float2 const_func __attribute__((overloadable)) log10(float2);
01369 float3 const_func __attribute__((overloadable)) log10(float3);
01370 float4 const_func __attribute__((overloadable)) log10(float4);
01371 float8 const_func __attribute__((overloadable)) log10(float8);
01372 float16 const_func __attribute__((overloadable)) log10(float16);
01373 double const_func __attribute__((overloadable)) log10(double);
01374 double2 const_func __attribute__((overloadable)) log10(double2);
01375 double3 const_func __attribute__((overloadable)) log10(double3);
01376 double4 const_func __attribute__((overloadable)) log10(double4);
01377 double8 const_func __attribute__((overloadable)) log10(double8);
01378 double16 const_func __attribute__((overloadable)) log10(double16);
01379 //half const_func __attribute__((overloadable)) log10(half);
01380 //half2 const_func __attribute__((overloadable)) log10(half2);
01381 //half3 const_func __attribute__((overloadable)) log10(half3);
01382 //half4 const_func __attribute__((overloadable)) log10(half4);
01383 //half8 const_func __attribute__((overloadable)) log10(half8);
01384 //half16 const_func __attribute__((overloadable)) log10(half16);
01385
01386 /**
01387 * Compute a base e logarithm of (1.0 + x).
01388 */
01389 float const_func __attribute__((overloadable)) log1p(float x);
01390 float2 const_func __attribute__((overloadable)) log1p(float2 x);
01391 float3 const_func __attribute__((overloadable)) log1p(float3 x);
01392 float4 const_func __attribute__((overloadable)) log1p(float4 x);
01393 float8 const_func __attribute__((overloadable)) log1p(float8 x);
01394 float16 const_func __attribute__((overloadable)) log1p(float16 x);
01395 double const_func __attribute__((overloadable)) log1p(double x);
01396 double2 const_func __attribute__((overloadable)) log1p(double2 x);
01397 double3 const_func __attribute__((overloadable)) log1p(double3 x);
01398 double4 const_func __attribute__((overloadable)) log1p(double4 x);
01399 double8 const_func __attribute__((overloadable)) log1p(double8 x);
01400 double16 const_func __attribute__((overloadable)) log1p(double16 x);
01401 //half const_func __attribute__((overloadable)) log1p(half x);
01402 //half2 const_func __attribute__((overloadable)) log1p(half2 x);
01403 //half3 const_func __attribute__((overloadable)) log1p(half3 x);
01404 //half4 const_func __attribute__((overloadable)) log1p(half4 x);
01405 //half8 const_func __attribute__((overloadable)) log1p(half8 x);
01406 //half16 const_func __attribute__((overloadable)) log1p(half16 x);
01407
01408 /**
01409 * Compute the exponent of x, which is the integral
01410 * part of logr | x |.
01411 */
01412 float const_func __attribute__((overloadable)) logb(float x);
01413 float2 const_func __attribute__((overloadable)) logb(float2 x);
01414 float3 const_func __attribute__((overloadable)) logb(float3 x);
01415 float4 const_func __attribute__((overloadable)) logb(float4 x);
01416 float8 const_func __attribute__((overloadable)) logb(float8 x);
01417 float16 const_func __attribute__((overloadable)) logb(float16 x);
01418 double const_func __attribute__((overloadable)) logb(double x);
01419 double2 const_func __attribute__((overloadable)) logb(double2 x);
01420 double3 const_func __attribute__((overloadable)) logb(double3 x);
01421 double4 const_func __attribute__((overloadable)) logb(double4 x);
01422 double8 const_func __attribute__((overloadable)) logb(double8 x);
01423 double16 const_func __attribute__((overloadable)) logb(double16 x);
01424 //half const_func __attribute__((overloadable)) logb(half x);
01425 //half2 const_func __attribute__((overloadable)) logb(half2 x);
01426 //half3 const_func __attribute__((overloadable)) logb(half3 x);
01427 //half4 const_func __attribute__((overloadable)) logb(half4 x);
01428 //half8 const_func __attribute__((overloadable)) logb(half8 x);
01429 //half16 const_func __attribute__((overloadable)) logb(half16 x);
01430
01431 /**
01432 * mad approximates a * b + c.  Whether or how the
01433 * product of a * b is rounded and how supernormal or
01434 * subnormal intermediate products are handled is not
01435 * defined.  mad is intended to be used where speed is
01436 * preferred over accuracy.
01437 */
01438 float const_func __attribute__((overloadable)) mad(float a, float b, float c);
01439 float2 const_func __attribute__((overloadable)) mad(float2 a, float2 b, float2 c);
01440 float3 const_func __attribute__((overloadable)) mad(float3 a, float3 b, float3 c);
01441 float4 const_func __attribute__((overloadable)) mad(float4 a, float4 b, float4 c);
01442 float8 const_func __attribute__((overloadable)) mad(float8 a, float8 b, float8 c);
01443 float16 const_func __attribute__((overloadable)) mad(float16 a, float16 b, float16 c);
01444 double const_func __attribute__((overloadable)) mad(double a, double b, double c);
```

```
01445 double2 const_func __attribute__((overloadable)) mad(double2 a, double2 b, double2 c);
01446 double3 const_func __attribute__((overloadable)) mad(double3 a, double3 b, double3 c);
01447 double4 const_func __attribute__((overloadable)) mad(double4 a, double4 b, double4 c);
01448 double8 const_func __attribute__((overloadable)) mad(double8 a, double8 b, double8 c);
01449 double16 const_func __attribute__((overloadable)) mad(double16 a, double16 b, double16 c);
01450 //half const_func __attribute__((overloadable)) mad(half a, half b, half c);
01451 //half2 const_func __attribute__((overloadable)) mad(half2 a, half2 b, half2 c);
01452 //half3 const_func __attribute__((overloadable)) mad(half3 a, half3 b, half3 c);
01453 //half4 const_func __attribute__((overloadable)) mad(half4 a, half4 b, half4 c);
01454 //half8 const_func __attribute__((overloadable)) mad(half8 a, half8 b, half8 c);
01455 //half16 const_func __attribute__((overloadable)) mad(half16 a, half16 b, half16 c);
01456
01457 /**
01458 * Returns x if | x | > | y |, y if | y | > | x |, otherwise
01459 * fmax(x, y).
01460 */
01461 float const_func __attribute__((overloadable)) maxmag(float x, float y);
01462 float2 const_func __attribute__((overloadable)) maxmag(float2 x, float2 y);
01463 float3 const_func __attribute__((overloadable)) maxmag(float3 x, float3 y);
01464 float4 const_func __attribute__((overloadable)) maxmag(float4 x, float4 y);
01465 float8 const_func __attribute__((overloadable)) maxmag(float8 x, float8 y);
01466 float16 const_func __attribute__((overloadable)) maxmag(float16 x, float16 y);
01467 double const_func __attribute__((overloadable)) maxmag(double x, double y);
01468 double2 const_func __attribute__((overloadable)) maxmag(double2 x, double2 y);
01469 double3 const_func __attribute__((overloadable)) maxmag(double3 x, double3 y);
01470 double4 const_func __attribute__((overloadable)) maxmag(double4 x, double4 y);
01471 double8 const_func __attribute__((overloadable)) maxmag(double8 x, double8 y);
01472 double16 const_func __attribute__((overloadable)) maxmag(double16 x, double16 y);
01473 //half const_func __attribute__((overloadable)) maxmag(half x, half y);
01474 //half2 const_func __attribute__((overloadable)) maxmag(half2 x, half2 y);
01475 //half3 const_func __attribute__((overloadable)) maxmag(half3 x, half3 y);
01476 //half4 const_func __attribute__((overloadable)) maxmag(half4 x, half4 y);
01477 //half8 const_func __attribute__((overloadable)) maxmag(half8 x, half8 y);
01478 //half16 const_func __attribute__((overloadable)) maxmag(half16 x, half16 y);
01479
01480 /**
01481 * Returns x if | x | < | y |, y if | y | < | x |, otherwise
01482 * fmin(x, y).
01483 */
01484 float const_func __attribute__((overloadable)) minmag(float x, float y);
01485 float2 const_func __attribute__((overloadable)) minmag(float2 x, float2 y);
01486 float3 const_func __attribute__((overloadable)) minmag(float3 x, float3 y);
01487 float4 const_func __attribute__((overloadable)) minmag(float4 x, float4 y);
01488 float8 const_func __attribute__((overloadable)) minmag(float8 x, float8 y);
01489 float16 const_func __attribute__((overloadable)) minmag(float16 x, float16 y);
01490 double const_func __attribute__((overloadable)) minmag(double x, double y);
01491 double2 const_func __attribute__((overloadable)) minmag(double2 x, double2 y);
01492 double3 const_func __attribute__((overloadable)) minmag(double3 x, double3 y);
01493 double4 const_func __attribute__((overloadable)) minmag(double4 x, double4 y);
01494 double8 const_func __attribute__((overloadable)) minmag(double8 x, double8 y);
01495 double16 const_func __attribute__((overloadable)) minmag(double16 x, double16 y);
01496 //half const_func __attribute__((overloadable)) minmag(half x, half y);
01497 //half2 const_func __attribute__((overloadable)) minmag(half2 x, half2 y);
01498 //half3 const_func __attribute__((overloadable)) minmag(half3 x, half3 y);
01499 //half4 const_func __attribute__((overloadable)) minmag(half4 x, half4 y);
01500 //half8 const_func __attribute__((overloadable)) minmag(half8 x, half8 y);
01501 //half16 const_func __attribute__((overloadable)) minmag(half16 x, half16 y);
01502
01503 /**
01504 * Decompose a floating-point number.  The modf
01505 * function breaks the argument x into integral and
01506 * fractional parts, each of which has the same sign as
01507 * the argument.  It stores the integral part in the object
01508 * pointed to by iptr.
01509 */
01510 float __attribute__((overloadable)) modf(float x, __global float *iptr);
01511 float2 __attribute__((overloadable)) modf(float2 x, __global float2 *iptr);
01512 float3 __attribute__((overloadable)) modf(float3 x, __global float3 *iptr);
01513 float4 __attribute__((overloadable)) modf(float4 x, __global float4 *iptr);
01514 float8 __attribute__((overloadable)) modf(float8 x, __global float8 *iptr);
01515 float16 __attribute__((overloadable)) modf(float16 x, __global float16 *iptr);
01516 float __attribute__((overloadable)) modf(float x, __local float *iptr);
01517 float2 __attribute__((overloadable)) modf(float2 x, __local float2 *iptr);
01518 float3 __attribute__((overloadable)) modf(float3 x, __local float3 *iptr);
01519 float4 __attribute__((overloadable)) modf(float4 x, __local float4 *iptr);
01520 float8 __attribute__((overloadable)) modf(float8 x, __local float8 *iptr);
01521 float16 __attribute__((overloadable)) modf(float16 x, __local float16 *iptr);
01522 float __attribute__((overloadable)) modf(float x, __private float *iptr);
01523 float2 __attribute__((overloadable)) modf(float2 x, __private float2 *iptr);
01524 float3 __attribute__((overloadable)) modf(float3 x, __private float3 *iptr);
01525 float4 __attribute__((overloadable)) modf(float4 x, __private float4 *iptr);
01526 float8 __attribute__((overloadable)) modf(float8 x, __private float8 *iptr);
01527 float16 __attribute__((overloadable)) modf(float16 x, __private float16 *iptr);
01528 double __attribute__((overloadable)) modf(double x, __global double *iptr);
01529 double2 __attribute__((overloadable)) modf(double2 x, __global double2 *iptr);
01530 double3 __attribute__((overloadable)) modf(double3 x, __global double3 *iptr);
01531 double4 __attribute__((overloadable)) modf(double4 x, __global double4 *iptr);
```

```
01532 double8 __attribute__((overloadable)) modf(double8 x, __global double8 *iptr);
01533 double16 __attribute__((overloadable)) modf(double16 x, __global double16 *iptr);
01534 double __attribute__((overloadable)) modf(double x, __local double *iptr);
01535 double2 __attribute__((overloadable)) modf(double2 x, __local double2 *iptr);
01536 double3 __attribute__((overloadable)) modf(double3 x, __local double3 *iptr);
01537 double4 __attribute__((overloadable)) modf(double4 x, __local double4 *iptr);
01538 double8 __attribute__((overloadable)) modf(double8 x, __local double8 *iptr);
01539 double16 __attribute__((overloadable)) modf(double16 x, __local double16 *iptr);
01540 double __attribute__((overloadable)) modf(double x, __private double *iptr);
01541 double2 __attribute__((overloadable)) modf(double2 x, __private double2 *iptr);
01542 double3 __attribute__((overloadable)) modf(double3 x, __private double3 *iptr);
01543 double4 __attribute__((overloadable)) modf(double4 x, __private double4 *iptr);
01544 double8 __attribute__((overloadable)) modf(double8 x, __private double8 *iptr);
01545 double16 __attribute__((overloadable)) modf(double16 x, __private double16 *iptr);
01546 //half __attribute__((overloadable)) modf(half x, __global half *iptr);
01547 //half2 __attribute__((overloadable)) modf(half2 x, __global half2 *iptr);
01548 //half3 __attribute__((overloadable)) modf(half3 x, __global half3 *iptr);
01549 //half4 __attribute__((overloadable)) modf(half4 x, __global half4 *iptr);
01550 //half8 __attribute__((overloadable)) modf(half8 x, __global half8 *iptr);
01551 //half16 __attribute__((overloadable)) modf(half16 x, __global half16 *iptr);
01552 //half __attribute__((overloadable)) modf(half x, __local half *iptr);
01553 //half2 __attribute__((overloadable)) modf(half2 x, __local half2 *iptr);
01554 //half3 __attribute__((overloadable)) modf(half3 x, __local half3 *iptr);
01555 //half4 __attribute__((overloadable)) modf(half4 x, __local half4 *iptr);
01556 //half8 __attribute__((overloadable)) modf(half8 x, __local half8 *iptr);
01557 //half16 __attribute__((overloadable)) modf(half16 x, __local half16 *iptr);
01558 //half __attribute__((overloadable)) modf(half x, __private half *iptr);
01559 //half2 __attribute__((overloadable)) modf(half2 x, __private half2 *iptr);
01560 //half3 __attribute__((overloadable)) modf(half3 x, __private half3 *iptr);
01561 //half4 __attribute__((overloadable)) modf(half4 x, __private half4 *iptr);
01562 //half8 __attribute__((overloadable)) modf(half8 x, __private half8 *iptr);
01563 //half16 __attribute__((overloadable)) modf(half16 x, __private half16 *iptr);
01564
01565 /**
01566 * Returns a quiet NaN. The nancode may be placed
01567 * in the significand of the resulting NaN.
01568 */
01569 float const_func __attribute__((overloadable)) nan(uint nancode);
01570 float2 const_func __attribute__((overloadable)) nan(uint2 nancode);
01571 float3 const_func __attribute__((overloadable)) nan(uint3 nancode);
01572 float4 const_func __attribute__((overloadable)) nan(uint4 nancode);
01573 float8 const_func __attribute__((overloadable)) nan(uint8 nancode);
01574 float16 const_func __attribute__((overloadable)) nan(uint16 nancode);
01575 double const_func __attribute__((overloadable)) nan(ulong nancode);
01576 double2 const_func __attribute__((overloadable)) nan(ulong2 nancode);
01577 double3 const_func __attribute__((overloadable)) nan(ulong3 nancode);
01578 double4 const_func __attribute__((overloadable)) nan(ulong4 nancode);
01579 double8 const_func __attribute__((overloadable)) nan(ulong8 nancode);
01580 double16 const_func __attribute__((overloadable)) nan(ulong16 nancode);
01581 //half const_func __attribute__((overloadable)) nan(ushort nancode);
01582 //half2 const_func __attribute__((overloadable)) nan(ushort2 nancode);
01583 //half3 const_func __attribute__((overloadable)) nan(ushort3 nancode);
01584 //half4 const_func __attribute__((overloadable)) nan(ushort4 nancode);
01585 //half8 const_func __attribute__((overloadable)) nan(ushort8 nancode);
01586 //half16 const_func __attribute__((overloadable)) nan(ushort16 nancode);
01587
01588 /**
01589 * Computes the next representable single-precision
01590 * floating-point value following x in the direction of
01591 * y.  Thus, if y is less than x, nextafter() returns the
01592 * largest representable floating-point number less
01593 * than x.
01594 */
01595 float const_func __attribute__((overloadable)) nextafter(float x, float y);
01596 float2 const_func __attribute__((overloadable)) nextafter(float2 x, float2 y);
01597 float3 const_func __attribute__((overloadable)) nextafter(float3 x, float3 y);
01598 float4 const_func __attribute__((overloadable)) nextafter(float4 x, float4 y);
01599 float8 const_func __attribute__((overloadable)) nextafter(float8 x, float8 y);
01600 float16 const_func __attribute__((overloadable)) nextafter(float16 x, float16 y);
01601 double const_func __attribute__((overloadable)) nextafter(double x, double y);
01602 double2 const_func __attribute__((overloadable)) nextafter(double2 x, double2 y);
01603 double3 const_func __attribute__((overloadable)) nextafter(double3 x, double3 y);
01604 double4 const_func __attribute__((overloadable)) nextafter(double4 x, double4 y);
01605 double8 const_func __attribute__((overloadable)) nextafter(double8 x, double8 y);
01606 double16 const_func __attribute__((overloadable)) nextafter(double16 x, double16 y);
01607 //half const_func __attribute__((overloadable)) nextafter(half x, half y);
01608 //half2 const_func __attribute__((overloadable)) nextafter(half2 x, half2 y);
01609 //half3 const_func __attribute__((overloadable)) nextafter(half3 x, half3 y);
01610 //half4 const_func __attribute__((overloadable)) nextafter(half4 x, half4 y);
01611 //half8 const_func __attribute__((overloadable)) nextafter(half8 x, half8 y);
01612 //half16 const_func __attribute__((overloadable)) nextafter(half16 x, half16 y);
01613
01614 /**
01615 * Compute x to the power y.
01616 */
01617 double const_func __attribute__((overloadable)) pow(double x, double y);
01618 double2 const_func __attribute__((overloadable)) pow(double2 x, double2 y);
```

```
01619 double3 const_func __attribute__((overloadable)) pow(double3 x, double3 y);
01620 double4 const_func __attribute__((overloadable)) pow(double4 x, double4 y);
01621 double8 const_func __attribute__((overloadable)) pow(double8 x, double8 y);
01622 double16 const_func __attribute__((overloadable)) pow(double16 x, double16 y);
01623 float const_func __attribute__((overloadable)) pow(float x, float y);
01624 float2 const_func __attribute__((overloadable)) pow(float2 x, float2 y);
01625 float3 const_func __attribute__((overloadable)) pow(float3 x, float3 y);
01626 float4 const_func __attribute__((overloadable)) pow(float4 x, float4 y);
01627 float8 const_func __attribute__((overloadable)) pow(float8 x, float8 y);
01628 float16 const_func __attribute__((overloadable)) pow(float16 x, float16 y);
01629 //half const_func __attribute__((overloadable)) pow(half x, half y);
01630 //half2 const_func __attribute__((overloadable)) pow(half2 x, half2 y);
01631 //half3 const_func __attribute__((overloadable)) pow(half3 x, half3 y);
01632 //half4 const_func __attribute__((overloadable)) pow(half4 x, half4 y);
01633 //half8 const_func __attribute__((overloadable)) pow(half8 x, half8 y);
01634 //half16 const_func __attribute__((overloadable)) pow(half16 x, half16 y);
01635
01636 /**
01637 * Compute x to the power y, where y is an integer.
01638 */
01639 float const_func __attribute__((overloadable)) pown(float x, int y);
01640 float2 const_func __attribute__((overloadable)) pown(float2 x, int2 y);
01641 float3 const_func __attribute__((overloadable)) pown(float3 x, int3 y);
01642 float4 const_func __attribute__((overloadable)) pown(float4 x, int4 y);
01643 float8 const_func __attribute__((overloadable)) pown(float8 x, int8 y);
01644 float16 const_func __attribute__((overloadable)) pown(float16 x, int16 y);
01645 double const_func __attribute__((overloadable)) pown(double x, int y);
01646 double2 const_func __attribute__((overloadable)) pown(double2 x, int2 y);
01647 double3 const_func __attribute__((overloadable)) pown(double3 x, int3 y);
01648 double4 const_func __attribute__((overloadable)) pown(double4 x, int4 y);
01649 double8 const_func __attribute__((overloadable)) pown(double8 x, int8 y);
01650 double16 const_func __attribute__((overloadable)) pown(double16 x, int16 y);
01651 //half const_func __attribute__((overloadable)) pown(half x, int y);
01652 //half2 const_func __attribute__((overloadable)) pown(half2 x, int2 y);
01653 //half3 const_func __attribute__((overloadable)) pown(half3 x, int3 y);
01654 //half4 const_func __attribute__((overloadable)) pown(half4 x, int4 y);
01655 //half8 const_func __attribute__((overloadable)) pown(half8 x, int8 y);
01656 //half16 const_func __attribute__((overloadable)) pown(half16 x, int16 y);
01657
01658 /**
01659 * Compute x to the power y, where x is >= 0.
01660 */
01661 float const_func __attribute__((overloadable)) powr(float x, float y);
01662 float2 const_func __attribute__((overloadable)) powr(float2 x, float2 y);
01663 float3 const_func __attribute__((overloadable)) powr(float3 x, float3 y);
01664 float4 const_func __attribute__((overloadable)) powr(float4 x, float4 y);
01665 float8 const_func __attribute__((overloadable)) powr(float8 x, float8 y);
01666 float16 const_func __attribute__((overloadable)) powr(float16 x, float16 y);
01667 double const_func __attribute__((overloadable)) powr(double x, double y);
01668 double2 const_func __attribute__((overloadable)) powr(double2 x, double2 y);
01669 double3 const_func __attribute__((overloadable)) powr(double3 x, double3 y);
01670 double4 const_func __attribute__((overloadable)) powr(double4 x, double4 y);
01671 double8 const_func __attribute__((overloadable)) powr(double8 x, double8 y);
01672 double16 const_func __attribute__((overloadable)) powr(double16 x, double16 y);
01673 //half const_func __attribute__((overloadable)) powr(half x, half y);
01674 //half2 const_func __attribute__((overloadable)) powr(half2 x, half2 y);
01675 //half3 const_func __attribute__((overloadable)) powr(half3 x, half3 y);
01676 //half4 const_func __attribute__((overloadable)) powr(half4 x, half4 y);
01677 //half8 const_func __attribute__((overloadable)) powr(half8 x, half8 y);
01678 //half16 const_func __attribute__((overloadable)) powr(half16 x, half16 y);
01679
01680 /**
01681 * Compute the value r such that r = x - n*y, where n
01682 * is the integer nearest the exact value of x/y.  If there
01683 * are two integers closest to x/y, n shall be the even
01684 * one.  If r is zero, it is given the same sign as x.
01685 */
01686 float const_func __attribute__((overloadable)) remainder(float x, float y);
01687 float2 const_func __attribute__((overloadable)) remainder(float2 x, float2 y);
01688 float3 const_func __attribute__((overloadable)) remainder(float3 x, float3 y);
01689 float4 const_func __attribute__((overloadable)) remainder(float4 x, float4 y);
01690 float8 const_func __attribute__((overloadable)) remainder(float8 x, float8 y);
01691 float16 const_func __attribute__((overloadable)) remainder(float16 x, float16 y);
01692 double const_func __attribute__((overloadable)) remainder(double x, double y);
01693 double2 const_func __attribute__((overloadable)) remainder(double2 x, double2 y);
01694 double3 const_func __attribute__((overloadable)) remainder(double3 x, double3 y);
01695 double4 const_func __attribute__((overloadable)) remainder(double4 x, double4 y);
01696 double8 const_func __attribute__((overloadable)) remainder(double8 x, double8 y);
01697 double16 const_func __attribute__((overloadable)) remainder(double16 x, double16 y);
01698 //half const_func __attribute__((overloadable)) remainder(half x, half y);
01699 //half2 const_func __attribute__((overloadable)) remainder(half2 x, half2 y);
01700 //half3 const_func __attribute__((overloadable)) remainder(half3 x, half3 y);
01701 //half4 const_func __attribute__((overloadable)) remainder(half4 x, half4 y);
01702 //half8 const_func __attribute__((overloadable)) remainder(half8 x, half8 y);
01703 //half16 const_func __attribute__((overloadable)) remainder(half16 x, half16 y);
01704
01705 /**
```

```
01706 * The remquo function computes the value r such
01707 * that r = x - n*y, where n is the integer nearest the
01708 * exact value of x/y.  If there are two integers closest
01709 * to x/y, n shall be the even one.  If r is zero, it is
01710 * given the same sign as x.  This is the same value
01711 * that is returned by the remainder function.
01712 * remquo also calculates the lower seven bits of the
01713 * integral quotient x/y, and gives that value the same
01714 * sign as x/y.  It stores this signed value in the object
01715 * pointed to by quo.
01716 */
01717 float __attribute__((overloadable)) remquo(float x, float y, __global int *quo);
01718 float2 __attribute__((overloadable)) remquo(float2 x, float2 y, __global int2 *quo);
01719 float3 __attribute__((overloadable)) remquo(float3 x, float3 y, __global int3 *quo);
01720 float4 __attribute__((overloadable)) remquo(float4 x, float4 y, __global int4 *quo);
01721 float8 __attribute__((overloadable)) remquo(float8 x, float8 y, __global int8 *quo);
01722 float16 __attribute__((overloadable)) remquo(float16 x, float16 y, __global int16 *quo);
01723 float __attribute__((overloadable)) remquo(float x, float y, __local int *quo);
01724 float2 __attribute__((overloadable)) remquo(float2 x, float2 y, __local int2 *quo);
01725 float3 __attribute__((overloadable)) remquo(float3 x, float3 y, __local int3 *quo);
01726 float4 __attribute__((overloadable)) remquo(float4 x, float4 y, __local int4 *quo);
01727 float8 __attribute__((overloadable)) remquo(float8 x, float8 y, __local int8 *quo);
01728 float16 __attribute__((overloadable)) remquo(float16 x, float16 y, __local int16 *quo);
01729 float __attribute__((overloadable)) remquo(float x, float y, __private int *quo);
01730 float2 __attribute__((overloadable)) remquo(float2 x, float2 y, __private int2 *quo);
01731 float3 __attribute__((overloadable)) remquo(float3 x, float3 y, __private int3 *quo);
01732 float4 __attribute__((overloadable)) remquo(float4 x, float4 y, __private int4 *quo);
01733 float8 __attribute__((overloadable)) remquo(float8 x, float8 y, __private int8 *quo);
01734 float16 __attribute__((overloadable)) remquo(float16 x, float16 y, __private int16 *quo);
01735 double __attribute__((overloadable)) remquo(double x, double y, __global int *quo);
01736 double2 __attribute__((overloadable)) remquo(double2 x, double2 y, __global int2 *quo);
01737 double3 __attribute__((overloadable)) remquo(double3 x, double3 y, __global int3 *quo);
01738 double4 __attribute__((overloadable)) remquo(double4 x, double4 y, __global int4 *quo);
01739 double8 __attribute__((overloadable)) remquo(double8 x, double8 y, __global int8 *quo);
01740 double16 __attribute__((overloadable)) remquo(double16 x, double16 y, __global int16 *quo);
01741 double __attribute__((overloadable)) remquo(double x, double y, __local int *quo);
01742 double2 __attribute__((overloadable)) remquo(double2 x, double2 y, __local int2 *quo);
01743 double3 __attribute__((overloadable)) remquo(double3 x, double3 y, __local int3 *quo);
01744 double4 __attribute__((overloadable)) remquo(double4 x, double4 y, __local int4 *quo);
01745 double8 __attribute__((overloadable)) remquo(double8 x, double8 y, __local int8 *quo);
01746 double16 __attribute__((overloadable)) remquo(double16 x, double16 y, __local int16 *quo);
01747 double __attribute__((overloadable)) remquo(double x, double y, __private int *quo);
01748 double2 __attribute__((overloadable)) remquo(double2 x, double2 y, __private int2 *quo);
01749 double3 __attribute__((overloadable)) remquo(double3 x, double3 y, __private int3 *quo);
01750 double4 __attribute__((overloadable)) remquo(double4 x, double4 y, __private int4 *quo);
01751 double8 __attribute__((overloadable)) remquo(double8 x, double8 y, __private int8 *quo);
01752 double16 __attribute__((overloadable)) remquo(double16 x, double16 y, __private int16 *quo);
01753 //half __attribute__((overloadable)) remquo(half x, half y, __global int *quo);
01754 //half2 __attribute__((overloadable)) remquo(half2 x, half2 y, __global int2 *quo);
01755 //half3 __attribute__((overloadable)) remquo(half3 x, half3 y, __global int3 *quo);
01756 //half4 __attribute__((overloadable)) remquo(half4 x, half4 y, __global int4 *quo);
01757 //half8 __attribute__((overloadable)) remquo(half8 x, half8 y, __global int8 *quo);
01758 //half16 __attribute__((overloadable)) remquo(half16 x, half16 y, __global int16 *quo);
01759 //half __attribute__((overloadable)) remquo(half x, half y, __local int *quo);
01760 //half2 __attribute__((overloadable)) remquo(half2 x, half2 y, __local int2 *quo);
01761 //half3 __attribute__((overloadable)) remquo(half3 x, half3 y, __local int3 *quo);
01762 //half4 __attribute__((overloadable)) remquo(half4 x, half4 y, __local int4 *quo);
01763 //half8 __attribute__((overloadable)) remquo(half8 x, half8 y, __local int8 *quo);
01764 //half16 __attribute__((overloadable)) remquo(half16 x, half16 y, __local int16 *quo);
01765 //half __attribute__((overloadable)) remquo(half x, half y, __private int *quo);
01766 //half2 __attribute__((overloadable)) remquo(half2 x, half2 y, __private int2 *quo);
01767 //half3 __attribute__((overloadable)) remquo(half3 x, half3 y, __private int3 *quo);
01768 //half4 __attribute__((overloadable)) remquo(half4 x, half4 y, __private int4 *quo);
01769 //half8 __attribute__((overloadable)) remquo(half8 x, half8 y, __private int8 *quo);
01770 //half16 __attribute__((overloadable)) remquo(half16 x, half16 y, __private int16 *quo);
01771
01772 /**
01773 * Round to integral value (using round to nearest
01774 * even rounding mode) in floating-point format.
01775 * Refer to section 7.1 for description of rounding
01776 * modes.
01777 */
01778 float const_func __attribute__((overloadable)) rint(float);
01779 float2 const_func __attribute__((overloadable)) rint(float2);
01780 float3 const_func __attribute__((overloadable)) rint(float3);
01781 float4 const_func __attribute__((overloadable)) rint(float4);
01782 float8 const_func __attribute__((overloadable)) rint(float8);
01783 float16 const_func __attribute__((overloadable)) rint(float16);
01784 double const_func __attribute__((overloadable)) rint(double);
01785 double2 const_func __attribute__((overloadable)) rint(double2);
01786 double3 const_func __attribute__((overloadable)) rint(double3);
01787 double4 const_func __attribute__((overloadable)) rint(double4);
01788 double8 const_func __attribute__((overloadable)) rint(double8);
01789 double16 const_func __attribute__((overloadable)) rint(double16);
01790 //half const_func __attribute__((overloadable)) rint(half);
01791 //half2 const_func __attribute__((overloadable)) rint(half2);
01792 //half3 const_func __attribute__((overloadable)) rint(half3);
```

```
01793 //half4 const_func __attribute__((overloadable)) rint(half4);
01794 //half8 const_func __attribute__((overloadable)) rint(half8);
01795 //half16 const_func __attribute__((overloadable)) rint(half16);
01796
01797 /**
01798 * Compute x to the power 1/y.
01799 */
01800 float const_func __attribute__((overloadable)) rootn(float x, int y);
01801 float2 const_func __attribute__((overloadable)) rootn(float2 x, int2 y);
01802 float3 const_func __attribute__((overloadable)) rootn(float3 x, int3 y);
01803 float4 const_func __attribute__((overloadable)) rootn(float4 x, int4 y);
01804 float8 const_func __attribute__((overloadable)) rootn(float8 x, int8 y);
01805 float16 const_func __attribute__((overloadable)) rootn(float16 x, int16 y);
01806 double const_func __attribute__((overloadable)) rootn(double x, int y);
01807 double2 const_func __attribute__((overloadable)) rootn(double2 x, int2 y);
01808 double3 const_func __attribute__((overloadable)) rootn(double3 x, int3 y);
01809 double4 const_func __attribute__((overloadable)) rootn(double4 x, int4 y);
01810 double8 const_func __attribute__((overloadable)) rootn(double8 x, int8 y);
01811 double16 const_func __attribute__((overloadable)) rootn(double16 x, int16 y);
01812 //half const_func __attribute__((overloadable)) rootn(half x, int y);
01813 //half2 const_func __attribute__((overloadable)) rootn(half2 x, int2 y);
01814 //half3 const_func __attribute__((overloadable)) rootn(half3 x, int3 y);
01815 //half4 const_func __attribute__((overloadable)) rootn(half4 x, int4 y);
01816 //half8 const_func __attribute__((overloadable)) rootn(half8 x, int8 y);
01817 //half16 const_func __attribute__((overloadable)) rootn(half16 x, int16 y);
01818
01819 /**
01820 * Return the integral value nearest to x rounding
01821 * halfway cases away from zero, regardless of the
01822 * current rounding direction.
01823 */
01824 float const_func __attribute__((overloadable)) round(float x);
01825 float2 const_func __attribute__((overloadable)) round(float2 x);
01826 float3 const_func __attribute__((overloadable)) round(float3 x);
01827 float4 const_func __attribute__((overloadable)) round(float4 x);
01828 float8 const_func __attribute__((overloadable)) round(float8 x);
01829 float16 const_func __attribute__((overloadable)) round(float16 x);
01830 double const_func __attribute__((overloadable)) round(double x);
01831 double2 const_func __attribute__((overloadable)) round(double2 x);
01832 double3 const_func __attribute__((overloadable)) round(double3 x);
01833 double4 const_func __attribute__((overloadable)) round(double4 x);
01834 double8 const_func __attribute__((overloadable)) round(double8 x);
01835 double16 const_func __attribute__((overloadable)) round(double16 x);
01836 //half const_func __attribute__((overloadable)) round(half x);
01837 //half2 const_func __attribute__((overloadable)) round(half2 x);
01838 //half3 const_func __attribute__((overloadable)) round(half3 x);
01839 //half4 const_func __attribute__((overloadable)) round(half4 x);
01840 //half8 const_func __attribute__((overloadable)) round(half8 x);
01841 //half16 const_func __attribute__((overloadable)) round(half16 x);
01842
01843 /**
01844 * Compute inverse square root.
01845 */
01846 float const_func __attribute__((overloadable)) rsqrt(float);
01847 float2 const_func __attribute__((overloadable)) rsqrt(float2);
01848 float3 const_func __attribute__((overloadable)) rsqrt(float3);
01849 float4 const_func __attribute__((overloadable)) rsqrt(float4);
01850 float8 const_func __attribute__((overloadable)) rsqrt(float8);
01851 float16 const_func __attribute__((overloadable)) rsqrt(float16);
01852 double const_func __attribute__((overloadable)) rsqrt(double);
01853 double2 const_func __attribute__((overloadable)) rsqrt(double2);
01854 double3 const_func __attribute__((overloadable)) rsqrt(double3);
01855 double4 const_func __attribute__((overloadable)) rsqrt(double4);
01856 double8 const_func __attribute__((overloadable)) rsqrt(double8);
01857 double16 const_func __attribute__((overloadable)) rsqrt(double16);
01858 //half const_func __attribute__((overloadable)) rsqrt(half);
01859 //half2 const_func __attribute__((overloadable)) rsqrt(half2);
01860 //half3 const_func __attribute__((overloadable)) rsqrt(half3);
01861 //half4 const_func __attribute__((overloadable)) rsqrt(half4);
01862 //half8 const_func __attribute__((overloadable)) rsqrt(half8);
01863 //half16 const_func __attribute__((overloadable)) rsqrt(half16);
01864
01865 /**
01866 * Compute sine.
01867 */
01868 double const_func __attribute__((overloadable)) sin(double);
01869 double2 const_func __attribute__((overloadable)) sin(double2);
01870 double3 const_func __attribute__((overloadable)) sin(double3);
01871 double4 const_func __attribute__((overloadable)) sin(double4);
01872 double8 const_func __attribute__((overloadable)) sin(double8);
01873 double16 const_func __attribute__((overloadable)) sin(double16);
01874 float const_func __attribute__((overloadable)) sin(float);
01875 float2 const_func __attribute__((overloadable)) sin(float2);
01876 float3 const_func __attribute__((overloadable)) sin(float3);
01877 float4 const_func __attribute__((overloadable)) sin(float4);
01878 float8 const_func __attribute__((overloadable)) sin(float8);
01879 float16 const_func __attribute__((overloadable)) sin(float16);
```

```
01880 //half const_func __attribute__((overloadable)) sin(half);
01881 //half2 const_func __attribute__((overloadable)) sin(half2);
01882 //half3 const_func __attribute__((overloadable)) sin(half3);
01883 //half4 const_func __attribute__((overloadable)) sin(half4);
01884 //half8 const_func __attribute__((overloadable)) sin(half8);
01885 //half16 const_func __attribute__((overloadable)) sin(half16);
01886
01887 /**
01888 * Compute sine and cosine of x.  The computed sine
01889 * is the return value and computed cosine is returned
01890 * in cosval.
01891 */
01892 float __attribute__((overloadable)) sincos(float x, __global float *cosval);
01893 float2 __attribute__((overloadable)) sincos(float2 x, __global float2 *cosval);
01894 float3 __attribute__((overloadable)) sincos(float3 x, __global float3 *cosval);
01895 float4 __attribute__((overloadable)) sincos(float4 x, __global float4 *cosval);
01896 float8 __attribute__((overloadable)) sincos(float8 x, __global float8 *cosval);
01897 float16 __attribute__((overloadable)) sincos(float16 x, __global float16 *cosval);
01898 float __attribute__((overloadable)) sincos(float x, __local float *cosval);
01899 float2 __attribute__((overloadable)) sincos(float2 x, __local float2 *cosval);
01900 float3 __attribute__((overloadable)) sincos(float3 x, __local float3 *cosval);
01901 float4 __attribute__((overloadable)) sincos(float4 x, __local float4 *cosval);
01902 float8 __attribute__((overloadable)) sincos(float8 x, __local float8 *cosval);
01903 float16 __attribute__((overloadable)) sincos(float16 x, __local float16 *cosval);
01904 float __attribute__((overloadable)) sincos(float x, __private float *cosval);
01905 float2 __attribute__((overloadable)) sincos(float2 x, __private float2 *cosval);
01906 float3 __attribute__((overloadable)) sincos(float3 x, __private float3 *cosval);
01907 float4 __attribute__((overloadable)) sincos(float4 x, __private float4 *cosval);
01908 float8 __attribute__((overloadable)) sincos(float8 x, __private float8 *cosval);
01909 float16 __attribute__((overloadable)) sincos(float16 x, __private float16 *cosval);
01910 double __attribute__((overloadable)) sincos(double x, __global double *cosval);
01911 double2 __attribute__((overloadable)) sincos(double2 x, __global double2 *cosval);
01912 double3 __attribute__((overloadable)) sincos(double3 x, __global double3 *cosval);
01913 double4 __attribute__((overloadable)) sincos(double4 x, __global double4 *cosval);
01914 double8 __attribute__((overloadable)) sincos(double8 x, __global double8 *cosval);
01915 double16 __attribute__((overloadable)) sincos(double16 x, __global double16 *cosval);
01916 double __attribute__((overloadable)) sincos(double x, __local double *cosval);
01917 double2 __attribute__((overloadable)) sincos(double2 x, __local double2 *cosval);
01918 double3 __attribute__((overloadable)) sincos(double3 x, __local double3 *cosval);
01919 double4 __attribute__((overloadable)) sincos(double4 x, __local double4 *cosval);
01920 double8 __attribute__((overloadable)) sincos(double8 x, __local double8 *cosval);
01921 double16 __attribute__((overloadable)) sincos(double16 x, __local double16 *cosval);
01922 double __attribute__((overloadable)) sincos(double x, __private double *cosval);
01923 double2 __attribute__((overloadable)) sincos(double2 x, __private double2 *cosval);
01924 double3 __attribute__((overloadable)) sincos(double3 x, __private double3 *cosval);
01925 double4 __attribute__((overloadable)) sincos(double4 x, __private double4 *cosval);
01926 double8 __attribute__((overloadable)) sincos(double8 x, __private double8 *cosval);
01927 double16 __attribute__((overloadable)) sincos(double16 x, __private double16 *cosval);
01928 //half __attribute__((overloadable)) sincos(half x, __global half *cosval);
01929 //half2 __attribute__((overloadable)) sincos(half2 x, __global half2 *cosval);
01930 //half3 __attribute__((overloadable)) sincos(half3 x, __global half3 *cosval);
01931 //half4 __attribute__((overloadable)) sincos(half4 x, __global half4 *cosval);
01932 //half8 __attribute__((overloadable)) sincos(half8 x, __global half8 *cosval);
01933 //half16 __attribute__((overloadable)) sincos(half16 x, __global half16 *cosval);
01934 //half __attribute__((overloadable)) sincos(half x, __local half *cosval);
01935 //half2 __attribute__((overloadable)) sincos(half2 x, __local half2 *cosval);
01936 //half3 __attribute__((overloadable)) sincos(half3 x, __local half3 *cosval);
01937 //half4 __attribute__((overloadable)) sincos(half4 x, __local half4 *cosval);
01938 //half8 __attribute__((overloadable)) sincos(half8 x, __local half8 *cosval);
01939 //half16 __attribute__((overloadable)) sincos(half16 x, __local half16 *cosval);
01940 //half __attribute__((overloadable)) sincos(half x, __private half *cosval);
01941 //half2 __attribute__((overloadable)) sincos(half2 x, __private half2 *cosval);
01942 //half3 __attribute__((overloadable)) sincos(half3 x, __private half3 *cosval);
01943 //half4 __attribute__((overloadable)) sincos(half4 x, __private half4 *cosval);
01944 //half8 __attribute__((overloadable)) sincos(half8 x, __private half8 *cosval);
01945 //half16 __attribute__((overloadable)) sincos(half16 x, __private half16 *cosval);
01946
01947 /**
01948 * Compute hyperbolic sine.
01949 */
01950 float const_func __attribute__((overloadable)) sinh(float);
01951 float2 const_func __attribute__((overloadable)) sinh(float2);
01952 float3 const_func __attribute__((overloadable)) sinh(float3);
01953 float4 const_func __attribute__((overloadable)) sinh(float4);
01954 float8 const_func __attribute__((overloadable)) sinh(float8);
01955 float16 const_func __attribute__((overloadable)) sinh(float16);
01956 double const_func __attribute__((overloadable)) sinh(double);
01957 double2 const_func __attribute__((overloadable)) sinh(double2);
01958 double3 const_func __attribute__((overloadable)) sinh(double3);
01959 double4 const_func __attribute__((overloadable)) sinh(double4);
01960 double8 const_func __attribute__((overloadable)) sinh(double8);
01961 double16 const_func __attribute__((overloadable)) sinh(double16);
01962 //half const_func __attribute__((overloadable)) sinh(half);
01963 //half2 const_func __attribute__((overloadable)) sinh(half2);
01964 //half3 const_func __attribute__((overloadable)) sinh(half3);
01965 //half4 const_func __attribute__((overloadable)) sinh(half4);
01966 //half8 const_func __attribute__((overloadable)) sinh(half8);
```

```
01967 //half16 const_func __attribute__((overloadable)) sinh(half16);
01968
01969 /**
01970 * Compute sin (PI * x).
01971 */
01972 float const_func __attribute__((overloadable)) sinpi(float x);
01973 float2 const_func __attribute__((overloadable)) sinpi(float2 x);
01974 float3 const_func __attribute__((overloadable)) sinpi(float3 x);
01975 float4 const_func __attribute__((overloadable)) sinpi(float4 x);
01976 float8 const_func __attribute__((overloadable)) sinpi(float8 x);
01977 float16 const_func __attribute__((overloadable)) sinpi(float16 x);
01978 double const_func __attribute__((overloadable)) sinpi(double x);
01979 double2 const_func __attribute__((overloadable)) sinpi(double2 x);
01980 double3 const_func __attribute__((overloadable)) sinpi(double3 x);
01981 double4 const_func __attribute__((overloadable)) sinpi(double4 x);
01982 double8 const_func __attribute__((overloadable)) sinpi(double8 x);
01983 double16 const_func __attribute__((overloadable)) sinpi(double16 x);
01984 //half const_func __attribute__((overloadable)) sinpi(half x);
01985 //half2 const_func __attribute__((overloadable)) sinpi(half2 x);
01986 //half3 const_func __attribute__((overloadable)) sinpi(half3 x);
01987 //half4 const_func __attribute__((overloadable)) sinpi(half4 x);
01988 //half8 const_func __attribute__((overloadable)) sinpi(half8 x);
01989 //half16 const_func __attribute__((overloadable)) sinpi(half16 x);
01990
01991 /**
01992 * Compute square root.
01993 */
01994 double const_func __attribute__((overloadable)) sqrt(double);
01995 double2 const_func __attribute__((overloadable)) sqrt(double2);
01996 double3 const_func __attribute__((overloadable)) sqrt(double3);
01997 double4 const_func __attribute__((overloadable)) sqrt(double4);
01998 double8 const_func __attribute__((overloadable)) sqrt(double8);
01999 double16 const_func __attribute__((overloadable)) sqrt(double16);
02000 float const_func __attribute__((overloadable)) sqrt(float);
02001 float2 const_func __attribute__((overloadable)) sqrt(float2);
02002 float3 const_func __attribute__((overloadable)) sqrt(float3);
02003 float4 const_func __attribute__((overloadable)) sqrt(float4);
02004 float8 const_func __attribute__((overloadable)) sqrt(float8);
02005 float16 const_func __attribute__((overloadable)) sqrt(float16);
02006 //half const_func __attribute__((overloadable)) sqrt(half);
02007 //half2 const_func __attribute__((overloadable)) sqrt(half2);
02008 //half3 const_func __attribute__((overloadable)) sqrt(half3);
02009 //half4 const_func __attribute__((overloadable)) sqrt(half4);
02010 //half8 const_func __attribute__((overloadable)) sqrt(half8);
02011 //half16 const_func __attribute__((overloadable)) sqrt(half16);
02012
02013 /**
02014 * Compute tangent.
02015 */
02016 float const_func __attribute__((overloadable)) tan(float);
02017 float2 const_func __attribute__((overloadable)) tan(float2);
02018 float3 const_func __attribute__((overloadable)) tan(float3);
02019 float4 const_func __attribute__((overloadable)) tan(float4);
02020 float8 const_func __attribute__((overloadable)) tan(float8);
02021 float16 const_func __attribute__((overloadable)) tan(float16);
02022 double const_func __attribute__((overloadable)) tan(double);
02023 double2 const_func __attribute__((overloadable)) tan(double2);
02024 double3 const_func __attribute__((overloadable)) tan(double3);
02025 double4 const_func __attribute__((overloadable)) tan(double4);
02026 double8 const_func __attribute__((overloadable)) tan(double8);
02027 double16 const_func __attribute__((overloadable)) tan(double16);
02028 //half const_func __attribute__((overloadable)) tan(half);
02029 //half2 const_func __attribute__((overloadable)) tan(half2);
02030 //half3 const_func __attribute__((overloadable)) tan(half3);
02031 //half4 const_func __attribute__((overloadable)) tan(half4);
02032 //half8 const_func __attribute__((overloadable)) tan(half8);
02033 //half16 const_func __attribute__((overloadable)) tan(half16);
02034
02035 /**
02036 * Compute hyperbolic tangent.
02037 */
02038 float const_func __attribute__((overloadable)) tanh(float);
02039 float2 const_func __attribute__((overloadable)) tanh(float2);
02040 float3 const_func __attribute__((overloadable)) tanh(float3);
02041 float4 const_func __attribute__((overloadable)) tanh(float4);
02042 float8 const_func __attribute__((overloadable)) tanh(float8);
02043 float16 const_func __attribute__((overloadable)) tanh(float16);
02044 double const_func __attribute__((overloadable)) tanh(double);
02045 double2 const_func __attribute__((overloadable)) tanh(double2);
02046 double3 const_func __attribute__((overloadable)) tanh(double3);
02047 double4 const_func __attribute__((overloadable)) tanh(double4);
02048 double8 const_func __attribute__((overloadable)) tanh(double8);
02049 double16 const_func __attribute__((overloadable)) tanh(double16);
02050 //half const_func __attribute__((overloadable)) tanh(half);
02051 //half2 const_func __attribute__((overloadable)) tanh(half2);
02052 //half3 const_func __attribute__((overloadable)) tanh(half3);
02053 //half4 const_func __attribute__((overloadable)) tanh(half4);
```

```
02054 //half8 const_func __attribute__((overloadable)) tanh(half8);
02055 //half16 const_func __attribute__((overloadable)) tanh(half16);
02056
02057 /**
02058 * Compute tan (PI * x).
02059 */
02060 float const_func __attribute__((overloadable)) tanpi(float x);
02061 float2 const_func __attribute__((overloadable)) tanpi(float2 x);
02062 float3 const_func __attribute__((overloadable)) tanpi(float3 x);
02063 float4 const_func __attribute__((overloadable)) tanpi(float4 x);
02064 float8 const_func __attribute__((overloadable)) tanpi(float8 x);
02065 float16 const_func __attribute__((overloadable)) tanpi(float16 x);
02066 double const_func __attribute__((overloadable)) tanpi(double x);
02067 double2 const_func __attribute__((overloadable)) tanpi(double2 x);
02068 double3 const_func __attribute__((overloadable)) tanpi(double3 x);
02069 double4 const_func __attribute__((overloadable)) tanpi(double4 x);
02070 double8 const_func __attribute__((overloadable)) tanpi(double8 x);
02071 double16 const_func __attribute__((overloadable)) tanpi(double16 x);
02072 //half const_func __attribute__((overloadable)) tanpi(half x);
02073 //half2 const_func __attribute__((overloadable)) tanpi(half2 x);
02074 //half3 const_func __attribute__((overloadable)) tanpi(half3 x);
02075 //half4 const_func __attribute__((overloadable)) tanpi(half4 x);
02076 //half8 const_func __attribute__((overloadable)) tanpi(half8 x);
02077 //half16 const_func __attribute__((overloadable)) tanpi(half16 x);
02078
02079 /**
02080 * Compute the gamma function.
02081 */
02082 float const_func __attribute__((overloadable)) tgamma(float);
02083 float2 const_func __attribute__((overloadable)) tgamma(float2);
02084 float3 const_func __attribute__((overloadable)) tgamma(float3);
02085 float4 const_func __attribute__((overloadable)) tgamma(float4);
02086 float8 const_func __attribute__((overloadable)) tgamma(float8);
02087 float16 const_func __attribute__((overloadable)) tgamma(float16);
02088 double const_func __attribute__((overloadable)) tgamma(double);
02089 double2 const_func __attribute__((overloadable)) tgamma(double2);
02090 double3 const_func __attribute__((overloadable)) tgamma(double3);
02091 double4 const_func __attribute__((overloadable)) tgamma(double4);
02092 double8 const_func __attribute__((overloadable)) tgamma(double8);
02093 double16 const_func __attribute__((overloadable)) tgamma(double16);
02094 //half const_func __attribute__((overloadable)) tgamma(half);
02095 //half2 const_func __attribute__((overloadable)) tgamma(half2);
02096 //half3 const_func __attribute__((overloadable)) tgamma(half3);
02097 //half4 const_func __attribute__((overloadable)) tgamma(half4);
02098 //half8 const_func __attribute__((overloadable)) tgamma(half8);
02099 //half16 const_func __attribute__((overloadable)) tgamma(half16);
02100
02101 /**
02102 * Round to integral value using the round to zero
02103 * rounding mode.
02104 */
02105 float const_func __attribute__((overloadable)) trunc(float);
02106 float2 const_func __attribute__((overloadable)) trunc(float2);
02107 float3 const_func __attribute__((overloadable)) trunc(float3);
02108 float4 const_func __attribute__((overloadable)) trunc(float4);
02109 float8 const_func __attribute__((overloadable)) trunc(float8);
02110 float16 const_func __attribute__((overloadable)) trunc(float16);
02111 double const_func __attribute__((overloadable)) trunc(double);
02112 double2 const_func __attribute__((overloadable)) trunc(double2);
02113 double3 const_func __attribute__((overloadable)) trunc(double3);
02114 double4 const_func __attribute__((overloadable)) trunc(double4);
02115 double8 const_func __attribute__((overloadable)) trunc(double8);
02116 double16 const_func __attribute__((overloadable)) trunc(double16);
02117 //half const_func __attribute__((overloadable)) trunc(half);
02118 //half2 const_func __attribute__((overloadable)) trunc(half2);
02119 //half3 const_func __attribute__((overloadable)) trunc(half3);
02120 //half4 const_func __attribute__((overloadable)) trunc(half4);
02121 //half8 const_func __attribute__((overloadable)) trunc(half8);
02122 //half16 const_func __attribute__((overloadable)) trunc(half16);
02123
02124 /**
02125 * Compute cosine.  x must be in the range -2^16 ...  +2^16.
02126 */
02127 float const_func __attribute__((overloadable)) half_cos(float x);
02128 float2 const_func __attribute__((overloadable)) half_cos(float2 x);
02129 float3 const_func __attribute__((overloadable)) half_cos(float3 x);
02130 float4 const_func __attribute__((overloadable)) half_cos(float4 x);
02131 float8 const_func __attribute__((overloadable)) half_cos(float8 x);
02132 float16 const_func __attribute__((overloadable)) half_cos(float16 x);
02133
02134 /**
02135 * Compute x / y.
02136 */
02137 float const_func __attribute__((overloadable)) half_divide(float x, float y);
02138 float2 const_func __attribute__((overloadable)) half_divide(float2 x, float2 y);
02139 float3 const_func __attribute__((overloadable)) half_divide(float3 x, float3 y);
02140 float4 const_func __attribute__((overloadable)) half_divide(float4 x, float4 y);
```

```
02141 float8 const_func __attribute__((overloadable)) half_divide(float8 x, float8 y);
02142 float16 const_func __attribute__((overloadable)) half_divide(float16 x, float16 y);
02143
02144 /**
02145 * Compute the base- e exponential of x.
02146 */
02147 float const_func __attribute__((overloadable)) half_exp(float x);
02148 float2 const_func __attribute__((overloadable)) half_exp(float2 x);
02149 float3 const_func __attribute__((overloadable)) half_exp(float3 x);
02150 float4 const_func __attribute__((overloadable)) half_exp(float4 x);
02151 float8 const_func __attribute__((overloadable)) half_exp(float8 x);
02152 float16 const_func __attribute__((overloadable)) half_exp(float16 x);
02153
02154 /**
02155 * Compute the base- 2 exponential of x.
02156 */
02157 float const_func __attribute__((overloadable)) half_exp2(float x);
02158 float2 const_func __attribute__((overloadable)) half_exp2(float2 x);
02159 float3 const_func __attribute__((overloadable)) half_exp2(float3 x);
02160 float4 const_func __attribute__((overloadable)) half_exp2(float4 x);
02161 float8 const_func __attribute__((overloadable)) half_exp2(float8 x);
02162 float16 const_func __attribute__((overloadable)) half_exp2(float16 x);
02163
02164 /**
02165 * Compute the base- 10 exponential of x.
02166 */
02167 float const_func __attribute__((overloadable)) half_exp10(float x);
02168 float2 const_func __attribute__((overloadable)) half_exp10(float2 x);
02169 float3 const_func __attribute__((overloadable)) half_exp10(float3 x);
02170 float4 const_func __attribute__((overloadable)) half_exp10(float4 x);
02171 float8 const_func __attribute__((overloadable)) half_exp10(float8 x);
02172 float16 const_func __attribute__((overloadable)) half_exp10(float16 x);
02173
02174 /**
02175 * Compute natural logarithm.
02176 */
02177 float const_func __attribute__((overloadable)) half_log(float x);
02178 float2 const_func __attribute__((overloadable)) half_log(float2 x);
02179 float3 const_func __attribute__((overloadable)) half_log(float3 x);
02180 float4 const_func __attribute__((overloadable)) half_log(float4 x);
02181 float8 const_func __attribute__((overloadable)) half_log(float8 x);
02182 float16 const_func __attribute__((overloadable)) half_log(float16 x);
02183
02184 /**
02185 * Compute a base 2 logarithm.
02186 */
02187 float const_func __attribute__((overloadable)) half_log2(float x);
02188 float2 const_func __attribute__((overloadable)) half_log2(float2 x);
02189 float3 const_func __attribute__((overloadable)) half_log2(float3 x);
02190 float4 const_func __attribute__((overloadable)) half_log2(float4 x);
02191 float8 const_func __attribute__((overloadable)) half_log2(float8 x);
02192 float16 const_func __attribute__((overloadable)) half_log2(float16 x);
02193
02194 /**
02195 * Compute a base 10 logarithm.
02196 */
02197 float const_func __attribute__((overloadable)) half_log10(float x);
02198 float2 const_func __attribute__((overloadable)) half_log10(float2 x);
02199 float3 const_func __attribute__((overloadable)) half_log10(float3 x);
02200 float4 const_func __attribute__((overloadable)) half_log10(float4 x);
02201 float8 const_func __attribute__((overloadable)) half_log10(float8 x);
02202 float16 const_func __attribute__((overloadable)) half_log10(float16 x);
02203
02204 /**
02205 * Compute x to the power y, where x is >= 0.
02206 */
02207 float const_func __attribute__((overloadable)) half_powr(float x, float y);
02208 float2 const_func __attribute__((overloadable)) half_powr(float2 x, float2 y);
02209 float3 const_func __attribute__((overloadable)) half_powr(float3 x, float3 y);
02210 float4 const_func __attribute__((overloadable)) half_powr(float4 x, float4 y);
02211 float8 const_func __attribute__((overloadable)) half_powr(float8 x, float8 y);
02212 float16 const_func __attribute__((overloadable)) half_powr(float16 x, float16 y);
02213
02214 /**
02215 * Compute reciprocal.
02216 */
02217 float const_func __attribute__((overloadable)) half_recip(float x);
02218 float2 const_func __attribute__((overloadable)) half_recip(float2 x);
02219 float3 const_func __attribute__((overloadable)) half_recip(float3 x);
02220 float4 const_func __attribute__((overloadable)) half_recip(float4 x);
02221 float8 const_func __attribute__((overloadable)) half_recip(float8 x);
02222 float16 const_func __attribute__((overloadable)) half_recip(float16 x);
02223
02224 /**
02225 * Compute inverse square root.
02226 */
02227 float const_func __attribute__((overloadable)) half_rsqrt(float x);
```

```
02228 float2 const_func __attribute__((overloadable)) half_rsqrt(float2 x);
02229 float3 const_func __attribute__((overloadable)) half_rsqrt(float3 x);
02230 float4 const_func __attribute__((overloadable)) half_rsqrt(float4 x);
02231 float8 const_func __attribute__((overloadable)) half_rsqrt(float8 x);
02232 float16 const_func __attribute__((overloadable)) half_rsqrt(float16 x);
02233
02234 /**
02235 * Compute sine.  x must be in the range -2^16 ...  +2^16.
02236 */
02237 float const_func __attribute__((overloadable)) half_sin(float x);
02238 float2 const_func __attribute__((overloadable)) half_sin(float2 x);
02239 float3 const_func __attribute__((overloadable)) half_sin(float3 x);
02240 float4 const_func __attribute__((overloadable)) half_sin(float4 x);
02241 float8 const_func __attribute__((overloadable)) half_sin(float8 x);
02242 float16 const_func __attribute__((overloadable)) half_sin(float16 x);
02243
02244 /**
02245 * Compute square root.
02246 */
02247 float const_func __attribute__((overloadable)) half_sqrt(float x);
02248 float2 const_func __attribute__((overloadable)) half_sqrt(float2 x);
02249 float3 const_func __attribute__((overloadable)) half_sqrt(float3 x);
02250 float4 const_func __attribute__((overloadable)) half_sqrt(float4 x);
02251 float8 const_func __attribute__((overloadable)) half_sqrt(float8 x);
02252 float16 const_func __attribute__((overloadable)) half_sqrt(float16 x);
02253
02254 /**
02255 * Compute tangent.  x must be in the range -216 ...  +216.
02256 */
02257 float const_func __attribute__((overloadable)) half_tan(float x);
02258 float2 const_func __attribute__((overloadable)) half_tan(float2 x);
02259 float3 const_func __attribute__((overloadable)) half_tan(float3 x);
02260 float4 const_func __attribute__((overloadable)) half_tan(float4 x);
02261 float8 const_func __attribute__((overloadable)) half_tan(float8 x);
02262 float16 const_func __attribute__((overloadable)) half_tan(float16 x);
02263
02264 /**
02265 * Compute cosine over an implementation-defined range.
02266 * The maximum error is implementation-defined.
02267 */
02268 float const_func __attribute__((overloadable)) native_cos(float x);
02269 float2 const_func __attribute__((overloadable)) native_cos(float2 x);
02270 float3 const_func __attribute__((overloadable)) native_cos(float3 x);
02271 float4 const_func __attribute__((overloadable)) native_cos(float4 x);
02272 float8 const_func __attribute__((overloadable)) native_cos(float8 x);
02273 float16 const_func __attribute__((overloadable)) native_cos(float16 x);
02274
02275 // EXTENSION: native double
02276 double const_func __attribute__((overloadable)) native_cos(double x);
02277 double2 const_func __attribute__((overloadable)) native_cos(double2 x);
02278 double3 const_func __attribute__((overloadable)) native_cos(double3 x);
02279 double4 const_func __attribute__((overloadable)) native_cos(double4 x);
02280 double8 const_func __attribute__((overloadable)) native_cos(double8 x);
02281 double16 const_func __attribute__((overloadable)) native_cos(double16 x);
02282
02283 /**
02284 * Compute x / y over an implementation-defined range.
02285 * The maximum error is implementation-defined.
02286 */
02287 float const_func __attribute__((overloadable)) native_divide(float x, float y);
02288 float2 const_func __attribute__((overloadable)) native_divide(float2 x, float2 y);
02289 float3 const_func __attribute__((overloadable)) native_divide(float3 x, float3 y);
02290 float4 const_func __attribute__((overloadable)) native_divide(float4 x, float4 y);
02291 float8 const_func __attribute__((overloadable)) native_divide(float8 x, float8 y);
02292 float16 const_func __attribute__((overloadable)) native_divide(float16 x, float16 y);
02293
02294 // EXTENSION: native double
02295 double const_func __attribute__((overloadable)) native_divide(double x, double y);
02296 double2 const_func __attribute__((overloadable)) native_divide(double2 x, double2 y);
02297 double3 const_func __attribute__((overloadable)) native_divide(double3 x, double3 y);
02298 double4 const_func __attribute__((overloadable)) native_divide(double4 x, double4 y);
02299 double8 const_func __attribute__((overloadable)) native_divide(double8 x, double8 y);
02300 double16 const_func __attribute__((overloadable)) native_divide(double16 x, double16 y);
02301
02302 /**
02303 * Compute the base- e exponential of x over an
02304 * implementation-defined range.  The maximum error is
02305 * implementation-defined.
02306 */
02307 float const_func __attribute__((overloadable)) native_exp(float x);
02308 float2 const_func __attribute__((overloadable)) native_exp(float2 x);
02309 float3 const_func __attribute__((overloadable)) native_exp(float3 x);
02310 float4 const_func __attribute__((overloadable)) native_exp(float4 x);
02311 float8 const_func __attribute__((overloadable)) native_exp(float8 x);
02312 float16 const_func __attribute__((overloadable)) native_exp(float16 x);
02313
02314 /**
```

```
02315 * Compute the base- 2 exponential of x over an
02316 * implementation-defined range.  The maximum error is
02317 * implementation-defined.
02318 */
02319 float const_func __attribute__((overloadable)) native_exp2(float x);
02320 float2 const_func __attribute__((overloadable)) native_exp2(float2 x);
02321 float3 const_func __attribute__((overloadable)) native_exp2(float3 x);
02322 float4 const_func __attribute__((overloadable)) native_exp2(float4 x);
02323 float8 const_func __attribute__((overloadable)) native_exp2(float8 x);
02324 float16 const_func __attribute__((overloadable)) native_exp2(float16 x);
02325
02326 /**
02327 * Compute the base- 10 exponential of x over an
02328 * implementation-defined range.  The maximum error is
02329 * implementation-defined.
02330 */
02331 float const_func __attribute__((overloadable)) native_exp10(float x);
02332 float2 const_func __attribute__((overloadable)) native_exp10(float2 x);
02333 float3 const_func __attribute__((overloadable)) native_exp10(float3 x);
02334 float4 const_func __attribute__((overloadable)) native_exp10(float4 x);
02335 float8 const_func __attribute__((overloadable)) native_exp10(float8 x);
02336 float16 const_func __attribute__((overloadable)) native_exp10(float16 x);
02337
02338 /**
02339 * Compute natural logarithm over an implementationdefined
02340 * range.  The maximum error is implementation
02341 * defined.
02342 */
02343 float const_func __attribute__((overloadable)) native_log(float x);
02344 float2 const_func __attribute__((overloadable)) native_log(float2 x);
02345 float3 const_func __attribute__((overloadable)) native_log(float3 x);
02346 float4 const_func __attribute__((overloadable)) native_log(float4 x);
02347 float8 const_func __attribute__((overloadable)) native_log(float8 x);
02348 float16 const_func __attribute__((overloadable)) native_log(float16 x);
02349
02350 /**
02351 * Compute a base 2 logarithm over an implementationdefined
02352 * range.  The maximum error is implementationdefined.
02353 */
02354 float const_func __attribute__((overloadable)) native_log2(float x);
02355 float2 const_func __attribute__((overloadable)) native_log2(float2 x);
02356 float3 const_func __attribute__((overloadable)) native_log2(float3 x);
02357 float4 const_func __attribute__((overloadable)) native_log2(float4 x);
02358 float8 const_func __attribute__((overloadable)) native_log2(float8 x);
02359 float16 const_func __attribute__((overloadable)) native_log2(float16 x);
02360
02361 /**
02362 * Compute a base 10 logarithm over an implementationdefined
02363 * range.  The maximum error is implementationdefined.
02364 */
02365 float const_func __attribute__((overloadable)) native_log10(float x);
02366 float2 const_func __attribute__((overloadable)) native_log10(float2 x);
02367 float3 const_func __attribute__((overloadable)) native_log10(float3 x);
02368 float4 const_func __attribute__((overloadable)) native_log10(float4 x);
02369 float8 const_func __attribute__((overloadable)) native_log10(float8 x);
02370 float16 const_func __attribute__((overloadable)) native_log10(float16 x);
02371
02372 /**
02373 * Compute x to the power y, where x is >= 0.  The range of
02374 * x and y are implementation-defined.  The maximum error
02375 * is implementation-defined.
02376 */
02377 float const_func __attribute__((overloadable)) native_powr(float x, float y);
02378 float2 const_func __attribute__((overloadable)) native_powr(float2 x, float2 y);
02379 float3 const_func __attribute__((overloadable)) native_powr(float3 x, float3 y);
02380 float4 const_func __attribute__((overloadable)) native_powr(float4 x, float4 y);
02381 float8 const_func __attribute__((overloadable)) native_powr(float8 x, float8 y);
02382 float16 const_func __attribute__((overloadable)) native_powr(float16 x, float16 y);
02383
02384 /**
02385 * Compute reciprocal over an implementation-defined
02386 * range.  The maximum error is implementation-defined.
02387 */
02388 float const_func __attribute__((overloadable)) native_recip(float x);
02389 float2 const_func __attribute__((overloadable)) native_recip(float2 x);
02390 float3 const_func __attribute__((overloadable)) native_recip(float3 x);
02391 float4 const_func __attribute__((overloadable)) native_recip(float4 x);
02392 float8 const_func __attribute__((overloadable)) native_recip(float8 x);
02393 float16 const_func __attribute__((overloadable)) native_recip(float16 x);
02394
02395 /**
02396 * Compute inverse square root over an implementationdefined
02397 * range.  The maximum error is implementationdefined.
02398 */
02399 float const_func __attribute__((overloadable)) native_rsqrt(float x);
02400 float2 const_func __attribute__((overloadable)) native_rsqrt(float2 x);
02401 float3 const_func __attribute__((overloadable)) native_rsqrt(float3 x);
```

```
02402 float4 const_func __attribute__((overloadable)) native_rsqrt(float4 x);
02403 float8 const_func __attribute__((overloadable)) native_rsqrt(float8 x);
02404 float16 const_func __attribute__((overloadable)) native_rsqrt(float16 x);
02405
02406 /**
02407 * Compute sine over an implementation-defined range.
02408 * The maximum error is implementation-defined.
02409 */
02410 float const_func __attribute__((overloadable)) native_sin(float x);
02411 float2 const_func __attribute__((overloadable)) native_sin(float2 x);
02412 float3 const_func __attribute__((overloadable)) native_sin(float3 x);
02413 float4 const_func __attribute__((overloadable)) native_sin(float4 x);
02414 float8 const_func __attribute__((overloadable)) native_sin(float8 x);
02415 float16 const_func __attribute__((overloadable)) native_sin(float16 x);
02416
02417 /**
02418 * Compute square root over an implementation-defined
02419 * range.  The maximum error is implementation-defined.
02420 */
02421 float const_func __attribute__((overloadable)) native_sqrt(float x);
02422 float2 const_func __attribute__((overloadable)) native_sqrt(float2 x);
02423 float3 const_func __attribute__((overloadable)) native_sqrt(float3 x);
02424 float4 const_func __attribute__((overloadable)) native_sqrt(float4 x);
02425 float8 const_func __attribute__((overloadable)) native_sqrt(float8 x);
02426 float16 const_func __attribute__((overloadable)) native_sqrt(float16 x);
02427
02428 /**
02429 * Compute tangent over an implementation-defined range.
02430 * The maximum error is implementation-defined.
02431 */
02432 float const_func __attribute__((overloadable)) native_tan(float x);
02433 float2 const_func __attribute__((overloadable)) native_tan(float2 x);
02434 float3 const_func __attribute__((overloadable)) native_tan(float3 x);
02435 float4 const_func __attribute__((overloadable)) native_tan(float4 x);
02436 float8 const_func __attribute__((overloadable)) native_tan(float8 x);
02437 float16 const_func __attribute__((overloadable)) native_tan(float16 x);
02438
02439 // Integer functions:
02440
02441 /**
02442 * Returns | x |.
02443 */
02444 uchar const_func __attribute__((overloadable)) abs(char x);
02445 uchar const_func __attribute__((overloadable)) abs(uchar x);
02446 uchar2 const_func __attribute__((overloadable)) abs(char2 x);
02447 uchar2 const_func __attribute__((overloadable)) abs(uchar2 x);
02448 uchar3 const_func __attribute__((overloadable)) abs(char3 x);
02449 uchar3 const_func __attribute__((overloadable)) abs(uchar3 x);
02450 uchar4 const_func __attribute__((overloadable)) abs(char4 x);
02451 uchar4 const_func __attribute__((overloadable)) abs(uchar4 x);
02452 uchar8 const_func __attribute__((overloadable)) abs(char8 x);
02453 uchar8 const_func __attribute__((overloadable)) abs(uchar8 x);
02454 uchar16 const_func __attribute__((overloadable)) abs(char16 x);
02455 uchar16 const_func __attribute__((overloadable)) abs(uchar16 x);
02456 ushort const_func __attribute__((overloadable)) abs(short x);
02457 ushort const_func __attribute__((overloadable)) abs(ushort x);
02458 ushort2 const_func __attribute__((overloadable)) abs(short2 x);
02459 ushort2 const_func __attribute__((overloadable)) abs(ushort2 x);
02460 ushort3 const_func __attribute__((overloadable)) abs(short3 x);
02461 ushort3 const_func __attribute__((overloadable)) abs(ushort3 x);
02462 ushort4 const_func __attribute__((overloadable)) abs(short4 x);
02463 ushort4 const_func __attribute__((overloadable)) abs(ushort4 x);
02464 ushort8 const_func __attribute__((overloadable)) abs(short8 x);
02465 ushort8 const_func __attribute__((overloadable)) abs(ushort8 x);
02466 ushort16 const_func __attribute__((overloadable)) abs(short16 x);
02467 ushort16 const_func __attribute__((overloadable)) abs(ushort16 x);
02468 uint const_func __attribute__((overloadable)) abs(int x);
02469 uint const_func __attribute__((overloadable)) abs(uint x);
02470 uint2 const_func __attribute__((overloadable)) abs(int2 x);
02471 uint2 const_func __attribute__((overloadable)) abs(uint2 x);
02472 uint3 const_func __attribute__((overloadable)) abs(int3 x);
02473 uint3 const_func __attribute__((overloadable)) abs(uint3 x);
02474 uint4 const_func __attribute__((overloadable)) abs(int4 x);
02475 uint4 const_func __attribute__((overloadable)) abs(uint4 x);
02476 uint8 const_func __attribute__((overloadable)) abs(int8 x);
02477 uint8 const_func __attribute__((overloadable)) abs(uint8 x);
02478 uint16 const_func __attribute__((overloadable)) abs(int16 x);
02479 uint16 const_func __attribute__((overloadable)) abs(uint16 x);
02480 ulong const_func __attribute__((overloadable)) abs(long x);
02481 ulong const_func __attribute__((overloadable)) abs(ulong x);
02482 ulong2 const_func __attribute__((overloadable)) abs(long2 x);
02483 ulong2 const_func __attribute__((overloadable)) abs(ulong2 x);
02484 ulong3 const_func __attribute__((overloadable)) abs(long3 x);
02485 ulong3 const_func __attribute__((overloadable)) abs(ulong3 x);
02486 ulong4 const_func __attribute__((overloadable)) abs(long4 x);
02487 ulong4 const_func __attribute__((overloadable)) abs(ulong4 x);
02488 ulong8 const_func __attribute__((overloadable)) abs(long8 x);
```

```
02489 ulong8 const_func __attribute__((overloadable)) abs(ulong8 x);
02490 ulong16 const_func __attribute__((overloadable)) abs(long16 x);
02491 ulong16 const_func __attribute__((overloadable)) abs(ulong16 x);
02492
02493 /**
02494 * Returns | x - y | without modulo overflow.
02495 */
02496 uchar const_func __attribute__((overloadable)) abs_diff(char x, char y);
02497 uchar const_func __attribute__((overloadable)) abs_diff(uchar x, uchar y);
02498 uchar2 const_func __attribute__((overloadable)) abs_diff(char2 x, char2 y);
02499 uchar2 const_func __attribute__((overloadable)) abs_diff(uchar2 x, uchar2 y);
02500 uchar3 const_func __attribute__((overloadable)) abs_diff(char3 x, char3 y);
02501 uchar3 const_func __attribute__((overloadable)) abs_diff(uchar3 x, uchar3 y);
02502 uchar4 const_func __attribute__((overloadable)) abs_diff(char4 x, char4 y);
02503 uchar4 const_func __attribute__((overloadable)) abs_diff(uchar4 x, uchar4 y);
02504 uchar8 const_func __attribute__((overloadable)) abs_diff(char8 x, char8 y);
02505 uchar8 const_func __attribute__((overloadable)) abs_diff(uchar8 x, uchar8 y);
02506 uchar16 const_func __attribute__((overloadable)) abs_diff(char16 x, char16 y);
02507 uchar16 const_func __attribute__((overloadable)) abs_diff(uchar16 x, uchar16 y);
02508 ushort const_func __attribute__((overloadable)) abs_diff(short x, short y);
02509 ushort const_func __attribute__((overloadable)) abs_diff(ushort x, ushort y);
02510 ushort2 const_func __attribute__((overloadable)) abs_diff(short2 x, short2 y);
02511 ushort2 const_func __attribute__((overloadable)) abs_diff(ushort2 x, ushort2 y);
02512 ushort3 const_func __attribute__((overloadable)) abs_diff(short3 x, short3 y);
02513 ushort3 const_func __attribute__((overloadable)) abs_diff(ushort3 x, ushort3 y);
02514 ushort4 const_func __attribute__((overloadable)) abs_diff(short4 x, short4 y);
02515 ushort4 const_func __attribute__((overloadable)) abs_diff(ushort4 x, ushort4 y);
02516 ushort8 const_func __attribute__((overloadable)) abs_diff(short8 x, short8 y);
02517 ushort8 const_func __attribute__((overloadable)) abs_diff(ushort8 x, ushort8 y);
02518 ushort16 const_func __attribute__((overloadable)) abs_diff(short16 x, short16 y);
02519 ushort16 const_func __attribute__((overloadable)) abs_diff(ushort16 x, ushort16 y);
02520 uint const_func __attribute__((overloadable)) abs_diff(int x, int y);
02521 uint const_func __attribute__((overloadable)) abs_diff(uint x, uint y);
02522 uint2 const_func __attribute__((overloadable)) abs_diff(int2 x, int2 y);
02523 uint2 const_func __attribute__((overloadable)) abs_diff(uint2 x, uint2 y);
02524 uint3 const_func __attribute__((overloadable)) abs_diff(int3 x, int3 y);
02525 uint3 const_func __attribute__((overloadable)) abs_diff(uint3 x, uint3 y);
02526 uint4 const_func __attribute__((overloadable)) abs_diff(int4 x, int4 y);
02527 uint4 const_func __attribute__((overloadable)) abs_diff(uint4 x, uint4 y);
02528 uint8 const_func __attribute__((overloadable)) abs_diff(int8 x, int8 y);
02529 uint8 const_func __attribute__((overloadable)) abs_diff(uint8 x, uint8 y);
02530 uint16 const_func __attribute__((overloadable)) abs_diff(int16 x, int16 y);
02531 uint16 const_func __attribute__((overloadable)) abs_diff(uint16 x, uint16 y);
02532 ulong const_func __attribute__((overloadable)) abs_diff(long x, long y);
02533 ulong const_func __attribute__((overloadable)) abs_diff(ulong x, ulong y);
02534 ulong2 const_func __attribute__((overloadable)) abs_diff(long2 x, long2 y);
02535 ulong2 const_func __attribute__((overloadable)) abs_diff(ulong2 x, ulong2 y);
02536 ulong3 const_func __attribute__((overloadable)) abs_diff(long3 x, long3 y);
02537 ulong3 const_func __attribute__((overloadable)) abs_diff(ulong3 x, ulong3 y);
02538 ulong4 const_func __attribute__((overloadable)) abs_diff(long4 x, long4 y);
02539 ulong4 const_func __attribute__((overloadable)) abs_diff(ulong4 x, ulong4 y);
02540 ulong8 const_func __attribute__((overloadable)) abs_diff(long8 x, long8 y);
02541 ulong8 const_func __attribute__((overloadable)) abs_diff(ulong8 x, ulong8 y);
02542 ulong16 const_func __attribute__((overloadable)) abs_diff(long16 x, long16 y);
02543 ulong16 const_func __attribute__((overloadable)) abs_diff(ulong16 x, ulong16 y);
02544
02545 /**
02546 * Returns x + y and saturates the result.
02547 */
02548 char const_func __attribute__((overloadable)) add_sat(char x, char y);
02549 uchar const_func __attribute__((overloadable)) add_sat(uchar x, uchar y);
02550 char2 const_func __attribute__((overloadable)) add_sat(char2 x, char2 y);
02551 uchar2 const_func __attribute__((overloadable)) add_sat(uchar2 x, uchar2 y);
02552 char3 const_func __attribute__((overloadable)) add_sat(char3 x, char3 y);
02553 uchar3 const_func __attribute__((overloadable)) add_sat(uchar3 x, uchar3 y);
02554 char4 const_func __attribute__((overloadable)) add_sat(char4 x, char4 y);
02555 uchar4 const_func __attribute__((overloadable)) add_sat(uchar4 x, uchar4 y);
02556 char8 const_func __attribute__((overloadable)) add_sat(char8 x, char8 y);
02557 uchar8 const_func __attribute__((overloadable)) add_sat(uchar8 x, uchar8 y);
02558 char16 const_func __attribute__((overloadable)) add_sat(char16 x, char16 y);
02559 uchar16 const_func __attribute__((overloadable)) add_sat(uchar16 x, uchar16 y);
02560 short const_func __attribute__((overloadable)) add_sat(short x, short y);
02561 ushort const_func __attribute__((overloadable)) add_sat(ushort x, ushort y);
02562 short2 const_func __attribute__((overloadable)) add_sat(short2 x, short2 y);
02563 ushort2 const_func __attribute__((overloadable)) add_sat(ushort2 x, ushort2 y);
02564 short3 const_func __attribute__((overloadable)) add_sat(short3 x, short3 y);
02565 ushort3 const_func __attribute__((overloadable)) add_sat(ushort3 x, ushort3 y);
02566 short4 const_func __attribute__((overloadable)) add_sat(short4 x, short4 y);
02567 ushort4 const_func __attribute__((overloadable)) add_sat(ushort4 x, ushort4 y);
02568 short8 const_func __attribute__((overloadable)) add_sat(short8 x, short8 y);
02569 ushort8 const_func __attribute__((overloadable)) add_sat(ushort8 x, ushort8 y);
02570 short16 const_func __attribute__((overloadable)) add_sat(short16 x, short16 y);
02571 ushort16 const_func __attribute__((overloadable)) add_sat(ushort16 x, ushort16 y);
02572 int const_func __attribute__((overloadable)) add_sat(int x, int y);
02573 uint const_func __attribute__((overloadable)) add_sat(uint x, uint y);
02574 int2 const_func __attribute__((overloadable)) add_sat(int2 x, int2 y);
02575 uint2 const_func __attribute__((overloadable)) add_sat(uint2 x, uint2 y);
```

```
02576 int3 const_func __attribute__((overloadable)) add_sat(int3 x, int3 y);
02577 uint3 const_func __attribute__((overloadable)) add_sat(uint3 x, uint3 y);
02578 int4 const_func __attribute__((overloadable)) add_sat(int4 x, int4 y);
02579 uint4 const_func __attribute__((overloadable)) add_sat(uint4 x, uint4 y);
02580 int8 const_func __attribute__((overloadable)) add_sat(int8 x, int8 y);
02581 uint8 const_func __attribute__((overloadable)) add_sat(uint8 x, uint8 y);
02582 int16 const_func __attribute__((overloadable)) add_sat(int16 x, int16 y);
02583 uint16 const_func __attribute__((overloadable)) add_sat(uint16 x, uint16 y);
02584 long const_func __attribute__((overloadable)) add_sat(long x, long y);
02585 ulong const_func __attribute__((overloadable)) add_sat(ulong x, ulong y);
02586 long2 const_func __attribute__((overloadable)) add_sat(long2 x, long2 y);
02587 ulong2 const_func __attribute__((overloadable)) add_sat(ulong2 x, ulong2 y);
02588 long3 const_func __attribute__((overloadable)) add_sat(long3 x, long3 y);
02589 ulong3 const_func __attribute__((overloadable)) add_sat(ulong3 x, ulong3 y);
02590 long4 const_func __attribute__((overloadable)) add_sat(long4 x, long4 y);
02591 ulong4 const_func __attribute__((overloadable)) add_sat(ulong4 x, ulong4 y);
02592 long8 const_func __attribute__((overloadable)) add_sat(long8 x, long8 y);
02593 ulong8 const_func __attribute__((overloadable)) add_sat(ulong8 x, ulong8 y);
02594 long16 const_func __attribute__((overloadable)) add_sat(long16 x, long16 y);
02595 ulong16 const_func __attribute__((overloadable)) add_sat(ulong16 x, ulong16 y);
02596
02597 /**
02598 * Returns (x + y) » 1.  The intermediate sum does
02599 * not modulo overflow.
02600 */
02601 char const_func __attribute__((overloadable)) hadd(char x, char y);
02602 uchar const_func __attribute__((overloadable)) hadd(uchar x, uchar y);
02603 char2 const_func __attribute__((overloadable)) hadd(char2 x, char2 y);
02604 uchar2 const_func __attribute__((overloadable)) hadd(uchar2 x, uchar2 y);
02605 char3 const_func __attribute__((overloadable)) hadd(char3 x, char3 y);
02606 uchar3 const_func __attribute__((overloadable)) hadd(uchar3 x, uchar3 y);
02607 char4 const_func __attribute__((overloadable)) hadd(char4 x, char4 y);
02608 uchar4 const_func __attribute__((overloadable)) hadd(uchar4 x, uchar4 y);
02609 char8 const_func __attribute__((overloadable)) hadd(char8 x, char8 y);
02610 uchar8 const_func __attribute__((overloadable)) hadd(uchar8 x, uchar8 y);
02611 char16 const_func __attribute__((overloadable)) hadd(char16 x, char16 y);
02612 uchar16 const_func __attribute__((overloadable)) hadd(uchar16 x, uchar16 y);
02613 short const_func __attribute__((overloadable)) hadd(short x, short y);
02614 ushort const_func __attribute__((overloadable)) hadd(ushort x, ushort y);
02615 short2 const_func __attribute__((overloadable)) hadd(short2 x, short2 y);
02616 ushort2 const_func __attribute__((overloadable)) hadd(ushort2 x, ushort2 y);
02617 short3 const_func __attribute__((overloadable)) hadd(short3 x, short3 y);
02618 ushort3 const_func __attribute__((overloadable)) hadd(ushort3 x, ushort3 y);
02619 short4 const_func __attribute__((overloadable)) hadd(short4 x, short4 y);
02620 ushort4 const_func __attribute__((overloadable)) hadd(ushort4 x, ushort4 y);
02621 short8 const_func __attribute__((overloadable)) hadd(short8 x, short8 y);
02622 ushort8 const_func __attribute__((overloadable)) hadd(ushort8 x, ushort8 y);
02623 short16 const_func __attribute__((overloadable)) hadd(short16 x, short16 y);
02624 ushort16 const_func __attribute__((overloadable)) hadd(ushort16 x, ushort16 y);
02625 int const_func __attribute__((overloadable)) hadd(int x, int y);
02626 uint const_func __attribute__((overloadable)) hadd(uint x, uint y);
02627 int2 const_func __attribute__((overloadable)) hadd(int2 x, int2 y);
02628 uint2 const_func __attribute__((overloadable)) hadd(uint2 x, uint2 y);
02629 int3 const_func __attribute__((overloadable)) hadd(int3 x, int3 y);
02630 uint3 const_func __attribute__((overloadable)) hadd(uint3 x, uint3 y);
02631 int4 const_func __attribute__((overloadable)) hadd(int4 x, int4 y);
02632 uint4 const_func __attribute__((overloadable)) hadd(uint4 x, uint4 y);
02633 int8 const_func __attribute__((overloadable)) hadd(int8 x, int8 y);
02634 uint8 const_func __attribute__((overloadable)) hadd(uint8 x, uint8 y);
02635 int16 const_func __attribute__((overloadable)) hadd(int16 x, int16 y);
02636 uint16 const_func __attribute__((overloadable)) hadd(uint16 x, uint16 y);
02637 long const_func __attribute__((overloadable)) hadd(long x, long y);
02638 ulong const_func __attribute__((overloadable)) hadd(ulong x, ulong y);
02639 long2 const_func __attribute__((overloadable)) hadd(long2 x, long2 y);
02640 ulong2 const_func __attribute__((overloadable)) hadd(ulong2 x, ulong2 y);
02641 long3 const_func __attribute__((overloadable)) hadd(long3 x, long3 y);
02642 ulong3 const_func __attribute__((overloadable)) hadd(ulong3 x, ulong3 y);
02643 long4 const_func __attribute__((overloadable)) hadd(long4 x, long4 y);
02644 ulong4 const_func __attribute__((overloadable)) hadd(ulong4 x, ulong4 y);
02645 long8 const_func __attribute__((overloadable)) hadd(long8 x, long8 y);
02646 ulong8 const_func __attribute__((overloadable)) hadd(ulong8 x, ulong8 y);
02647 long16 const_func __attribute__((overloadable)) hadd(long16 x, long16 y);
02648 ulong16 const_func __attribute__((overloadable)) hadd(ulong16 x, ulong16 y);
02649
02650 /**
02651 * Returns (x + y + 1) » 1.  The intermediate sum
02652 * does not modulo overflow.
02653 */
02654 char const_func __attribute__((overloadable)) rhadd(char x, char y);
02655 uchar const_func __attribute__((overloadable)) rhadd(uchar x, uchar y);
02656 char2 const_func __attribute__((overloadable)) rhadd(char2 x, char2 y);
02657 uchar2 const_func __attribute__((overloadable)) rhadd(uchar2 x, uchar2 y);
02658 char3 const_func __attribute__((overloadable)) rhadd(char3 x, char3 y);
02659 uchar3 const_func __attribute__((overloadable)) rhadd(uchar3 x, uchar3 y);
02660 char4 const_func __attribute__((overloadable)) rhadd(char4 x, char4 y);
02661 uchar4 const_func __attribute__((overloadable)) rhadd(uchar4 x, uchar4 y);
02662 char8 const_func __attribute__((overloadable)) rhadd(char8 x, char8 y);
```

```
02663 uchar8 const_func __attribute__((overloadable)) rhadd(uchar8 x, uchar8 y);
02664 char16 const_func __attribute__((overloadable)) rhadd(char16 x, char16 y);
02665 uchar16 const_func __attribute__((overloadable)) rhadd(uchar16 x, uchar16 y);
02666 short const_func __attribute__((overloadable)) rhadd(short x, short y);
02667 ushort const_func __attribute__((overloadable)) rhadd(ushort x, ushort y);
02668 short2 const_func __attribute__((overloadable)) rhadd(short2 x, short2 y);
02669 ushort2 const_func __attribute__((overloadable)) rhadd(ushort2 x, ushort2 y);
02670 short3 const_func __attribute__((overloadable)) rhadd(short3 x, short3 y);
02671 ushort3 const_func __attribute__((overloadable)) rhadd(ushort3 x, ushort3 y);
02672 short4 const_func __attribute__((overloadable)) rhadd(short4 x, short4 y);
02673 ushort4 const_func __attribute__((overloadable)) rhadd(ushort4 x, ushort4 y);
02674 short8 const_func __attribute__((overloadable)) rhadd(short8 x, short8 y);
02675 ushort8 const_func __attribute__((overloadable)) rhadd(ushort8 x, ushort8 y);
02676 short16 const_func __attribute__((overloadable)) rhadd(short16 x, short16 y);
02677 ushort16 const_func __attribute__((overloadable)) rhadd(ushort16 x, ushort16 y);
02678 int const_func __attribute__((overloadable)) rhadd(int x, int y);
02679 uint const_func __attribute__((overloadable)) rhadd(uint x, uint y);
02680 int2 const_func __attribute__((overloadable)) rhadd(int2 x, int2 y);
02681 uint2 const_func __attribute__((overloadable)) rhadd(uint2 x, uint2 y);
02682 int3 const_func __attribute__((overloadable)) rhadd(int3 x, int3 y);
02683 uint3 const_func __attribute__((overloadable)) rhadd(uint3 x, uint3 y);
02684 int4 const_func __attribute__((overloadable)) rhadd(int4 x, int4 y);
02685 uint4 const_func __attribute__((overloadable)) rhadd(uint4 x, uint4 y);
02686 int8 const_func __attribute__((overloadable)) rhadd(int8 x, int8 y);
02687 uint8 const_func __attribute__((overloadable)) rhadd(uint8 x, uint8 y);
02688 int16 const_func __attribute__((overloadable)) rhadd(int16 x, int16 y);
02689 uint16 const_func __attribute__((overloadable)) rhadd(uint16 x, uint16 y);
02690 long const_func __attribute__((overloadable)) rhadd(long x, long y);
02691 ulong const_func __attribute__((overloadable)) rhadd(ulong x, ulong y);
02692 long2 const_func __attribute__((overloadable)) rhadd(long2 x, long2 y);
02693 ulong2 const_func __attribute__((overloadable)) rhadd(ulong2 x, ulong2 y);
02694 long3 const_func __attribute__((overloadable)) rhadd(long3 x, long3 y);
02695 ulong3 const_func __attribute__((overloadable)) rhadd(ulong3 x, ulong3 y);
02696 long4 const_func __attribute__((overloadable)) rhadd(long4 x, long4 y);
02697 ulong4 const_func __attribute__((overloadable)) rhadd(ulong4 x, ulong4 y);
02698 long8 const_func __attribute__((overloadable)) rhadd(long8 x, long8 y);
02699 ulong8 const_func __attribute__((overloadable)) rhadd(ulong8 x, ulong8 y);
02700 long16 const_func __attribute__((overloadable)) rhadd(long16 x, long16 y);
02701 ulong16 const_func __attribute__((overloadable)) rhadd(ulong16 x, ulong16 y);
02702
02703 /**
02704 * Returns min(max(x, minval), maxval).
02705 * Results are undefined if minval > maxval.
02706 */
02707 char const_func __attribute__((overloadable)) clamp(char x, char minval, char maxval);
02708 uchar const_func __attribute__((overloadable)) clamp(uchar x, uchar minval, uchar maxval);
02709 char2 const_func __attribute__((overloadable)) clamp(char2 x, char2 minval, char2 maxval);
02710 uchar2 const_func __attribute__((overloadable)) clamp(uchar2 x, uchar2 minval, uchar2 maxval);
02711 char3 const_func __attribute__((overloadable)) clamp(char3 x, char3 minval, char3 maxval);
02712 uchar3 const_func __attribute__((overloadable)) clamp(uchar3 x, uchar3 minval, uchar3 maxval);
02713 char4 const_func __attribute__((overloadable)) clamp(char4 x, char4 minval, char4 maxval);
02714 uchar4 const_func __attribute__((overloadable)) clamp(uchar4 x, uchar4 minval, uchar4 maxval);
02715 char8 const_func __attribute__((overloadable)) clamp(char8 x, char8 minval, char8 maxval);
02716 uchar8 const_func __attribute__((overloadable)) clamp(uchar8 x, uchar8 minval, uchar8 maxval);
02717 char16 const_func __attribute__((overloadable)) clamp(char16 x, char16 minval, char16 maxval);
02718 uchar16 const_func __attribute__((overloadable)) clamp(uchar16 x, uchar16 minval, uchar16 maxval);
02719 short const_func __attribute__((overloadable)) clamp(short x, short minval, short maxval);
02720 ushort const_func __attribute__((overloadable)) clamp(ushort x, ushort minval, ushort maxval);
02721 short2 const_func __attribute__((overloadable)) clamp(short2 x, short2 minval, short2 maxval);
02722 ushort2 const_func __attribute__((overloadable)) clamp(ushort2 x, ushort2 minval, ushort2 maxval);
02723 short3 const_func __attribute__((overloadable)) clamp(short3 x, short3 minval, short3 maxval);
02724 ushort3 const_func __attribute__((overloadable)) clamp(ushort3 x, ushort3 minval, ushort3 maxval);
02725 short4 const_func __attribute__((overloadable)) clamp(short4 x, short4 minval, short4 maxval);
02726 ushort4 const_func __attribute__((overloadable)) clamp(ushort4 x, ushort4 minval, ushort4 maxval);
02727 short8 const_func __attribute__((overloadable)) clamp(short8 x, short8 minval, short8 maxval);
02728 ushort8 const_func __attribute__((overloadable)) clamp(ushort8 x, ushort8 minval, ushort8 maxval);
02729 short16 const_func __attribute__((overloadable)) clamp(short16 x, short16 minval, short16 maxval);
02730 ushort16 const_func __attribute__((overloadable)) clamp(ushort16 x, ushort16 minval, ushort16 maxval);
02731 int const_func __attribute__((overloadable)) clamp(int x, int minval, int maxval);
02732 uint const_func __attribute__((overloadable)) clamp(uint x, uint minval, uint maxval);
02733 int2 const_func __attribute__((overloadable)) clamp(int2 x, int2 minval, int2 maxval);
02734 uint2 const_func __attribute__((overloadable)) clamp(uint2 x, uint2 minval, uint2 maxval);
02735 int3 const_func __attribute__((overloadable)) clamp(int3 x, int3 minval, int3 maxval);
02736 uint3 const_func __attribute__((overloadable)) clamp(uint3 x, uint3 minval, uint3 maxval);
02737 int4 const_func __attribute__((overloadable)) clamp(int4 x, int4 minval, int4 maxval);
02738 uint4 const_func __attribute__((overloadable)) clamp(uint4 x, uint4 minval, uint4 maxval);
02739 int8 const_func __attribute__((overloadable)) clamp(int8 x, int8 minval, int8 maxval);
02740 uint8 const_func __attribute__((overloadable)) clamp(uint8 x, uint8 minval, uint8 maxval);
02741 int16 const_func __attribute__((overloadable)) clamp(int16 x, int16 minval, int16 maxval);
02742 uint16 const_func __attribute__((overloadable)) clamp(uint16 x, uint16 minval, uint16 maxval);
02743 long const_func __attribute__((overloadable)) clamp(long x, long minval, long maxval);
02744 ulong const_func __attribute__((overloadable)) clamp(ulong x, ulong minval, ulong maxval);
02745 long2 const_func __attribute__((overloadable)) clamp(long2 x, long2 minval, long2 maxval);
02746 ulong2 const_func __attribute__((overloadable)) clamp(ulong2 x, ulong2 minval, ulong2 maxval);
02747 long3 const_func __attribute__((overloadable)) clamp(long3 x, long3 minval, long3 maxval);
02748 ulong3 const_func __attribute__((overloadable)) clamp(ulong3 x, ulong3 minval, ulong3 maxval);
02749 long4 const_func __attribute__((overloadable)) clamp(long4 x, long4 minval, long4 maxval);
```

```
02750 ulong4 const_func __attribute__((overloadable)) clamp(ulong4 x, ulong4 minval, ulong4 maxval);
02751 long8 const_func __attribute__((overloadable)) clamp(long8 x, long8 minval, long8 maxval);
02752 ulong8 const_func __attribute__((overloadable)) clamp(ulong8 x, ulong8 minval, ulong8 maxval);
02753 long16 const_func __attribute__((overloadable)) clamp(long16 x, long16 minval, long16 maxval);
02754 ulong16 const_func __attribute__((overloadable)) clamp(ulong16 x, ulong16 minval, ulong16 maxval);
02755 char const_func __attribute__((overloadable)) clamp(char x, char minval, char maxval);
02756 uchar const_func __attribute__((overloadable)) clamp(uchar x, uchar minval, uchar maxval);
02757 char2 const_func __attribute__((overloadable)) clamp(char2 x, char minval, char maxval);
02758 uchar2 const_func __attribute__((overloadable)) clamp(uchar2 x, uchar minval, uchar maxval);
02759 char3 const_func __attribute__((overloadable)) clamp(char3 x, char minval, char maxval);
02760 uchar3 const_func __attribute__((overloadable)) clamp(uchar3 x, uchar minval, uchar maxval);
02761 char4 const_func __attribute__((overloadable)) clamp(char4 x, char minval, char maxval);
02762 uchar4 const_func __attribute__((overloadable)) clamp(uchar4 x, uchar minval, uchar maxval);
02763 char8 const_func __attribute__((overloadable)) clamp(char8 x, char minval, char maxval);
02764 uchar8 const_func __attribute__((overloadable)) clamp(uchar8 x, uchar minval, uchar maxval);
02765 char16 const_func __attribute__((overloadable)) clamp(char16 x, char minval, char maxval);
02766 uchar16 const_func __attribute__((overloadable)) clamp(uchar16 x, uchar minval, uchar maxval);
02767 short const_func __attribute__((overloadable)) clamp(short x, short minval, short maxval);
02768 ushort const_func __attribute__((overloadable)) clamp(ushort x, ushort minval, ushort maxval);
02769 short2 const_func __attribute__((overloadable)) clamp(short2 x, short minval, short maxval);
02770 ushort2 const_func __attribute__((overloadable)) clamp(ushort2 x, ushort minval, ushort maxval);
02771 short3 const_func __attribute__((overloadable)) clamp(short3 x, short minval, short maxval);
02772 ushort3 const_func __attribute__((overloadable)) clamp(ushort3 x, ushort minval, ushort maxval);
02773 short4 const_func __attribute__((overloadable)) clamp(short4 x, short minval, short maxval);
02774 ushort4 const_func __attribute__((overloadable)) clamp(ushort4 x, ushort minval, ushort maxval);
02775 short8 const_func __attribute__((overloadable)) clamp(short8 x, short minval, short maxval);
02776 ushort8 const_func __attribute__((overloadable)) clamp(ushort8 x, ushort minval, ushort maxval);
02777 short16 const_func __attribute__((overloadable)) clamp(short16 x, short minval, short maxval);
02778 ushort16 const_func __attribute__((overloadable)) clamp(ushort16 x, ushort minval, ushort maxval);
02779 int const_func __attribute__((overloadable)) clamp(int x, int minval, int maxval);
02780 uint const_func __attribute__((overloadable)) clamp(uint x, uint minval, uint maxval);
02781 int2 const_func __attribute__((overloadable)) clamp(int2 x, int minval, int maxval);
02782 uint2 const_func __attribute__((overloadable)) clamp(uint2 x, uint minval, uint maxval);
02783 int3 const_func __attribute__((overloadable)) clamp(int3 x, int minval, int maxval);
02784 uint3 const_func __attribute__((overloadable)) clamp(uint3 x, uint minval, uint maxval);
02785 int4 const_func __attribute__((overloadable)) clamp(int4 x, int minval, int maxval);
02786 uint4 const_func __attribute__((overloadable)) clamp(uint4 x, uint minval, uint maxval);
02787 int8 const_func __attribute__((overloadable)) clamp(int8 x, int minval, int maxval);
02788 uint8 const_func __attribute__((overloadable)) clamp(uint8 x, uint minval, uint maxval);
02789 int16 const_func __attribute__((overloadable)) clamp(int16 x, int minval, int maxval);
02790 uint16 const_func __attribute__((overloadable)) clamp(uint16 x, uint minval, uint maxval);
02791 long const_func __attribute__((overloadable)) clamp(long x, long minval, long maxval);
02792 ulong const_func __attribute__((overloadable)) clamp(ulong x, ulong minval, ulong maxval);
02793 long2 const_func __attribute__((overloadable)) clamp(long2 x, long minval, long maxval);
02794 ulong2 const_func __attribute__((overloadable)) clamp(ulong2 x, ulong minval, ulong maxval);
02795 long3 const_func __attribute__((overloadable)) clamp(long3 x, long minval, long maxval);
02796 ulong3 const_func __attribute__((overloadable)) clamp(ulong3 x, ulong minval, ulong maxval);
02797 long4 const_func __attribute__((overloadable)) clamp(long4 x, long minval, long maxval);
02798 ulong4 const_func __attribute__((overloadable)) clamp(ulong4 x, ulong minval, ulong maxval);
02799 long8 const_func __attribute__((overloadable)) clamp(long8 x, long minval, long maxval);
02800 ulong8 const_func __attribute__((overloadable)) clamp(ulong8 x, ulong minval, ulong maxval);
02801 long16 const_func __attribute__((overloadable)) clamp(long16 x, long minval, long maxval);
02802 ulong16 const_func __attribute__((overloadable)) clamp(ulong16 x, ulong minval, ulong maxval);
02803
02804 /**
02805 * Returns the number of leading 0-bits in x, starting
02806 * at the most significant bit position.
02807 */
02808 char const_func __attribute__((overloadable)) clz(char x);
02809 uchar const_func __attribute__((overloadable)) clz(uchar x);
02810 char2 const_func __attribute__((overloadable)) clz(char2 x);
02811 uchar2 const_func __attribute__((overloadable)) clz(uchar2 x);
02812 char3 const_func __attribute__((overloadable)) clz(char3 x);
02813 uchar3 const_func __attribute__((overloadable)) clz(uchar3 x);
02814 char4 const_func __attribute__((overloadable)) clz(char4 x);
02815 uchar4 const_func __attribute__((overloadable)) clz(uchar4 x);
02816 char8 const_func __attribute__((overloadable)) clz(char8 x);
02817 uchar8 const_func __attribute__((overloadable)) clz(uchar8 x);
02818 char16 const_func __attribute__((overloadable)) clz(char16 x);
02819 uchar16 const_func __attribute__((overloadable)) clz(uchar16 x);
02820 short const_func __attribute__((overloadable)) clz(short x);
02821 ushort const_func __attribute__((overloadable)) clz(ushort x);
02822 short2 const_func __attribute__((overloadable)) clz(short2 x);
02823 ushort2 const_func __attribute__((overloadable)) clz(ushort2 x);
02824 short3 const_func __attribute__((overloadable)) clz(short3 x);
02825 ushort3 const_func __attribute__((overloadable)) clz(ushort3 x);
02826 short4 const_func __attribute__((overloadable)) clz(short4 x);
02827 ushort4 const_func __attribute__((overloadable)) clz(ushort4 x);
02828 short8 const_func __attribute__((overloadable)) clz(short8 x);
02829 ushort8 const_func __attribute__((overloadable)) clz(ushort8 x);
02830 short16 const_func __attribute__((overloadable)) clz(short16 x);
02831 ushort16 const_func __attribute__((overloadable)) clz(ushort16 x);
02832 int const_func __attribute__((overloadable)) clz(int x);
02833 uint const_func __attribute__((overloadable)) clz(uint x);
02834 int2 const_func __attribute__((overloadable)) clz(int2 x);
02835 uint2 const_func __attribute__((overloadable)) clz(uint2 x);
02836 int3 const_func __attribute__((overloadable)) clz(int3 x);
```

```
02837 uint3 const_func __attribute__((overloadable)) clz(uint3 x);
02838 int4 const_func __attribute__((overloadable)) clz(int4 x);
02839 uint4 const_func __attribute__((overloadable)) clz(uint4 x);
02840 int8 const_func __attribute__((overloadable)) clz(int8 x);
02841 uint8 const_func __attribute__((overloadable)) clz(uint8 x);
02842 int16 const_func __attribute__((overloadable)) clz(int16 x);
02843 uint16 const_func __attribute__((overloadable)) clz(uint16 x);
02844 long const_func __attribute__((overloadable)) clz(long x);
02845 ulong const_func __attribute__((overloadable)) clz(ulong x);
02846 long2 const_func __attribute__((overloadable)) clz(long2 x);
02847 ulong2 const_func __attribute__((overloadable)) clz(ulong2 x);
02848 long3 const_func __attribute__((overloadable)) clz(long3 x);
02849 ulong3 const_func __attribute__((overloadable)) clz(ulong3 x);
02850 long4 const_func __attribute__((overloadable)) clz(long4 x);
02851 ulong4 const_func __attribute__((overloadable)) clz(ulong4 x);
02852 long8 const_func __attribute__((overloadable)) clz(long8 x);
02853 ulong8 const_func __attribute__((overloadable)) clz(ulong8 x);
02854 long16 const_func __attribute__((overloadable)) clz(long16 x);
02855 ulong16 const_func __attribute__((overloadable)) clz(ulong16 x);
02856
02857 /**
02858 * Returns mul_hi(a, b) + c.
02859 */
02860 char const_func __attribute__((overloadable)) mad_hi(char a, char b, char c);
02861 uchar const_func __attribute__((overloadable)) mad_hi(uchar a, uchar b, uchar c);
02862 char2 const_func __attribute__((overloadable)) mad_hi(char2 a, char2 b, char2 c);
02863 uchar2 const_func __attribute__((overloadable)) mad_hi(uchar2 a, uchar2 b, uchar2 c);
02864 char3 const_func __attribute__((overloadable)) mad_hi(char3 a, char3 b, char3 c);
02865 uchar3 const_func __attribute__((overloadable)) mad_hi(uchar3 a, uchar3 b, uchar3 c);
02866 char4 const_func __attribute__((overloadable)) mad_hi(char4 a, char4 b, char4 c);
02867 uchar4 const_func __attribute__((overloadable)) mad_hi(uchar4 a, uchar4 b, uchar4 c);
02868 char8 const_func __attribute__((overloadable)) mad_hi(char8 a, char8 b, char8 c);
02869 uchar8 const_func __attribute__((overloadable)) mad_hi(uchar8 a, uchar8 b, uchar8 c);
02870 char16 const_func __attribute__((overloadable)) mad_hi(char16 a, char16 b, char16 c);
02871 uchar16 const_func __attribute__((overloadable)) mad_hi(uchar16 a, uchar16 b, uchar16 c);
02872 short const_func __attribute__((overloadable)) mad_hi(short a, short b, short c);
02873 ushort const_func __attribute__((overloadable)) mad_hi(ushort a, ushort b, ushort c);
02874 short2 const_func __attribute__((overloadable)) mad_hi(short2 a, short2 b, short2 c);
02875 ushort2 const_func __attribute__((overloadable)) mad_hi(ushort2 a, ushort2 b, ushort2 c);
02876 short3 const_func __attribute__((overloadable)) mad_hi(short3 a, short3 b, short3 c);
02877 ushort3 const_func __attribute__((overloadable)) mad_hi(ushort3 a, ushort3 b, ushort3 c);
02878 short4 const_func __attribute__((overloadable)) mad_hi(short4 a, short4 b, short4 c);
02879 ushort4 const_func __attribute__((overloadable)) mad_hi(ushort4 a, ushort4 b, ushort4 c);
02880 short8 const_func __attribute__((overloadable)) mad_hi(short8 a, short8 b, short8 c);
02881 ushort8 const_func __attribute__((overloadable)) mad_hi(ushort8 a, ushort8 b, ushort8 c);
02882 short16 const_func __attribute__((overloadable)) mad_hi(short16 a, short16 b, short16 c);
02883 ushort16 const_func __attribute__((overloadable)) mad_hi(ushort16 a, ushort16 b, ushort16 c);
02884 int const_func __attribute__((overloadable)) mad_hi(int a, int b, int c);
02885 uint const_func __attribute__((overloadable)) mad_hi(uint a, uint b, uint c);
02886 int2 const_func __attribute__((overloadable)) mad_hi(int2 a, int2 b, int2 c);
02887 uint2 const_func __attribute__((overloadable)) mad_hi(uint2 a, uint2 b, uint2 c);
02888 int3 const_func __attribute__((overloadable)) mad_hi(int3 a, int3 b, int3 c);
02889 uint3 const_func __attribute__((overloadable)) mad_hi(uint3 a, uint3 b, uint3 c);
02890 int4 const_func __attribute__((overloadable)) mad_hi(int4 a, int4 b, int4 c);
02891 uint4 const_func __attribute__((overloadable)) mad_hi(uint4 a, uint4 b, uint4 c);
02892 int8 const_func __attribute__((overloadable)) mad_hi(int8 a, int8 b, int8 c);
02893 uint8 const_func __attribute__((overloadable)) mad_hi(uint8 a, uint8 b, uint8 c);
02894 int16 const_func __attribute__((overloadable)) mad_hi(int16 a, int16 b, int16 c);
02895 uint16 const_func __attribute__((overloadable)) mad_hi(uint16 a, uint16 b, uint16 c);
02896 long const_func __attribute__((overloadable)) mad_hi(long a, long b, long c);
02897 ulong const_func __attribute__((overloadable)) mad_hi(ulong a, ulong b, ulong c);
02898 long2 const_func __attribute__((overloadable)) mad_hi(long2 a, long2 b, long2 c);
02899 ulong2 const_func __attribute__((overloadable)) mad_hi(ulong2 a, ulong2 b, ulong2 c);
02900 long3 const_func __attribute__((overloadable)) mad_hi(long3 a, long3 b, long3 c);
02901 ulong3 const_func __attribute__((overloadable)) mad_hi(ulong3 a, ulong3 b, ulong3 c);
02902 long4 const_func __attribute__((overloadable)) mad_hi(long4 a, long4 b, long4 c);
02903 ulong4 const_func __attribute__((overloadable)) mad_hi(ulong4 a, ulong4 b, ulong4 c);
02904 long8 const_func __attribute__((overloadable)) mad_hi(long8 a, long8 b, long8 c);
02905 ulong8 const_func __attribute__((overloadable)) mad_hi(ulong8 a, ulong8 b, ulong8 c);
02906 long16 const_func __attribute__((overloadable)) mad_hi(long16 a, long16 b, long16 c);
02907 ulong16 const_func __attribute__((overloadable)) mad_hi(ulong16 a, ulong16 b, ulong16 c);
02908
02909 /**
02910 * Returns a * b + c and saturates the result.
02911 */
02912 char const_func __attribute__((overloadable)) mad_sat(char a, char b, char c);
02913 uchar const_func __attribute__((overloadable)) mad_sat(uchar a, uchar b, uchar c);
02914 char2 const_func __attribute__((overloadable)) mad_sat(char2 a, char2 b, char2 c);
02915 uchar2 const_func __attribute__((overloadable)) mad_sat(uchar2 a, uchar2 b, uchar2 c);
02916 char3 const_func __attribute__((overloadable)) mad_sat(char3 a, char3 b, char3 c);
02917 uchar3 const_func __attribute__((overloadable)) mad_sat(uchar3 a, uchar3 b, uchar3 c);
02918 char4 const_func __attribute__((overloadable)) mad_sat(char4 a, char4 b, char4 c);
02919 uchar4 const_func __attribute__((overloadable)) mad_sat(uchar4 a, uchar4 b, uchar4 c);
02920 char8 const_func __attribute__((overloadable)) mad_sat(char8 a, char8 b, char8 c);
02921 uchar8 const_func __attribute__((overloadable)) mad_sat(uchar8 a, uchar8 b, uchar8 c);
02922 char16 const_func __attribute__((overloadable)) mad_sat(char16 a, char16 b, char16 c);
02923 uchar16 const_func __attribute__((overloadable)) mad_sat(uchar16 a, uchar16 b, uchar16 c);
```

```
02924 short const_func __attribute__((overloadable)) mad_sat(short a, short b, short c);
02925 ushort const_func __attribute__((overloadable)) mad_sat(ushort a, ushort b, ushort c);
02926 short2 const_func __attribute__((overloadable)) mad_sat(short2 a, short2 b, short2 c);
02927 ushort2 const_func __attribute__((overloadable)) mad_sat(ushort2 a, ushort2 b, ushort2 c);
02928 short3 const_func __attribute__((overloadable)) mad_sat(short3 a, short3 b, short3 c);
02929 ushort3 const_func __attribute__((overloadable)) mad_sat(ushort3 a, ushort3 b, ushort3 c);
02930 short4 const_func __attribute__((overloadable)) mad_sat(short4 a, short4 b, short4 c);
02931 ushort4 const_func __attribute__((overloadable)) mad_sat(ushort4 a, ushort4 b, ushort4 c);
02932 short8 const_func __attribute__((overloadable)) mad_sat(short8 a, short8 b, short8 c);
02933 ushort8 const_func __attribute__((overloadable)) mad_sat(ushort8 a, ushort8 b, ushort8 c);
02934 short16 const_func __attribute__((overloadable)) mad_sat(short16 a, short16 b, short16 c);
02935 ushort16 const_func __attribute__((overloadable)) mad_sat(ushort16 a, ushort16 b, ushort16 c);
02936 int const_func __attribute__((overloadable)) mad_sat(int a, int b, int c);
02937 uint const_func __attribute__((overloadable)) mad_sat(uint a, uint b, uint c);
02938 int2 const_func __attribute__((overloadable)) mad_sat(int2 a, int2 b, int2 c);
02939 uint2 const_func __attribute__((overloadable)) mad_sat(uint2 a, uint2 b, uint2 c);
02940 int3 const_func __attribute__((overloadable)) mad_sat(int3 a, int3 b, int3 c);
02941 uint3 const_func __attribute__((overloadable)) mad_sat(uint3 a, uint3 b, uint3 c);
02942 int4 const_func __attribute__((overloadable)) mad_sat(int4 a, int4 b, int4 c);
02943 uint4 const_func __attribute__((overloadable)) mad_sat(uint4 a, uint4 b, uint4 c);
02944 int8 const_func __attribute__((overloadable)) mad_sat(int8 a, int8 b, int8 c);
02945 uint8 const_func __attribute__((overloadable)) mad_sat(uint8 a, uint8 b, uint8 c);
02946 int16 const_func __attribute__((overloadable)) mad_sat(int16 a, int16 b, int16 c);
02947 uint16 const_func __attribute__((overloadable)) mad_sat(uint16 a, uint16 b, uint16 c);
02948 long const_func __attribute__((overloadable)) mad_sat(long a, long b, long c);
02949 ulong const_func __attribute__((overloadable)) mad_sat(ulong a, ulong b, ulong c);
02950 long2 const_func __attribute__((overloadable)) mad_sat(long2 a, long2 b, long2 c);
02951 ulong2 const_func __attribute__((overloadable)) mad_sat(ulong2 a, ulong2 b, ulong2 c);
02952 long3 const_func __attribute__((overloadable)) mad_sat(long3 a, long3 b, long3 c);
02953 ulong3 const_func __attribute__((overloadable)) mad_sat(ulong3 a, ulong3 b, ulong3 c);
02954 long4 const_func __attribute__((overloadable)) mad_sat(long4 a, long4 b, long4 c);
02955 ulong4 const_func __attribute__((overloadable)) mad_sat(ulong4 a, ulong4 b, ulong4 c);
02956 long8 const_func __attribute__((overloadable)) mad_sat(long8 a, long8 b, long8 c);
02957 ulong8 const_func __attribute__((overloadable)) mad_sat(ulong8 a, ulong8 b, ulong8 c);
02958 long16 const_func __attribute__((overloadable)) mad_sat(long16 a, long16 b, long16 c);
02959 ulong16 const_func __attribute__((overloadable)) mad_sat(ulong16 a, ulong16 b, ulong16 c);
02960
02961 /**
02962 * Returns y if x < y, otherwise it returns x.
02963 */
02964 char const_func __attribute__((overloadable)) max(char x, char y);
02965 uchar const_func __attribute__((overloadable)) max(uchar x, uchar y);
02966 char2 const_func __attribute__((overloadable)) max(char2 x, char2 y);
02967 uchar2 const_func __attribute__((overloadable)) max(uchar2 x, uchar2 y);
02968 char3 const_func __attribute__((overloadable)) max(char3 x, char3 y);
02969 uchar3 const_func __attribute__((overloadable)) max(uchar3 x, uchar3 y);
02970 char4 const_func __attribute__((overloadable)) max(char4 x, char4 y);
02971 uchar4 const_func __attribute__((overloadable)) max(uchar4 x, uchar4 y);
02972 char8 const_func __attribute__((overloadable)) max(char8 x, char8 y);
02973 uchar8 const_func __attribute__((overloadable)) max(uchar8 x, uchar8 y);
02974 char16 const_func __attribute__((overloadable)) max(char16 x, char16 y);
02975 uchar16 const_func __attribute__((overloadable)) max(uchar16 x, uchar16 y);
02976 short const_func __attribute__((overloadable)) max(short x, short y);
02977 ushort const_func __attribute__((overloadable)) max(ushort x, ushort y);
02978 short2 const_func __attribute__((overloadable)) max(short2 x, short2 y);
02979 ushort2 const_func __attribute__((overloadable)) max(ushort2 x, ushort2 y);
02980 short3 const_func __attribute__((overloadable)) max(short3 x, short3 y);
02981 ushort3 const_func __attribute__((overloadable)) max(ushort3 x, ushort3 y);
02982 short4 const_func __attribute__((overloadable)) max(short4 x, short4 y);
02983 ushort4 const_func __attribute__((overloadable)) max(ushort4 x, ushort4 y);
02984 short8 const_func __attribute__((overloadable)) max(short8 x, short8 y);
02985 ushort8 const_func __attribute__((overloadable)) max(ushort8 x, ushort8 y);
02986 short16 const_func __attribute__((overloadable)) max(short16 x, short16 y);
02987 ushort16 const_func __attribute__((overloadable)) max(ushort16 x, ushort16 y);
02988 int const_func __attribute__((overloadable)) max(int x, int y);
02989 uint const_func __attribute__((overloadable)) max(uint x, uint y);
02990 int2 const_func __attribute__((overloadable)) max(int2 x, int2 y);
02991 uint2 const_func __attribute__((overloadable)) max(uint2 x, uint2 y);
02992 int3 const_func __attribute__((overloadable)) max(int3 x, int3 y);
02993 uint3 const_func __attribute__((overloadable)) max(uint3 x, uint3 y);
02994 int4 const_func __attribute__((overloadable)) max(int4 x, int4 y);
02995 uint4 const_func __attribute__((overloadable)) max(uint4 x, uint4 y);
02996 int8 const_func __attribute__((overloadable)) max(int8 x, int8 y);
02997 uint8 const_func __attribute__((overloadable)) max(uint8 x, uint8 y);
02998 int16 const_func __attribute__((overloadable)) max(int16 x, int16 y);
02999 uint16 const_func __attribute__((overloadable)) max(uint16 x, uint16 y);
03000 long const_func __attribute__((overloadable)) max(long x, long y);
03001 ulong const_func __attribute__((overloadable)) max(ulong x, ulong y);
03002 long2 const_func __attribute__((overloadable)) max(long2 x, long2 y);
03003 ulong2 const_func __attribute__((overloadable)) max(ulong2 x, ulong2 y);
03004 long3 const_func __attribute__((overloadable)) max(long3 x, long3 y);
03005 ulong3 const_func __attribute__((overloadable)) max(ulong3 x, ulong3 y);
03006 long4 const_func __attribute__((overloadable)) max(long4 x, long4 y);
03007 ulong4 const_func __attribute__((overloadable)) max(ulong4 x, ulong4 y);
03008 long8 const_func __attribute__((overloadable)) max(long8 x, long8 y);
03009 ulong8 const_func __attribute__((overloadable)) max(ulong8 x, ulong8 y);
03010 long16 const_func __attribute__((overloadable)) max(long16 x, long16 y);
```

```
03011 ulong16 const_func __attribute__((overloadable)) max(ulong16 x, ulong16 y);
03012 char const_func __attribute__((overloadable)) max(char x, char y);
03013 uchar const_func __attribute__((overloadable)) max(uchar x, uchar y);
03014 char2 const_func __attribute__((overloadable)) max(char2 x, char y);
03015 uchar2 const_func __attribute__((overloadable)) max(uchar2 x, uchar y);
03016 char3 const_func __attribute__((overloadable)) max(char3 x, char y);
03017 uchar3 const_func __attribute__((overloadable)) max(uchar3 x, uchar y);
03018 char4 const_func __attribute__((overloadable)) max(char4 x, char y);
03019 uchar4 const_func __attribute__((overloadable)) max(uchar4 x, uchar y);
03020 char8 const_func __attribute__((overloadable)) max(char8 x, char y);
03021 uchar8 const_func __attribute__((overloadable)) max(uchar8 x, uchar y);
03022 char16 const_func __attribute__((overloadable)) max(char16 x, char y);
03023 uchar16 const_func __attribute__((overloadable)) max(uchar16 x, uchar y);
03024 short const_func __attribute__((overloadable)) max(short x, short y);
03025 ushort const_func __attribute__((overloadable)) max(ushort x, ushort y);
03026 short2 const_func __attribute__((overloadable)) max(short2 x, short y);
03027 ushort2 const_func __attribute__((overloadable)) max(ushort2 x, ushort y);
03028 short3 const_func __attribute__((overloadable)) max(short3 x, short y);
03029 ushort3 const_func __attribute__((overloadable)) max(ushort3 x, ushort y);
03030 short4 const_func __attribute__((overloadable)) max(short4 x, short y);
03031 ushort4 const_func __attribute__((overloadable)) max(ushort4 x, ushort y);
03032 short8 const_func __attribute__((overloadable)) max(short8 x, short y);
03033 ushort8 const_func __attribute__((overloadable)) max(ushort8 x, ushort y);
03034 short16 const_func __attribute__((overloadable)) max(short16 x, short y);
03035 ushort16 const_func __attribute__((overloadable)) max(ushort16 x, ushort y);
03036 int const_func __attribute__((overloadable)) max(int x, int y);
03037 uint const_func __attribute__((overloadable)) max(uint x, uint y);
03038 int2 const_func __attribute__((overloadable)) max(int2 x, int y);
03039 uint2 const_func __attribute__((overloadable)) max(uint2 x, uint y);
03040 int3 const_func __attribute__((overloadable)) max(int3 x, int y);
03041 uint3 const_func __attribute__((overloadable)) max(uint3 x, uint y);
03042 int4 const_func __attribute__((overloadable)) max(int4 x, int y);
03043 uint4 const_func __attribute__((overloadable)) max(uint4 x, uint y);
03044 int8 const_func __attribute__((overloadable)) max(int8 x, int y);
03045 uint8 const_func __attribute__((overloadable)) max(uint8 x, uint y);
03046 int16 const_func __attribute__((overloadable)) max(int16 x, int y);
03047 uint16 const_func __attribute__((overloadable)) max(uint16 x, uint y);
03048 long const_func __attribute__((overloadable)) max(long x, long y);
03049 ulong const_func __attribute__((overloadable)) max(ulong x, ulong y);
03050 long2 const_func __attribute__((overloadable)) max(long2 x, long y);
03051 ulong2 const_func __attribute__((overloadable)) max(ulong2 x, ulong y);
03052 long3 const_func __attribute__((overloadable)) max(long3 x, long y);
03053 ulong3 const_func __attribute__((overloadable)) max(ulong3 x, ulong y);
03054 long4 const_func __attribute__((overloadable)) max(long4 x, long y);
03055 ulong4 const_func __attribute__((overloadable)) max(ulong4 x, ulong y);
03056 long8 const_func __attribute__((overloadable)) max(long8 x, long y);
03057 ulong8 const_func __attribute__((overloadable)) max(ulong8 x, ulong y);
03058 long16 const_func __attribute__((overloadable)) max(long16 x, long y);
03059 ulong16 const_func __attribute__((overloadable)) max(ulong16 x, ulong y);
03060
03061 /**
03062 * Returns y if y < x, otherwise it returns x.
03063 */
03064 char const_func __attribute__((overloadable)) min(char x, char y);
03065 uchar const_func __attribute__((overloadable)) min(uchar x, uchar y);
03066 char2 const_func __attribute__((overloadable)) min(char2 x, char2 y);
03067 uchar2 const_func __attribute__((overloadable)) min(uchar2 x, uchar2 y);
03068 char3 const_func __attribute__((overloadable)) min(char3 x, char3 y);
03069 uchar3 const_func __attribute__((overloadable)) min(uchar3 x, uchar3 y);
03070 char4 const_func __attribute__((overloadable)) min(char4 x, char4 y);
03071 uchar4 const_func __attribute__((overloadable)) min(uchar4 x, uchar4 y);
03072 char8 const_func __attribute__((overloadable)) min(char8 x, char8 y);
03073 uchar8 const_func __attribute__((overloadable)) min(uchar8 x, uchar8 y);
03074 char16 const_func __attribute__((overloadable)) min(char16 x, char16 y);
03075 uchar16 const_func __attribute__((overloadable)) min(uchar16 x, uchar16 y);
03076 short const_func __attribute__((overloadable)) min(short x, short y);
03077 ushort const_func __attribute__((overloadable)) min(ushort x, ushort y);
03078 short2 const_func __attribute__((overloadable)) min(short2 x, short2 y);
03079 ushort2 const_func __attribute__((overloadable)) min(ushort2 x, ushort2 y);
03080 short3 const_func __attribute__((overloadable)) min(short3 x, short3 y);
03081 ushort3 const_func __attribute__((overloadable)) min(ushort3 x, ushort3 y);
03082 short4 const_func __attribute__((overloadable)) min(short4 x, short4 y);
03083 ushort4 const_func __attribute__((overloadable)) min(ushort4 x, ushort4 y);
03084 short8 const_func __attribute__((overloadable)) min(short8 x, short8 y);
03085 ushort8 const_func __attribute__((overloadable)) min(ushort8 x, ushort8 y);
03086 short16 const_func __attribute__((overloadable)) min(short16 x, short16 y);
03087 ushort16 const_func __attribute__((overloadable)) min(ushort16 x, ushort16 y);
03088 int const_func __attribute__((overloadable)) min(int x, int y);
03089 uint const_func __attribute__((overloadable)) min(uint x, uint y);
03090 int2 const_func __attribute__((overloadable)) min(int2 x, int2 y);
03091 uint2 const_func __attribute__((overloadable)) min(uint2 x, uint2 y);
03092 int3 const_func __attribute__((overloadable)) min(int3 x, int3 y);
03093 uint3 const_func __attribute__((overloadable)) min(uint3 x, uint3 y);
03094 int4 const_func __attribute__((overloadable)) min(int4 x, int4 y);
03095 uint4 const_func __attribute__((overloadable)) min(uint4 x, uint4 y);
03096 int8 const_func __attribute__((overloadable)) min(int8 x, int8 y);
03097 uint8 const_func __attribute__((overloadable)) min(uint8 x, uint8 y);
```

```
03098 int16 const_func __attribute__((overloadable)) min(int16 x, int16 y);
03099 uint16 const_func __attribute__((overloadable)) min(uint16 x, uint16 y);
03100 long const_func __attribute__((overloadable)) min(long x, long y);
03101 ulong const_func __attribute__((overloadable)) min(ulong x, ulong y);
03102 long2 const_func __attribute__((overloadable)) min(long2 x, long2 y);
03103 ulong2 const_func __attribute__((overloadable)) min(ulong2 x, ulong2 y);
03104 long3 const_func __attribute__((overloadable)) min(long3 x, long3 y);
03105 ulong3 const_func __attribute__((overloadable)) min(ulong3 x, ulong3 y);
03106 long4 const_func __attribute__((overloadable)) min(long4 x, long4 y);
03107 ulong4 const_func __attribute__((overloadable)) min(ulong4 x, ulong4 y);
03108 long8 const_func __attribute__((overloadable)) min(long8 x, long8 y);
03109 ulong8 const_func __attribute__((overloadable)) min(ulong8 x, ulong8 y);
03110 long16 const_func __attribute__((overloadable)) min(long16 x, long16 y);
03111 ulong16 const_func __attribute__((overloadable)) min(ulong16 x, ulong16 y);
03112 char const_func __attribute__((overloadable)) min(char x, char y);
03113 uchar const_func __attribute__((overloadable)) min(uchar x, uchar y);
03114 char2 const_func __attribute__((overloadable)) min(char2 x, char y);
03115 uchar2 const_func __attribute__((overloadable)) min(uchar2 x, uchar y);
03116 char3 const_func __attribute__((overloadable)) min(char3 x, char y);
03117 uchar3 const_func __attribute__((overloadable)) min(uchar3 x, uchar y);
03118 char4 const_func __attribute__((overloadable)) min(char4 x, char y);
03119 uchar4 const_func __attribute__((overloadable)) min(uchar4 x, uchar y);
03120 char8 const_func __attribute__((overloadable)) min(char8 x, char y);
03121 uchar8 const_func __attribute__((overloadable)) min(uchar8 x, uchar y);
03122 char16 const_func __attribute__((overloadable)) min(char16 x, char y);
03123 uchar16 const_func __attribute__((overloadable)) min(uchar16 x, uchar y);
03124 short const_func __attribute__((overloadable)) min(short x, short y);
03125 ushort const_func __attribute__((overloadable)) min(ushort x, ushort y);
03126 short2 const_func __attribute__((overloadable)) min(short2 x, short y);
03127 ushort2 const_func __attribute__((overloadable)) min(ushort2 x, ushort y);
03128 short3 const_func __attribute__((overloadable)) min(short3 x, short y);
03129 ushort3 const_func __attribute__((overloadable)) min(ushort3 x, ushort y);
03130 short4 const_func __attribute__((overloadable)) min(short4 x, short y);
03131 ushort4 const_func __attribute__((overloadable)) min(ushort4 x, ushort y);
03132 short8 const_func __attribute__((overloadable)) min(short8 x, short y);
03133 ushort8 const_func __attribute__((overloadable)) min(ushort8 x, ushort y);
03134 short16 const_func __attribute__((overloadable)) min(short16 x, short y);
03135 ushort16 const_func __attribute__((overloadable)) min(ushort16 x, ushort y);
03136 int const_func __attribute__((overloadable)) min(int x, int y);
03137 uint const_func __attribute__((overloadable)) min(uint x, uint y);
03138 int2 const_func __attribute__((overloadable)) min(int2 x, int y);
03139 uint2 const_func __attribute__((overloadable)) min(uint2 x, uint y);
03140 int3 const_func __attribute__((overloadable)) min(int3 x, int y);
03141 uint3 const_func __attribute__((overloadable)) min(uint3 x, uint y);
03142 int4 const_func __attribute__((overloadable)) min(int4 x, int y);
03143 uint4 const_func __attribute__((overloadable)) min(uint4 x, uint y);
03144 int8 const_func __attribute__((overloadable)) min(int8 x, int y);
03145 uint8 const_func __attribute__((overloadable)) min(uint8 x, uint y);
03146 int16 const_func __attribute__((overloadable)) min(int16 x, int y);
03147 uint16 const_func __attribute__((overloadable)) min(uint16 x, uint y);
03148 long const_func __attribute__((overloadable)) min(long x, long y);
03149 ulong const_func __attribute__((overloadable)) min(ulong x, ulong y);
03150 long2 const_func __attribute__((overloadable)) min(long2 x, long y);
03151 ulong2 const_func __attribute__((overloadable)) min(ulong2 x, ulong y);
03152 long3 const_func __attribute__((overloadable)) min(long3 x, long y);
03153 ulong3 const_func __attribute__((overloadable)) min(ulong3 x, ulong y);
03154 long4 const_func __attribute__((overloadable)) min(long4 x, long y);
03155 ulong4 const_func __attribute__((overloadable)) min(ulong4 x, ulong y);
03156 long8 const_func __attribute__((overloadable)) min(long8 x, long y);
03157 ulong8 const_func __attribute__((overloadable)) min(ulong8 x, ulong y);
03158 long16 const_func __attribute__((overloadable)) min(long16 x, long y);
03159 ulong16 const_func __attribute__((overloadable)) min(ulong16 x, ulong y);
03160
03161 /**
03162 * Computes x * y and returns the high half of the
03163 * product of x and y.
03164 */
03165 char const_func __attribute__((overloadable)) mul_hi(char x, char y);
03166 uchar const_func __attribute__((overloadable)) mul_hi(uchar x, uchar y);
03167 char2 const_func __attribute__((overloadable)) mul_hi(char2 x, char2 y);
03168 uchar2 const_func __attribute__((overloadable)) mul_hi(uchar2 x, uchar2 y);
03169 char3 const_func __attribute__((overloadable)) mul_hi(char3 x, char3 y);
03170 uchar3 const_func __attribute__((overloadable)) mul_hi(uchar3 x, uchar3 y);
03171 char4 const_func __attribute__((overloadable)) mul_hi(char4 x, char4 y);
03172 uchar4 const_func __attribute__((overloadable)) mul_hi(uchar4 x, uchar4 y);
03173 char8 const_func __attribute__((overloadable)) mul_hi(char8 x, char8 y);
03174 uchar8 const_func __attribute__((overloadable)) mul_hi(uchar8 x, uchar8 y);
03175 char16 const_func __attribute__((overloadable)) mul_hi(char16 x, char16 y);
03176 uchar16 const_func __attribute__((overloadable)) mul_hi(uchar16 x, uchar16 y);
03177 short const_func __attribute__((overloadable)) mul_hi(short x, short y);
03178 ushort const_func __attribute__((overloadable)) mul_hi(ushort x, ushort y);
03179 short2 const_func __attribute__((overloadable)) mul_hi(short2 x, short2 y);
03180 ushort2 const_func __attribute__((overloadable)) mul_hi(ushort2 x, ushort2 y);
03181 short3 const_func __attribute__((overloadable)) mul_hi(short3 x, short3 y);
03182 ushort3 const_func __attribute__((overloadable)) mul_hi(ushort3 x, ushort3 y);
03183 short4 const_func __attribute__((overloadable)) mul_hi(short4 x, short4 y);
03184 ushort4 const_func __attribute__((overloadable)) mul_hi(ushort4 x, ushort4 y);
```

```
03185 short8 const_func __attribute__((overloadable)) mul_hi(short8 x, short8 y);
03186 ushort8 const_func __attribute__((overloadable)) mul_hi(ushort8 x, ushort8 y);
03187 short16 const_func __attribute__((overloadable)) mul_hi(short16 x, short16 y);
03188 ushort16 const_func __attribute__((overloadable)) mul_hi(ushort16 x, ushort16 y);
03189 int const_func __attribute__((overloadable)) mul_hi(int x, int y);
03190 uint const_func __attribute__((overloadable)) mul_hi(uint x, uint y);
03191 int2 const_func __attribute__((overloadable)) mul_hi(int2 x, int2 y);
03192 uint2 const_func __attribute__((overloadable)) mul_hi(uint2 x, uint2 y);
03193 int3 const_func __attribute__((overloadable)) mul_hi(int3 x, int3 y);
03194 uint3 const_func __attribute__((overloadable)) mul_hi(uint3 x, uint3 y);
03195 int4 const_func __attribute__((overloadable)) mul_hi(int4 x, int4 y);
03196 uint4 const_func __attribute__((overloadable)) mul_hi(uint4 x, uint4 y);
03197 int8 const_func __attribute__((overloadable)) mul_hi(int8 x, int8 y);
03198 uint8 const_func __attribute__((overloadable)) mul_hi(uint8 x, uint8 y);
03199 int16 const_func __attribute__((overloadable)) mul_hi(int16 x, int16 y);
03200 uint16 const_func __attribute__((overloadable)) mul_hi(uint16 x, uint16 y);
03201 long const_func __attribute__((overloadable)) mul_hi(long x, long y);
03202 ulong const_func __attribute__((overloadable)) mul_hi(ulong x, ulong y);
03203 long2 const_func __attribute__((overloadable)) mul_hi(long2 x, long2 y);
03204 ulong2 const_func __attribute__((overloadable)) mul_hi(ulong2 x, ulong2 y);
03205 long3 const_func __attribute__((overloadable)) mul_hi(long3 x, long3 y);
03206 ulong3 const_func __attribute__((overloadable)) mul_hi(ulong3 x, ulong3 y);
03207 long4 const_func __attribute__((overloadable)) mul_hi(long4 x, long4 y);
03208 ulong4 const_func __attribute__((overloadable)) mul_hi(ulong4 x, ulong4 y);
03209 long8 const_func __attribute__((overloadable)) mul_hi(long8 x, long8 y);
03210 ulong8 const_func __attribute__((overloadable)) mul_hi(ulong8 x, ulong8 y);
03211 long16 const_func __attribute__((overloadable)) mul_hi(long16 x, long16 y);
03212 ulong16 const_func __attribute__((overloadable)) mul_hi(ulong16 x, ulong16 y);
03213
03214 /**
03215 * For each element in v, the bits are shifted left by
03216 * the number of bits given by the corresponding
03217 * element in i (subject to usual shift modulo rules
03218 * described in section 6.3).  Bits shifted off the left
03219 * side of the element are shifted back in from the
03220 * right.
03221 */
03222 char const_func __attribute__((overloadable)) rotate(char v, char i);
03223 uchar const_func __attribute__((overloadable)) rotate(uchar v, uchar i);
03224 char2 const_func __attribute__((overloadable)) rotate(char2 v, char2 i);
03225 uchar2 const_func __attribute__((overloadable)) rotate(uchar2 v, uchar2 i);
03226 char3 const_func __attribute__((overloadable)) rotate(char3 v, char3 i);
03227 uchar3 const_func __attribute__((overloadable)) rotate(uchar3 v, uchar3 i);
03228 char4 const_func __attribute__((overloadable)) rotate(char4 v, char4 i);
03229 uchar4 const_func __attribute__((overloadable)) rotate(uchar4 v, uchar4 i);
03230 char8 const_func __attribute__((overloadable)) rotate(char8 v, char8 i);
03231 uchar8 const_func __attribute__((overloadable)) rotate(uchar8 v, uchar8 i);
03232 char16 const_func __attribute__((overloadable)) rotate(char16 v, char16 i);
03233 uchar16 const_func __attribute__((overloadable)) rotate(uchar16 v, uchar16 i);
03234 short const_func __attribute__((overloadable)) rotate(short v, short i);
03235 ushort const_func __attribute__((overloadable)) rotate(ushort v, ushort i);
03236 short2 const_func __attribute__((overloadable)) rotate(short2 v, short2 i);
03237 ushort2 const_func __attribute__((overloadable)) rotate(ushort2 v, ushort2 i);
03238 short3 const_func __attribute__((overloadable)) rotate(short3 v, short3 i);
03239 ushort3 const_func __attribute__((overloadable)) rotate(ushort3 v, ushort3 i);
03240 short4 const_func __attribute__((overloadable)) rotate(short4 v, short4 i);
03241 ushort4 const_func __attribute__((overloadable)) rotate(ushort4 v, ushort4 i);
03242 short8 const_func __attribute__((overloadable)) rotate(short8 v, short8 i);
03243 ushort8 const_func __attribute__((overloadable)) rotate(ushort8 v, ushort8 i);
03244 short16 const_func __attribute__((overloadable)) rotate(short16 v, short16 i);
03245 ushort16 const_func __attribute__((overloadable)) rotate(ushort16 v, ushort16 i);
03246 int const_func __attribute__((overloadable)) rotate(int v, int i);
03247 uint const_func __attribute__((overloadable)) rotate(uint v, uint i);
03248 int2 const_func __attribute__((overloadable)) rotate(int2 v, int2 i);
03249 uint2 const_func __attribute__((overloadable)) rotate(uint2 v, uint2 i);
03250 int3 const_func __attribute__((overloadable)) rotate(int3 v, int3 i);
03251 uint3 const_func __attribute__((overloadable)) rotate(uint3 v, uint3 i);
03252 int4 const_func __attribute__((overloadable)) rotate(int4 v, int4 i);
03253 uint4 const_func __attribute__((overloadable)) rotate(uint4 v, uint4 i);
03254 int8 const_func __attribute__((overloadable)) rotate(int8 v, int8 i);
03255 uint8 const_func __attribute__((overloadable)) rotate(uint8 v, uint8 i);
03256 int16 const_func __attribute__((overloadable)) rotate(int16 v, int16 i);
03257 uint16 const_func __attribute__((overloadable)) rotate(uint16 v, uint16 i);
03258 long const_func __attribute__((overloadable)) rotate(long v, long i);
03259 ulong const_func __attribute__((overloadable)) rotate(ulong v, ulong i);
03260 long2 const_func __attribute__((overloadable)) rotate(long2 v, long2 i);
03261 ulong2 const_func __attribute__((overloadable)) rotate(ulong2 v, ulong2 i);
03262 long3 const_func __attribute__((overloadable)) rotate(long3 v, long3 i);
03263 ulong3 const_func __attribute__((overloadable)) rotate(ulong3 v, ulong3 i);
03264 long4 const_func __attribute__((overloadable)) rotate(long4 v, long4 i);
03265 ulong4 const_func __attribute__((overloadable)) rotate(ulong4 v, ulong4 i);
03266 long8 const_func __attribute__((overloadable)) rotate(long8 v, long8 i);
03267 ulong8 const_func __attribute__((overloadable)) rotate(ulong8 v, ulong8 i);
03268 long16 const_func __attribute__((overloadable)) rotate(long16 v, long16 i);
03269 ulong16 const_func __attribute__((overloadable)) rotate(ulong16 v, ulong16 i);
03270
03271 /**
```

```
03272 * Returns x - y and saturates the result.
03273 */
03274 char const_func __attribute__((overloadable)) sub_sat(char x, char y);
03275 uchar const_func __attribute__((overloadable)) sub_sat(uchar x, uchar y);
03276 char2 const_func __attribute__((overloadable)) sub_sat(char2 x, char2 y);
03277 uchar2 const_func __attribute__((overloadable)) sub_sat(uchar2 x, uchar2 y);
03278 char3 const_func __attribute__((overloadable)) sub_sat(char3 x, char3 y);
03279 uchar3 const_func __attribute__((overloadable)) sub_sat(uchar3 x, uchar3 y);
03280 char4 const_func __attribute__((overloadable)) sub_sat(char4 x, char4 y);
03281 uchar4 const_func __attribute__((overloadable)) sub_sat(uchar4 x, uchar4 y);
03282 char8 const_func __attribute__((overloadable)) sub_sat(char8 x, char8 y);
03283 uchar8 const_func __attribute__((overloadable)) sub_sat(uchar8 x, uchar8 y);
03284 char16 const_func __attribute__((overloadable)) sub_sat(char16 x, char16 y);
03285 uchar16 const_func __attribute__((overloadable)) sub_sat(uchar16 x, uchar16 y);
03286 short const_func __attribute__((overloadable)) sub_sat(short x, short y);
03287 ushort const_func __attribute__((overloadable)) sub_sat(ushort x, ushort y);
03288 short2 const_func __attribute__((overloadable)) sub_sat(short2 x, short2 y);
03289 ushort2 const_func __attribute__((overloadable)) sub_sat(ushort2 x, ushort2 y);
03290 short3 const_func __attribute__((overloadable)) sub_sat(short3 x, short3 y);
03291 ushort3 const_func __attribute__((overloadable)) sub_sat(ushort3 x, ushort3 y);
03292 short4 const_func __attribute__((overloadable)) sub_sat(short4 x, short4 y);
03293 ushort4 const_func __attribute__((overloadable)) sub_sat(ushort4 x, ushort4 y);
03294 short8 const_func __attribute__((overloadable)) sub_sat(short8 x, short8 y);
03295 ushort8 const_func __attribute__((overloadable)) sub_sat(ushort8 x, ushort8 y);
03296 short16 const_func __attribute__((overloadable)) sub_sat(short16 x, short16 y);
03297 ushort16 const_func __attribute__((overloadable)) sub_sat(ushort16 x, ushort16 y);
03298 int const_func __attribute__((overloadable)) sub_sat(int x, int y);
03299 uint const_func __attribute__((overloadable)) sub_sat(uint x, uint y);
03300 int2 const_func __attribute__((overloadable)) sub_sat(int2 x, int2 y);
03301 uint2 const_func __attribute__((overloadable)) sub_sat(uint2 x, uint2 y);
03302 int3 const_func __attribute__((overloadable)) sub_sat(int3 x, int3 y);
03303 uint3 const_func __attribute__((overloadable)) sub_sat(uint3 x, uint3 y);
03304 int4 const_func __attribute__((overloadable)) sub_sat(int4 x, int4 y);
03305 uint4 const_func __attribute__((overloadable)) sub_sat(uint4 x, uint4 y);
03306 int8 const_func __attribute__((overloadable)) sub_sat(int8 x, int8 y);
03307 uint8 const_func __attribute__((overloadable)) sub_sat(uint8 x, uint8 y);
03308 int16 const_func __attribute__((overloadable)) sub_sat(int16 x, int16 y);
03309 uint16 const_func __attribute__((overloadable)) sub_sat(uint16 x, uint16 y);
03310 long const_func __attribute__((overloadable)) sub_sat(long x, long y);
03311 ulong const_func __attribute__((overloadable)) sub_sat(ulong x, ulong y);
03312 long2 const_func __attribute__((overloadable)) sub_sat(long2 x, long2 y);
03313 ulong2 const_func __attribute__((overloadable)) sub_sat(ulong2 x, ulong2 y);
03314 long3 const_func __attribute__((overloadable)) sub_sat(long3 x, long3 y);
03315 ulong3 const_func __attribute__((overloadable)) sub_sat(ulong3 x, ulong3 y);
03316 long4 const_func __attribute__((overloadable)) sub_sat(long4 x, long4 y);
03317 ulong4 const_func __attribute__((overloadable)) sub_sat(ulong4 x, ulong4 y);
03318 long8 const_func __attribute__((overloadable)) sub_sat(long8 x, long8 y);
03319 ulong8 const_func __attribute__((overloadable)) sub_sat(ulong8 x, ulong8 y);
03320 long16 const_func __attribute__((overloadable)) sub_sat(long16 x, long16 y);
03321 ulong16 const_func __attribute__((overloadable)) sub_sat(ulong16 x, ulong16 y);
03322
03323 /**
03324 * result[i] = ((short)hi[i] « 8) | lo[i]
03325 * result[i] = ((ushort)hi[i] « 8) | lo[i]
03326 */
03327 short const_func __attribute__((overloadable)) upsample(char hi, uchar lo);
03328 ushort const_func __attribute__((overloadable)) upsample (uchar hi, uchar lo);
03329 short2 const_func __attribute__((overloadable)) upsample (char2 hi, uchar2 lo);
03330 short3 const_func __attribute__((overloadable)) upsample (char3 hi, uchar3 lo);
03331 short4 const_func __attribute__((overloadable)) upsample (char4 hi, uchar4 lo);
03332 short8 const_func __attribute__((overloadable)) upsample (char8 hi, uchar8 lo);
03333 short16 const_func __attribute__((overloadable)) upsample (char16 hi, uchar16 lo);
03334 ushort2 const_func __attribute__((overloadable)) upsample (uchar2 hi, uchar2 lo);
03335 ushort3 const_func __attribute__((overloadable)) upsample (uchar3 hi, uchar3 lo);
03336 ushort4 const_func __attribute__((overloadable)) upsample (uchar4 hi, uchar4 lo);
03337 ushort8 const_func __attribute__((overloadable)) upsample (uchar8 hi, uchar8 lo);
03338 ushort16 const_func __attribute__((overloadable)) upsample (uchar16 hi, uchar16 lo);
03339
03340 /**
03341 * result[i] = ((int)hi[i] « 16) | lo[i]
03342 * result[i] = ((uint)hi[i] « 16) | lo[i]
03343 */
03344 int const_func __attribute__((overloadable)) upsample(short hi, ushort lo);
03345 uint const_func __attribute__((overloadable)) upsample (ushort hi, ushort lo);
03346 int2 const_func __attribute__((overloadable)) upsample (short2 hi, ushort2 lo);
03347 int3 const_func __attribute__((overloadable)) upsample (short3 hi, ushort3 lo);
03348 int4 const_func __attribute__((overloadable)) upsample (short4 hi, ushort4 lo);
03349 int8 const_func __attribute__((overloadable)) upsample (short8 hi, ushort8 lo);
03350 int16 const_func __attribute__((overloadable)) upsample (short16 hi, ushort16 lo);
03351 uint2 const_func __attribute__((overloadable)) upsample (ushort2 hi, ushort2 lo);
03352 uint3 const_func __attribute__((overloadable)) upsample (ushort3 hi, ushort3 lo);
03353 uint4 const_func __attribute__((overloadable)) upsample (ushort4 hi, ushort4 lo);
03354 uint8 const_func __attribute__((overloadable)) upsample (ushort8 hi, ushort8 lo);
03355 uint16 const_func __attribute__((overloadable)) upsample (ushort16 hi, ushort16 lo);
03356 /**
03357 * result[i] = ((long)hi[i] « 32) | lo[i]
03358 * result[i] = ((ulong)hi[i] « 32) | lo[i]
```

```
03359 */
03360 long const_func __attribute__((overloadable)) upsample(int hi, uint lo);
03361 ulong const_func __attribute__((overloadable)) upsample (uint hi, uint lo);
03362 long2 const_func __attribute__((overloadable)) upsample (int2 hi, uint2 lo);
03363 long3 const_func __attribute__((overloadable)) upsample (int3 hi, uint3 lo);
03364 long4 const_func __attribute__((overloadable)) upsample (int4 hi, uint4 lo);
03365 long8 const_func __attribute__((overloadable)) upsample (int8 hi, uint8 lo);
03366 long16 const_func __attribute__((overloadable)) upsample (int16 hi, uint16 lo);
03367 ulong2 const_func __attribute__((overloadable)) upsample (uint2 hi, uint2 lo);
03368 ulong3 const_func __attribute__((overloadable)) upsample (uint3 hi, uint3 lo);
03369 ulong4 const_func __attribute__((overloadable)) upsample (uint4 hi, uint4 lo);
03370 ulong8 const_func __attribute__((overloadable)) upsample (uint8 hi, uint8 lo);
03371 ulong16 const_func __attribute__((overloadable)) upsample (uint16 hi, uint16 lo);
03372
03373 /*
03374 * popcount(x):  returns the number of set bit in x
03375 */
03376 char const_func __attribute__((overloadable)) popcount(char x);
03377 uchar const_func __attribute__((overloadable)) popcount(uchar x);
03378 char2 const_func __attribute__((overloadable)) popcount(char2 x);
03379 uchar2 const_func __attribute__((overloadable)) popcount(uchar2 x);
03380 char3 const_func __attribute__((overloadable)) popcount(char3 x);
03381 uchar3 const_func __attribute__((overloadable)) popcount(uchar3 x);
03382 char4 const_func __attribute__((overloadable)) popcount(char4 x);
03383 uchar4 const_func __attribute__((overloadable)) popcount(uchar4 x);
03384 char8 const_func __attribute__((overloadable)) popcount(char8 x);
03385 uchar8 const_func __attribute__((overloadable)) popcount(uchar8 x);
03386 char16 const_func __attribute__((overloadable)) popcount(char16 x);
03387 uchar16 const_func __attribute__((overloadable)) popcount(uchar16 x);
03388 short const_func __attribute__((overloadable)) popcount(short x);
03389 ushort const_func __attribute__((overloadable)) popcount(ushort x);
03390 short2 const_func __attribute__((overloadable)) popcount(short2 x);
03391 ushort2 const_func __attribute__((overloadable)) popcount(ushort2 x);
03392 short3 const_func __attribute__((overloadable)) popcount(short3 x);
03393 ushort3 const_func __attribute__((overloadable)) popcount(ushort3 x);
03394 short4 const_func __attribute__((overloadable)) popcount(short4 x);
03395 ushort4 const_func __attribute__((overloadable)) popcount(ushort4 x);
03396 short8 const_func __attribute__((overloadable)) popcount(short8 x);
03397 ushort8 const_func __attribute__((overloadable)) popcount(ushort8 x);
03398 short16 const_func __attribute__((overloadable)) popcount(short16 x);
03399 ushort16 const_func __attribute__((overloadable)) popcount(ushort16 x);
03400 int const_func __attribute__((overloadable)) popcount(int x);
03401 uint const_func __attribute__((overloadable)) popcount(uint x);
03402 int2 const_func __attribute__((overloadable)) popcount(int2 x);
03403 uint2 const_func __attribute__((overloadable)) popcount(uint2 x);
03404 int3 const_func __attribute__((overloadable)) popcount(int3 x);
03405 uint3 const_func __attribute__((overloadable)) popcount(uint3 x);
03406 int4 const_func __attribute__((overloadable)) popcount(int4 x);
03407 uint4 const_func __attribute__((overloadable)) popcount(uint4 x);
03408 int8 const_func __attribute__((overloadable)) popcount(int8 x);
03409 uint8 const_func __attribute__((overloadable)) popcount(uint8 x);
03410 int16 const_func __attribute__((overloadable)) popcount(int16 x);
03411 uint16 const_func __attribute__((overloadable)) popcount(uint16 x);
03412 long const_func __attribute__((overloadable)) popcount(long x);
03413 ulong const_func __attribute__((overloadable)) popcount(ulong x);
03414 long2 const_func __attribute__((overloadable)) popcount(long2 x);
03415 ulong2 const_func __attribute__((overloadable)) popcount(ulong2 x);
03416 long3 const_func __attribute__((overloadable)) popcount(long3 x);
03417 ulong3 const_func __attribute__((overloadable)) popcount(ulong3 x);
03418 long4 const_func __attribute__((overloadable)) popcount(long4 x);
03419 ulong4 const_func __attribute__((overloadable)) popcount(ulong4 x);
03420 long8 const_func __attribute__((overloadable)) popcount(long8 x);
03421 ulong8 const_func __attribute__((overloadable)) popcount(ulong8 x);
03422 long16 const_func __attribute__((overloadable)) popcount(long16 x);
03423 ulong16 const_func __attribute__((overloadable)) popcount(ulong16 x);
03424
03425 /**
03426 * Multiply two 24-bit integer values x and y and add
03427 * the 32-bit integer result to the 32-bit integer z.
03428 * Refer to definition of mul24 to see how the 24-bit
03429 * integer multiplication is performed.
03430 */
03431 int const_func __attribute__((overloadable)) mad24(int x, int y, int z);
03432 uint const_func __attribute__((overloadable)) mad24(uint x, uint y, uint z);
03433 int2 const_func __attribute__((overloadable)) mad24(int2 x, int2 y, int2 z);
03434 uint2 const_func __attribute__((overloadable)) mad24(uint2 x, uint2 y, uint2 z);
03435 int3 const_func __attribute__((overloadable)) mad24(int3 x, int3 y, int3 z);
03436 uint3 const_func __attribute__((overloadable)) mad24(uint3 x, uint3 y, uint3 z);
03437 int4 const_func __attribute__((overloadable)) mad24(int4 x, int4 y, int4 z);
03438 uint4 const_func __attribute__((overloadable)) mad24(uint4 x, uint4 y, uint4 z);
03439 int8 const_func __attribute__((overloadable)) mad24(int8 x, int8 y, int8 z);
03440 uint8 const_func __attribute__((overloadable)) mad24(uint8 x, uint8 y, uint8 z);
03441 int16 const_func __attribute__((overloadable)) mad24(int16 x, int16 y, int16 z);
03442 uint16 const_func __attribute__((overloadable)) mad24(uint16 x, uint16 y, uint16 z);
03443
03444 /**
03445 * Multiply two 24-bit integer values x and y.  x and y
```

```
03446 * are 32-bit integers but only the low 24-bits are used
03447 * to perform the multiplication.  mul24 should only
03448 * be used when values in x and y are in the range [-
03449 * 2^23, 2^23-1] if x and y are signed integers and in the
03450 * range [0, 2^24-1] if x and y are unsigned integers.  If
03451 * x and y are not in this range, the multiplication
03452 * result is implementation-defined.
03453 */
03454 int const_func __attribute__((overloadable)) mul24(int x, int y);
03455 uint const_func __attribute__((overloadable)) mul24(uint x, uint y);
03456 int2 const_func __attribute__((overloadable)) mul24(int2 x, int2 y);
03457 uint2 const_func __attribute__((overloadable)) mul24(uint2 x, uint2 y);
03458 int3 const_func __attribute__((overloadable)) mul24(int3 x, int3 y);
03459 uint3 const_func __attribute__((overloadable)) mul24(uint3 x, uint3 y);
03460 int4 const_func __attribute__((overloadable)) mul24(int4 x, int4 y);
03461 uint4 const_func __attribute__((overloadable)) mul24(uint4 x, uint4 y);
03462 int8 const_func __attribute__((overloadable)) mul24(int8 x, int8 y);
03463 uint8 const_func __attribute__((overloadable)) mul24(uint8 x, uint8 y);
03464 int16 const_func __attribute__((overloadable)) mul24(int16 x, int16 y);
03465 uint16 const_func __attribute__((overloadable)) mul24(uint16 x, uint16 y);
03466
03467 // Common functions:
03468
03469 /**
03470 * Returns fmin(fmax(x, minval), maxval).
03471 * Results are undefined if minval > maxval.
03472 */
03473 float const_func __attribute__((overloadable)) clamp(float x, float minval, float maxval);
03474 float2 const_func __attribute__((overloadable)) clamp(float2 x, float2 minval, float2 maxval);
03475 float3 const_func __attribute__((overloadable)) clamp(float3 x, float3 minval, float3 maxval);
03476 float4 const_func __attribute__((overloadable)) clamp(float4 x, float4 minval, float4 maxval);
03477 float8 const_func __attribute__((overloadable)) clamp(float8 x, float8 minval, float8 maxval);
03478 float16 const_func __attribute__((overloadable)) clamp(float16 x, float16 minval, float16 maxval);
03479 float2 const_func __attribute__((overloadable)) clamp(float2 x, float minval, float maxval);
03480 float3 const_func __attribute__((overloadable)) clamp(float3 x, float minval, float maxval);
03481 float4 const_func __attribute__((overloadable)) clamp(float4 x, float minval, float maxval);
03482 float8 const_func __attribute__((overloadable)) clamp(float8 x, float minval, float maxval);
03483 float16 const_func __attribute__((overloadable)) clamp(float16 x, float minval, float maxval);
03484 double const_func __attribute__((overloadable)) clamp(double x, double minval, double maxval);
03485 double2 const_func __attribute__((overloadable)) clamp(double2 x, double2 minval, double2 maxval);
03486 double3 const_func __attribute__((overloadable)) clamp(double3 x, double3 minval, double3 maxval);
03487 double4 const_func __attribute__((overloadable)) clamp(double4 x, double4 minval, double4 maxval);
03488 double8 const_func __attribute__((overloadable)) clamp(double8 x, double8 minval, double8 maxval);
03489 double16 const_func __attribute__((overloadable)) clamp(double16 x, double16 minval, double16 maxval);
03490 double2 const_func __attribute__((overloadable)) clamp(double2 x, double minval, double maxval);
03491 double3 const_func __attribute__((overloadable)) clamp(double3 x, double minval, double maxval);
03492 double4 const_func __attribute__((overloadable)) clamp(double4 x, double minval, double maxval);
03493 double8 const_func __attribute__((overloadable)) clamp(double8 x, double minval, double maxval);
03494 double16 const_func __attribute__((overloadable)) clamp(double16 x, double minval, double maxval);
03495 //half const_func __attribute__((overloadable)) clamp(half x, half minval, half maxval);
03496 //half2 const_func __attribute__((overloadable)) clamp(half2 x, half2 minval, half2 maxval);
03497 //half3 const_func __attribute__((overloadable)) clamp(half3 x, half3 minval, half3 maxval);
03498 //half4 const_func __attribute__((overloadable)) clamp(half4 x, half4 minval, half4 maxval);
03499 //half8 const_func __attribute__((overloadable)) clamp(half8 x, half8 minval, half8 maxval);
03500 //half16 const_func __attribute__((overloadable)) clamp(half16 x, half16 minval, half16 maxval);
03501 //half2 const_func __attribute__((overloadable)) clamp(half2 x, half minval, half maxval);
03502 //half3 const_func __attribute__((overloadable)) clamp(half3 x, half minval, half maxval);
03503 //half4 const_func __attribute__((overloadable)) clamp(half4 x, half minval, half maxval);
03504 //half8 const_func __attribute__((overloadable)) clamp(half8 x, half minval, half maxval);
03505 //half16 const_func __attribute__((overloadable)) clamp(half16 x, half minval, half maxval);
03506
03507 /**
03508 * Converts radians to degrees, i.e.  (180 / PI) *
03509 * radians.
03510 */
03511 float const_func __attribute__((overloadable)) degrees(float radians);
03512 float2 const_func __attribute__((overloadable)) degrees(float2 radians);
03513 float3 const_func __attribute__((overloadable)) degrees(float3 radians);
03514 float4 const_func __attribute__((overloadable)) degrees(float4 radians);
03515 float8 const_func __attribute__((overloadable)) degrees(float8 radians);
03516 float16 const_func __attribute__((overloadable)) degrees(float16 radians);
03517 double const_func __attribute__((overloadable)) degrees(double radians);
03518 double2 const_func __attribute__((overloadable)) degrees(double2 radians);
03519 double3 const_func __attribute__((overloadable)) degrees(double3 radians);
03520 double4 const_func __attribute__((overloadable)) degrees(double4 radians);
03521 double8 const_func __attribute__((overloadable)) degrees(double8 radians);
03522 double16 const_func __attribute__((overloadable)) degrees(double16 radians);
03523 //half const_func __attribute__((overloadable)) degrees(half radians);
03524 //half2 const_func __attribute__((overloadable)) degrees(half2 radians);
03525 //half3 const_func __attribute__((overloadable)) degrees(half3 radians);
03526 //half4 const_func __attribute__((overloadable)) degrees(half4 radians);
03527 //half8 const_func __attribute__((overloadable)) degrees(half8 radians);
03528 //half16 const_func __attribute__((overloadable)) degrees(half16 radians);
03529
03530 /**
03531 * Returns y if x < y, otherwise it returns x.  If x and y
03532 * are infinite or NaN, the return values are undefined.
```

```
03533 */
03534 float const_func __attribute__((overloadable)) max(float x, float y);
03535 float2 const_func __attribute__((overloadable)) max(float2 x, float2 y);
03536 float3 const_func __attribute__((overloadable)) max(float3 x, float3 y);
03537 float4 const_func __attribute__((overloadable)) max(float4 x, float4 y);
03538 float8 const_func __attribute__((overloadable)) max(float8 x, float8 y);
03539 float16 const_func __attribute__((overloadable)) max(float16 x, float16 y);
03540 float2 const_func __attribute__((overloadable)) max(float2 x, float y);
03541 float3 const_func __attribute__((overloadable)) max(float3 x, float y);
03542 float4 const_func __attribute__((overloadable)) max(float4 x, float y);
03543 float8 const_func __attribute__((overloadable)) max(float8 x, float y);
03544 float16 const_func __attribute__((overloadable)) max(float16 x, float y);
03545 double const_func __attribute__((overloadable)) max(double x, double y);
03546 double2 const_func __attribute__((overloadable)) max(double2 x, double2 y);
03547 double3 const_func __attribute__((overloadable)) max(double3 x, double3 y);
03548 double4 const_func __attribute__((overloadable)) max(double4 x, double4 y);
03549 double8 const_func __attribute__((overloadable)) max(double8 x, double8 y);
03550 double16 const_func __attribute__((overloadable)) max(double16 x, double16 y);
03551 double2 const_func __attribute__((overloadable)) max(double2 x, double y);
03552 double3 const_func __attribute__((overloadable)) max(double3 x, double y);
03553 double4 const_func __attribute__((overloadable)) max(double4 x, double y);
03554 double8 const_func __attribute__((overloadable)) max(double8 x, double y);
03555 double16 const_func __attribute__((overloadable)) max(double16 x, double y);
03556 //half const_func __attribute__((overloadable)) max(half x, half y);
03557 //half2 const_func __attribute__((overloadable)) max(half2 x, half2 y);
03558 //half3 const_func __attribute__((overloadable)) max(half3 x, half3 y);
03559 //half4 const_func __attribute__((overloadable)) max(half4 x, half4 y);
03560 //half8 const_func __attribute__((overloadable)) max(half8 x, half8 y);
03561 //half16 const_func __attribute__((overloadable)) max(half16 x, half16 y);
03562 //half2 const_func __attribute__((overloadable)) max(half2 x, half y);
03563 //half3 const_func __attribute__((overloadable)) max(half3 x, half y);
03564 //half4 const_func __attribute__((overloadable)) max(half4 x, half y);
03565 //half8 const_func __attribute__((overloadable)) max(half8 x, half y);
03566 //half16 const_func __attribute__((overloadable)) max(half16 x, half y);
03567
03568 /**
03569 * Returns y if y < x, otherwise it returns x.  If x and y
03570 * are infinite or NaN, the return values are undefined.
03571 */
03572 float const_func __attribute__((overloadable)) min(float x, float y);
03573 float2 const_func __attribute__((overloadable)) min(float2 x, float2 y);
03574 float3 const_func __attribute__((overloadable)) min(float3 x, float3 y);
03575 float4 const_func __attribute__((overloadable)) min(float4 x, float4 y);
03576 float8 const_func __attribute__((overloadable)) min(float8 x, float8 y);
03577 float16 const_func __attribute__((overloadable)) min(float16 x, float16 y);
03578 float2 const_func __attribute__((overloadable)) min(float2 x, float y);
03579 float3 const_func __attribute__((overloadable)) min(float3 x, float y);
03580 float4 const_func __attribute__((overloadable)) min(float4 x, float y);
03581 float8 const_func __attribute__((overloadable)) min(float8 x, float y);
03582 float16 const_func __attribute__((overloadable)) min(float16 x, float y);
03583 double const_func __attribute__((overloadable)) min(double x, double y);
03584 double2 const_func __attribute__((overloadable)) min(double2 x, double2 y);
03585 double3 const_func __attribute__((overloadable)) min(double3 x, double3 y);
03586 double4 const_func __attribute__((overloadable)) min(double4 x, double4 y);
03587 double8 const_func __attribute__((overloadable)) min(double8 x, double8 y);
03588 double16 const_func __attribute__((overloadable)) min(double16 x, double16 y);
03589 double2 const_func __attribute__((overloadable)) min(double2 x, double y);
03590 double3 const_func __attribute__((overloadable)) min(double3 x, double y);
03591 double4 const_func __attribute__((overloadable)) min(double4 x, double y);
03592 double8 const_func __attribute__((overloadable)) min(double8 x, double y);
03593 double16 const_func __attribute__((overloadable)) min(double16 x, double y);
03594 //half const_func __attribute__((overloadable)) min(half x, half y);
03595 //half2 const_func __attribute__((overloadable)) min(half2 x, half2 y);
03596 //half3 const_func __attribute__((overloadable)) min(half3 x, half3 y);
03597 //half4 const_func __attribute__((overloadable)) min(half4 x, half4 y);
03598 //half8 const_func __attribute__((overloadable)) min(half8 x, half8 y);
03599 //half16 const_func __attribute__((overloadable)) min(half16 x, half16 y);
03600 //half2 const_func __attribute__((overloadable)) min(half2 x, half y);
03601 //half3 const_func __attribute__((overloadable)) min(half3 x, half y);
03602 //half4 const_func __attribute__((overloadable)) min(half4 x, half y);
03603 //half8 const_func __attribute__((overloadable)) min(half8 x, half y);
03604 //half16 const_func __attribute__((overloadable)) min(half16 x, half y);
03605
03606 /**
03607 * Returns the linear blend of x & y implemented as:
03608 * x + (y - x) * a
03609 * a must be a value in the range 0.0 ...  1.0.  If a is not
03610 * in the range 0.0 ...  1.0, the return values are
03611 * undefined.
03612 */
03613 float const_func __attribute__((overloadable)) mix(float x, float y, float a);
03614 float2 const_func __attribute__((overloadable)) mix(float2 x, float2 y, float2 a);
03615 float3 const_func __attribute__((overloadable)) mix(float3 x, float3 y, float3 a);
03616 float4 const_func __attribute__((overloadable)) mix(float4 x, float4 y, float4 a);
03617 float8 const_func __attribute__((overloadable)) mix(float8 x, float8 y, float8 a);
03618 float16 const_func __attribute__((overloadable)) mix(float16 x, float16 y, float16 a);
03619 float2 const_func __attribute__((overloadable)) mix(float2 x, float2 y, float a);
```

```
03620 float3 const_func __attribute__((overloadable)) mix(float3 x, float3 y, float a);
03621 float4 const_func __attribute__((overloadable)) mix(float4 x, float4 y, float a);
03622 float8 const_func __attribute__((overloadable)) mix(float8 x, float8 y, float a);
03623 float16 const_func __attribute__((overloadable)) mix(float16 x, float16 y, float a);
03624 double const_func __attribute__((overloadable)) mix(double x, double y, double a);
03625 double2 const_func __attribute__((overloadable)) mix(double2 x, double2 y, double2 a);
03626 double3 const_func __attribute__((overloadable)) mix(double3 x, double3 y, double3 a);
03627 double4 const_func __attribute__((overloadable)) mix(double4 x, double4 y, double4 a);
03628 double8 const_func __attribute__((overloadable)) mix(double8 x, double8 y, double8 a);
03629 double16 const_func __attribute__((overloadable)) mix(double16 x, double16 y, double16 a);
03630 double2 const_func __attribute__((overloadable)) mix(double2 x, double2 y, double a);
03631 double3 const_func __attribute__((overloadable)) mix(double3 x, double3 y, double a);
03632 double4 const_func __attribute__((overloadable)) mix(double4 x, double4 y, double a);
03633 double8 const_func __attribute__((overloadable)) mix(double8 x, double8 y, double a);
03634 double16 const_func __attribute__((overloadable)) mix(double16 x, double16 y, double a);
03635 //half const_func __attribute__((overloadable)) mix(half x, half y, half a);
03636 //half2 const_func __attribute__((overloadable)) mix(half2 x, half2 y, half2 a);
03637 //half3 const_func __attribute__((overloadable)) mix(half3 x, half3 y, half3 a);
03638 //half4 const_func __attribute__((overloadable)) mix(half4 x, half4 y, half4 a);
03639 //half8 const_func __attribute__((overloadable)) mix(half8 x, half8 y, half8 a);
03640 //half16 const_func __attribute__((overloadable)) mix(half16 x, half16 y, half16 a);
03641 //half2 const_func __attribute__((overloadable)) mix(half2 x, half2 y, half a);
03642 //half3 const_func __attribute__((overloadable)) mix(half3 x, half3 y, half a);
03643 //half4 const_func __attribute__((overloadable)) mix(half4 x, half4 y, half a);
03644 //half8 const_func __attribute__((overloadable)) mix(half8 x, half8 y, half a);
03645 //half16 const_func __attribute__((overloadable)) mix(half16 x, half16 y, half a);
03646
03647 /**
03648 * Converts degrees to radians, i.e.  (PI / 180) *
03649 * degrees.
03650 */
03651 float const_func __attribute__((overloadable)) radians(float degrees);
03652 float2 const_func __attribute__((overloadable)) radians(float2 degrees);
03653 float3 const_func __attribute__((overloadable)) radians(float3 degrees);
03654 float4 const_func __attribute__((overloadable)) radians(float4 degrees);
03655 float8 const_func __attribute__((overloadable)) radians(float8 degrees);
03656 float16 const_func __attribute__((overloadable)) radians(float16 degrees);
03657 double const_func __attribute__((overloadable)) radians(double degrees);
03658 double2 const_func __attribute__((overloadable)) radians(double2 degrees);
03659 double3 const_func __attribute__((overloadable)) radians(double3 degrees);
03660 double4 const_func __attribute__((overloadable)) radians(double4 degrees);
03661 double8 const_func __attribute__((overloadable)) radians(double8 degrees);
03662 double16 const_func __attribute__((overloadable)) radians(double16 degrees);
03663 //half const_func __attribute__((overloadable)) radians(half degrees);
03664 //half2 const_func __attribute__((overloadable)) radians(half2 degrees);
03665 //half3 const_func __attribute__((overloadable)) radians(half3 degrees);
03666 //half4 const_func __attribute__((overloadable)) radians(half4 degrees);
03667 //half8 const_func __attribute__((overloadable)) radians(half8 degrees);
03668 //half16 const_func __attribute__((overloadable)) radians(half16 degrees);
03669
03670 /**
03671 * Returns 0.0 if x < edge, otherwise it returns 1.0.
03672 */
03673 float const_func __attribute__((overloadable)) step(float edge, float x);
03674 float2 const_func __attribute__((overloadable)) step(float2 edge, float2 x);
03675 float3 const_func __attribute__((overloadable)) step(float3 edge, float3 x);
03676 float4 const_func __attribute__((overloadable)) step(float4 edge, float4 x);
03677 float8 const_func __attribute__((overloadable)) step(float8 edge, float8 x);
03678 float16 const_func __attribute__((overloadable)) step(float16 edge, float16 x);
03679 float2 const_func __attribute__((overloadable)) step(float edge, float2 x);
03680 float3 const_func __attribute__((overloadable)) step(float edge, float3 x);
03681 float4 const_func __attribute__((overloadable)) step(float edge, float4 x);
03682 float8 const_func __attribute__((overloadable)) step(float edge, float8 x);
03683 float16 const_func __attribute__((overloadable)) step(float edge, float16 x);
03684 double const_func __attribute__((overloadable)) step(double edge, double x);
03685 double2 const_func __attribute__((overloadable)) step(double2 edge, double2 x);
03686 double3 const_func __attribute__((overloadable)) step(double3 edge, double3 x);
03687 double4 const_func __attribute__((overloadable)) step(double4 edge, double4 x);
03688 double8 const_func __attribute__((overloadable)) step(double8 edge, double8 x);
03689 double16 const_func __attribute__((overloadable)) step(double16 edge, double16 x);
03690 double const_func __attribute__((overloadable)) step(double edge, double x);
03691 double2 const_func __attribute__((overloadable)) step(double edge, double2 x);
03692 double3 const_func __attribute__((overloadable)) step(double edge, double3 x);
03693 double4 const_func __attribute__((overloadable)) step(double edge, double4 x);
03694 double8 const_func __attribute__((overloadable)) step(double edge, double8 x);
03695 double16 const_func __attribute__((overloadable)) step(double edge, double16 x);
03696 //half const_func __attribute__((overloadable)) step(half edge, half x);
03697 //half2 const_func __attribute__((overloadable)) step(half2 edge, half2 x);
03698 //half3 const_func __attribute__((overloadable)) step(half3 edge, half3 x);
03699 //half4 const_func __attribute__((overloadable)) step(half4 edge, half4 x);
03700 //half8 const_func __attribute__((overloadable)) step(half8 edge, half8 x);
03701 //half16 const_func __attribute__((overloadable)) step(half16 edge, half16 x);
03702 //half const_func __attribute__((overloadable)) step(half edge, half x);
03703 //half2 const_func __attribute__((overloadable)) step(half edge, half2 x);
03704 //half3 const_func __attribute__((overloadable)) step(half edge, half3 x);
03705 //half4 const_func __attribute__((overloadable)) step(half edge, half4 x);
03706 //half8 const_func __attribute__((overloadable)) step(half edge, half8 x);
```

```
03707 //half16 const_func __attribute__((overloadable)) step(half edge, half16 x);
03708
03709 /**
03710 * Returns 0.0 if x <= edge0 and 1.0 if x >= edge1 and
03711 * performs smooth Hermite interpolation between 0
03712 * and 1when edge0 < x < edge1.  This is useful in
03713 * cases where you would want a threshold function
03714 * with a smooth transition.
03715 * This is equivalent to:
03716 * gentype t;
03717 * t = clamp ((x - edge0) / (edge1 - edge0), 0, 1);
03718 * return t * t * (3 - 2 * t);
03719 * Results are undefined if edge0 >= edge1 or if x,
03720 * edge0 or edge1 is a NaN.
03721 */
03722 float const_func __attribute__((overloadable)) smoothstep(float edge0, float edge1, float x);
03723 float2 const_func __attribute__((overloadable)) smoothstep(float2 edge0, float2 edge1, float2 x);
03724 float3 const_func __attribute__((overloadable)) smoothstep(float3 edge0, float3 edge1, float3 x);
03725 float4 const_func __attribute__((overloadable)) smoothstep(float4 edge0, float4 edge1, float4 x);
03726 float8 const_func __attribute__((overloadable)) smoothstep(float8 edge0, float8 edge1, float8 x);
03727 float16 const_func __attribute__((overloadable)) smoothstep(float16 edge0, float16 edge1, float16 x);
03728 float2 const_func __attribute__((overloadable)) smoothstep(float edge0, float edge1, float2 x);
03729 float3 const_func __attribute__((overloadable)) smoothstep(float edge0, float edge1, float3 x);
03730 float4 const_func __attribute__((overloadable)) smoothstep(float edge0, float edge1, float4 x);
03731 float8 const_func __attribute__((overloadable)) smoothstep(float edge0, float edge1, float8 x);
03732 float16 const_func __attribute__((overloadable)) smoothstep(float edge0, float edge1, float16 x);
03733 double const_func __attribute__((overloadable)) smoothstep(double edge0, double edge1, double x);
03734 double2 const_func __attribute__((overloadable)) smoothstep(double edge0, double edge1, double2 x);
03735 double3 const_func __attribute__((overloadable)) smoothstep(double edge0, double edge1, double3 x);
03736 double4 const_func __attribute__((overloadable)) smoothstep(double edge0, double edge1, double4 x);
03737 double8 const_func __attribute__((overloadable)) smoothstep(double edge0, double edge1, double8 x);
03738 double16 const_func __attribute__((overloadable)) smoothstep(double edge0, double edge1, double16 x);
03739 double const_func __attribute__((overloadable)) smoothstep(double edge0, double edge1, double x);
03740 double2 const_func __attribute__((overloadable)) smoothstep(double2 edge0, double2 edge1, double2 x);
03741 double3 const_func __attribute__((overloadable)) smoothstep(double3 edge0, double3 edge1, double3 x);
03742 double4 const_func __attribute__((overloadable)) smoothstep(double4 edge0, double4 edge1, double4 x);
03743 double8 const_func __attribute__((overloadable)) smoothstep(double8 edge0, double8 edge1, double8 x);
03744 double16 const_func __attribute__((overloadable)) smoothstep(double16 edge0, double16 edge1, double16
      x);
03745 //half const_func __attribute__((overloadable)) smoothstep(half edge0, half edge1, half x);
03746 //half2 const_func __attribute__((overloadable)) smoothstep(half2 edge0, half2 edge1, half2 x);
03747 //half3 const_func __attribute__((overloadable)) smoothstep(half3 edge0, half3 edge1, half3 x);
03748 //half4 const_func __attribute__((overloadable)) smoothstep(half4 edge0, half4 edge1, half4 x);
03749 //half8 const_func __attribute__((overloadable)) smoothstep(half8 edge0, half8 edge1, half8 x);
03750 //half16 const_func __attribute__((overloadable)) smoothstep(half16 edge0, half16 edge1, half16 x);
03751 //half const_func __attribute__((overloadable)) smoothstep(half edge0, half edge1, half x);
03752 //half2 const_func __attribute__((overloadable)) smoothstep(half edge0, half edge1, half2 x);
03753 //half3 const_func __attribute__((overloadable)) smoothstep(half edge0, half edge1, half3 x);
03754 //half4 const_func __attribute__((overloadable)) smoothstep(half edge0, half edge1, half4 x);
03755 //half8 const_func __attribute__((overloadable)) smoothstep(half edge0, half edge1, half8 x);
03756 //half16 const_func __attribute__((overloadable)) smoothstep(half edge0, half edge1, half16 x);
03757
03758 /**
03759 * Returns 1.0 if x > 0, -0.0 if x = -0.0, +0.0 if x =
03760 * +0.0, or -1.0 if x < 0.  Returns 0.0 if x is a NaN.
03761 */
03762 float const_func __attribute__((overloadable)) sign(float x);
03763 float2 const_func __attribute__((overloadable)) sign(float2 x);
03764 float3 const_func __attribute__((overloadable)) sign(float3 x);
03765 float4 const_func __attribute__((overloadable)) sign(float4 x);
03766 float8 const_func __attribute__((overloadable)) sign(float8 x);
03767 float16 const_func __attribute__((overloadable)) sign(float16 x);
03768 double const_func __attribute__((overloadable)) sign(double x);
03769 double2 const_func __attribute__((overloadable)) sign(double2 x);
03770 double3 const_func __attribute__((overloadable)) sign(double3 x);
03771 double4 const_func __attribute__((overloadable)) sign(double4 x);
03772 double8 const_func __attribute__((overloadable)) sign(double8 x);
03773 double16 const_func __attribute__((overloadable)) sign(double16 x);
03774 //half const_func __attribute__((overloadable)) sign(half x);
03775 //half2 const_func __attribute__((overloadable)) sign(half2 x);
03776 //half3 const_func __attribute__((overloadable)) sign(half3 x);
03777 //half4 const_func __attribute__((overloadable)) sign(half4 x);
03778 //half8 const_func __attribute__((overloadable)) sign(half8 x);
03779 //half16 const_func __attribute__((overloadable)) sign(half16 x);
03780
03781 // Geometric functions:
03782
03783 /**
03784 * Returns the cross product of p0.xyz and p1.xyz.  The
03785 * w component of float4 result returned will be 0.0.
03786 */
03787 float4 const_func __attribute__((overloadable)) cross(float4 p0, float4 p1);
03788 float3 const_func __attribute__((overloadable)) cross (float3 p0, float3 p1);
03789 double4 const_func __attribute__((overloadable)) cross(double4 p0, double4 p1);
03790 double3 const_func __attribute__((overloadable)) cross(double3 p0, double3 p1);
03791 //half4 const_func __attribute__((overloadable)) cross(half4 p0, half4 p1);
03792 //half3 const_func __attribute__((overloadable)) cross(half3 p0, half3 p1);
```

```
03793
03794 /**
03795 * Compute dot product.
03796 */
03797 float const_func __attribute__((overloadable)) dot(float p0, float p1);
03798 float const_func __attribute__((overloadable)) dot(float2 p0, float2 p1);
03799 float const_func __attribute__((overloadable)) dot(float3 p0, float3 p1);
03800 float const_func __attribute__((overloadable)) dot(float4 p0, float4 p1);
03801 double const_func __attribute__((overloadable)) dot(double p0, double p1);
03802 double const_func __attribute__((overloadable)) dot(double2 p0, double2 p1);
03803 double const_func __attribute__((overloadable)) dot(double3 p0, double3 p1);
03804 double const_func __attribute__((overloadable)) dot(double4 p0, double4 p1);
03805 //half const_func __attribute__((overloadable)) dot(half p0, half p1);
03806 //half const_func __attribute__((overloadable)) dot(half2 p0, half2 p1);
03807 //half const_func __attribute__((overloadable)) dot(half3 p0, half3 p1);
03808 //half const_func __attribute__((overloadable)) dot(half4 p0, half4 p1);
03809
03810 /**
03811 * Returns the distance between p0 and p1.  This is
03812 * calculated as length(p0 - p1).
03813 */
03814 float const_func __attribute__((overloadable)) distance(float p0, float p1);
03815 float const_func __attribute__((overloadable)) distance(float2 p0, float2 p1);
03816 float const_func __attribute__((overloadable)) distance(float3 p0, float3 p1);
03817 float const_func __attribute__((overloadable)) distance(float4 p0, float4 p1);
03818 double const_func __attribute__((overloadable)) distance(double p0, double p1);
03819 double const_func __attribute__((overloadable)) distance(double2 p0, double2 p1);
03820 double const_func __attribute__((overloadable)) distance(double3 p0, double3 p1);
03821 double const_func __attribute__((overloadable)) distance(double4 p0, double4 p1);
03822 //half const_func __attribute__((overloadable)) distance(half p0, half p1);
03823 //half const_func __attribute__((overloadable)) distance(half2 p0, half2 p1);
03824 //half const_func __attribute__((overloadable)) distance(half3 p0, half3 p1);
03825 //half const_func __attribute__((overloadable)) distance(half4 p0, half4 p1);
03826
03827 /**
03828 * Return the length of vector p, i.e.,
03829 * sqrt(p.x2 + p.y 2 + ...)
03830 */
03831 float const_func __attribute__((overloadable)) length(float p);
03832 float const_func __attribute__((overloadable)) length(float2 p);
03833 float const_func __attribute__((overloadable)) length(float3 p);
03834 float const_func __attribute__((overloadable)) length(float4 p);
03835 double const_func __attribute__((overloadable)) length(double p);
03836 double const_func __attribute__((overloadable)) length(double2 p);
03837 double const_func __attribute__((overloadable)) length(double3 p);
03838 double const_func __attribute__((overloadable)) length(double4 p);
03839 //half const_func __attribute__((overloadable)) length(half p);
03840 //half const_func __attribute__((overloadable)) length(half2 p);
03841 //half const_func __attribute__((overloadable)) length(half3 p);
03842 //half const_func __attribute__((overloadable)) length(half4 p);
03843
03844 /**
03845 * Returns a vector in the same direction as p but with a
03846 * length of 1.
03847 */
03848 float const_func __attribute__((overloadable)) normalize(float p);
03849 float2 const_func __attribute__((overloadable)) normalize(float2 p);
03850 float3 const_func __attribute__((overloadable)) normalize(float3 p);
03851 float4 const_func __attribute__((overloadable)) normalize(float4 p);
03852 double const_func __attribute__((overloadable)) normalize(double p);
03853 double2 const_func __attribute__((overloadable)) normalize(double2 p);
03854 double3 const_func __attribute__((overloadable)) normalize(double3 p);
03855 double4 const_func __attribute__((overloadable)) normalize(double4 p);
03856 //half const_func __attribute__((overloadable)) normalize(half p);
03857 //half2 const_func __attribute__((overloadable)) normalize(half2 p);
03858 //half3 const_func __attribute__((overloadable)) normalize(half3 p);
03859 //half4 const_func __attribute__((overloadable)) normalize(half4 p);
03860
03861 /**
03862 * Returns fast_length(p0 - p1).
03863 */
03864 float const_func __attribute__((overloadable)) fast_distance(float p0, float p1);
03865 float const_func __attribute__((overloadable)) fast_distance(float2 p0, float2 p1);
03866 float const_func __attribute__((overloadable)) fast_distance(float3 p0, float3 p1);
03867 float const_func __attribute__((overloadable)) fast_distance(float4 p0, float4 p1);
03868
03869 /**
03870 * Returns the length of vector p computed as:
03871 * half_sqrt(p.x2 + p.y2 + ...)
03872 */
03873 float const_func __attribute__((overloadable)) fast_length(float p);
03874 float const_func __attribute__((overloadable)) fast_length(float2 p);
03875 float const_func __attribute__((overloadable)) fast_length(float3 p);
03876 float const_func __attribute__((overloadable)) fast_length(float4 p);
03877
03878 /**
03879 * Returns a vector in the same direction as p but with a
```

```
03880 * length of 1.  fast_normalize is computed as:
03881 * p * half_rsqrt (p.x^2 + p.y^2 + ...  )
03882 * The result shall be within 8192 ulps error from the
03883 * infinitely precise result of
03884 * if (all(p == 0.0f))
03885 * result = p;
03886 * else
03887 * result = p / sqrt (p.x^2 + p.y^2 + ...  );
03888 * with the following exceptions:
03889 * 1) If the sum of squares is greater than FLT_MAX
03890 * then the value of the floating-point values in the
03891 * result vector are undefined.
03892 * 2) If the sum of squares is less than FLT_MIN then
03893 * the implementation may return back p.
03894 * 3) If the device is in "denorms are flushed to zero"
03895 * mode, individual operand elements with magnitude
03896 * less than sqrt(FLT_MIN) may be flushed to zero
03897 * before proceeding with the calculation.
03898 */
03899 float const_func __attribute__((overloadable)) fast_normalize(float p);
03900 float2 const_func __attribute__((overloadable)) fast_normalize(float2 p);
03901 float3 const_func __attribute__((overloadable)) fast_normalize(float3 p);
03902 float4 const_func __attribute__((overloadable)) fast_normalize(float4 p);
03903
03904 // Relational functions:
03905
03906 /**
03907 * intn isequal (floatn x, floatn y)
03908 * Returns the component-wise compare of x == y.
03909 */
03910 int const_func __attribute__((overloadable)) isequal(float x, float y);
03911 int2 const_func __attribute__((overloadable)) isequal(float2 x, float2 y);
03912 int3 const_func __attribute__((overloadable)) isequal(float3 x, float3 y);
03913 int4 const_func __attribute__((overloadable)) isequal(float4 x, float4 y);
03914 int8 const_func __attribute__((overloadable)) isequal(float8 x, float8 y);
03915 int16 const_func __attribute__((overloadable)) isequal(float16 x, float16 y);
03916 int const_func __attribute__((overloadable)) isequal(double x, double y);
03917 long2 const_func __attribute__((overloadable)) isequal(double2 x, double2 y);
03918 long3 const_func __attribute__((overloadable)) isequal(double3 x, double3 y);
03919 long4 const_func __attribute__((overloadable)) isequal(double4 x, double4 y);
03920 long8 const_func __attribute__((overloadable)) isequal(double8 x, double8 y);
03921 long16 const_func __attribute__((overloadable)) isequal(double16 x, double16 y);
03922 //int const_func __attribute__((overloadable)) isequal(half x, half y);
03923 //short2 const_func __attribute__((overloadable)) isequal(half2 x, half2 y);
03924 //short3 const_func __attribute__((overloadable)) isequal(half3 x, half3 y);
03925 //short4 const_func __attribute__((overloadable)) isequal(half4 x, half4 y);
03926 //short8 const_func __attribute__((overloadable)) isequal(half8 x, half8 y);
03927 //short16 const_func __attribute__((overloadable)) isequal(half16 x, half16 y);
03928
03929 /**
03930 * Returns the component-wise compare of x != y.
03931 */
03932 int const_func __attribute__((overloadable)) isnotequal(float x, float y);
03933 int2 const_func __attribute__((overloadable)) isnotequal(float2 x, float2 y);
03934 int3 const_func __attribute__((overloadable)) isnotequal(float3 x, float3 y);
03935 int4 const_func __attribute__((overloadable)) isnotequal(float4 x, float4 y);
03936 int8 const_func __attribute__((overloadable)) isnotequal(float8 x, float8 y);
03937 int16 const_func __attribute__((overloadable)) isnotequal(float16 x, float16 y);
03938 int const_func __attribute__((overloadable)) isnotequal(double x, double y);
03939 long2 const_func __attribute__((overloadable)) isnotequal(double2 x, double2 y);
03940 long3 const_func __attribute__((overloadable)) isnotequal(double3 x, double3 y);
03941 long4 const_func __attribute__((overloadable)) isnotequal(double4 x, double4 y);
03942 long8 const_func __attribute__((overloadable)) isnotequal(double8 x, double8 y);
03943 long16 const_func __attribute__((overloadable)) isnotequal(double16 x, double16 y);
03944 //int const_func __attribute__((overloadable)) isnotequal(half x, half y);
03945 //short2 const_func __attribute__((overloadable)) isnotequal(half2 x, half2 y);
03946 //short3 const_func __attribute__((overloadable)) isnotequal(half3 x, half3 y);
03947 //short4 const_func __attribute__((overloadable)) isnotequal(half4 x, half4 y);
03948 //short8 const_func __attribute__((overloadable)) isnotequal(half8 x, half8 y);
03949 //short16 const_func __attribute__((overloadable)) isnotequal(half16 x, half16 y);
03950
03951 /**
03952 * Returns the component-wise compare of x > y.
03953 */
03954 int const_func __attribute__((overloadable)) isgreater(float x, float y);
03955 int2 const_func __attribute__((overloadable)) isgreater(float2 x, float2 y);
03956 int3 const_func __attribute__((overloadable)) isgreater(float3 x, float3 y);
03957 int4 const_func __attribute__((overloadable)) isgreater(float4 x, float4 y);
03958 int8 const_func __attribute__((overloadable)) isgreater(float8 x, float8 y);
03959 int16 const_func __attribute__((overloadable)) isgreater(float16 x, float16 y);
03960 int const_func __attribute__((overloadable)) isgreater(double x, double y);
03961 long2 const_func __attribute__((overloadable)) isgreater(double2 x, double2 y);
03962 long3 const_func __attribute__((overloadable)) isgreater(double3 x, double3 y);
03963 long4 const_func __attribute__((overloadable)) isgreater(double4 x, double4 y);
03964 long8 const_func __attribute__((overloadable)) isgreater(double8 x, double8 y);
03965 long16 const_func __attribute__((overloadable)) isgreater(double16 x, double16 y);
03966 //int const_func __attribute__((overloadable)) isgreater(half x, half y);
```

```
03967 //short2 const_func __attribute__((overloadable)) isgreater(half2 x, half2 y);
03968 //short3 const_func __attribute__((overloadable)) isgreater(half3 x, half3 y);
03969 //short4 const_func __attribute__((overloadable)) isgreater(half4 x, half4 y);
03970 //short8 const_func __attribute__((overloadable)) isgreater(half8 x, half8 y);
03971 //short16 const_func __attribute__((overloadable)) isgreater(half16 x, half16 y);
03972
03973 /**
03974 * Returns the component-wise compare of x >= y.
03975 */
03976 int const_func __attribute__((overloadable)) isgreaterequal(float x, float y);
03977 int2 const_func __attribute__((overloadable)) isgreaterequal(float2 x, float2 y);
03978 int3 const_func __attribute__((overloadable)) isgreaterequal(float3 x, float3 y);
03979 int4 const_func __attribute__((overloadable)) isgreaterequal(float4 x, float4 y);
03980 int8 const_func __attribute__((overloadable)) isgreaterequal(float8 x, float8 y);
03981 int16 const_func __attribute__((overloadable)) isgreaterequal(float16 x, float16 y);
03982 int const_func __attribute__((overloadable)) isgreaterequal(double x, double y);
03983 long2 const_func __attribute__((overloadable)) isgreaterequal(double2 x, double2 y);
03984 long3 const_func __attribute__((overloadable)) isgreaterequal(double3 x, double3 y);
03985 long4 const_func __attribute__((overloadable)) isgreaterequal(double4 x, double4 y);
03986 long8 const_func __attribute__((overloadable)) isgreaterequal(double8 x, double8 y);
03987 long16 const_func __attribute__((overloadable)) isgreaterequal(double16 x, double16 y);
03988 //int const_func __attribute__((overloadable)) isgreaterequal(half x, half y);
03989 //short2 const_func __attribute__((overloadable)) isgreaterequal(half2 x, half2 y);
03990 //short3 const_func __attribute__((overloadable)) isgreaterequal(half3 x, half3 y);
03991 //short4 const_func __attribute__((overloadable)) isgreaterequal(half4 x, half4 y);
03992 //short8 const_func __attribute__((overloadable)) isgreaterequal(half8 x, half8 y);
03993 //short16 const_func __attribute__((overloadable)) isgreaterequal(half16 x, half16 y);
03994
03995 /**
03996 * Returns the component-wise compare of x < y.
03997 */
03998 int const_func __attribute__((overloadable)) isless(float x, float y);
03999 int2 const_func __attribute__((overloadable)) isless(float2 x, float2 y);
04000 int3 const_func __attribute__((overloadable)) isless(float3 x, float3 y);
04001 int4 const_func __attribute__((overloadable)) isless(float4 x, float4 y);
04002 int8 const_func __attribute__((overloadable)) isless(float8 x, float8 y);
04003 int16 const_func __attribute__((overloadable)) isless(float16 x, float16 y);
04004 int const_func __attribute__((overloadable)) isless(double x, double y);
04005 long2 const_func __attribute__((overloadable)) isless(double2 x, double2 y);
04006 long3 const_func __attribute__((overloadable)) isless(double3 x, double3 y);
04007 long4 const_func __attribute__((overloadable)) isless(double4 x, double4 y);
04008 long8 const_func __attribute__((overloadable)) isless(double8 x, double8 y);
04009 long16 const_func __attribute__((overloadable)) isless(double16 x, double16 y);
04010 //int const_func __attribute__((overloadable)) isless(half x, half y);
04011 //short2 const_func __attribute__((overloadable)) isless(half2 x, half2 y);
04012 //short3 const_func __attribute__((overloadable)) isless(half3 x, half3 y);
04013 //short4 const_func __attribute__((overloadable)) isless(half4 x, half4 y);
04014 //short8 const_func __attribute__((overloadable)) isless(half8 x, half8 y);
04015 //short16 const_func __attribute__((overloadable)) isless(half16 x, half16 y);
04016
04017 /**
04018 * Returns the component-wise compare of x <= y.
04019 */
04020 int const_func __attribute__((overloadable)) islessequal(float x, float y);
04021 int2 const_func __attribute__((overloadable)) islessequal(float2 x, float2 y);
04022 int3 const_func __attribute__((overloadable)) islessequal(float3 x, float3 y);
04023 int4 const_func __attribute__((overloadable)) islessequal(float4 x, float4 y);
04024 int8 const_func __attribute__((overloadable)) islessequal(float8 x, float8 y);
04025 int16 const_func __attribute__((overloadable)) islessequal(float16 x, float16 y);
04026 int const_func __attribute__((overloadable)) islessequal(double x, double y);
04027 long2 const_func __attribute__((overloadable)) islessequal(double2 x, double2 y);
04028 long3 const_func __attribute__((overloadable)) islessequal(double3 x, double3 y);
04029 long4 const_func __attribute__((overloadable)) islessequal(double4 x, double4 y);
04030 long8 const_func __attribute__((overloadable)) islessequal(double8 x, double8 y);
04031 long16 const_func __attribute__((overloadable)) islessequal(double16 x, double16 y);
04032 //int const_func __attribute__((overloadable)) islessequal(half x, half y);
04033 //short2 const_func __attribute__((overloadable)) islessequal(half2 x, half2 y);
04034 //short3 const_func __attribute__((overloadable)) islessequal(half3 x, half3 y);
04035 //short4 const_func __attribute__((overloadable)) islessequal(half4 x, half4 y);
04036 //short8 const_func __attribute__((overloadable)) islessequal(half8 x, half8 y);
04037 //short16 const_func __attribute__((overloadable)) islessequal(half16 x, half16 y);
04038
04039 /**
04040 * Returns the component-wise compare of
04041 * (x < y) || (x > y) .
04042 */
04043 int const_func __attribute__((overloadable)) islessgreater(float x, float y);
04044 int2 const_func __attribute__((overloadable)) islessgreater(float2 x, float2 y);
04045 int3 const_func __attribute__((overloadable)) islessgreater(float3 x, float3 y);
04046 int4 const_func __attribute__((overloadable)) islessgreater(float4 x, float4 y);
04047 int8 const_func __attribute__((overloadable)) islessgreater(float8 x, float8 y);
04048 int16 const_func __attribute__((overloadable)) islessgreater(float16 x, float16 y);
04049 int const_func __attribute__((overloadable)) islessgreater(double x, double y);
04050 long2 const_func __attribute__((overloadable)) islessgreater(double2 x, double2 y);
04051 long3 const_func __attribute__((overloadable)) islessgreater(double3 x, double3 y);
04052 long4 const_func __attribute__((overloadable)) islessgreater(double4 x, double4 y);
04053 long8 const_func __attribute__((overloadable)) islessgreater(double8 x, double8 y);
```

```
04054 long16 const_func __attribute__((overloadable)) islessgreater(double16 x, double16 y);
04055 //int const_func __attribute__((overloadable)) islessgreater(half x, half y);
04056 //short2 const_func __attribute__((overloadable)) islessgreater(half2 x, half2 y);
04057 //short3 const_func __attribute__((overloadable)) islessgreater(half3 x, half3 y);
04058 //short4 const_func __attribute__((overloadable)) islessgreater(half4 x, half4 y);
04059 //short8 const_func __attribute__((overloadable)) islessgreater(half8 x, half8 y);
04060 //short16 const_func __attribute__((overloadable)) islessgreater(half16 x, half16 y);
04061
04062 /**
04063 * Test for finite value.
04064 */
04065 int const_func __attribute__((overloadable)) isfinite(float);
04066 int2 const_func __attribute__((overloadable)) isfinite(float2);
04067 int3 const_func __attribute__((overloadable)) isfinite(float3);
04068 int4 const_func __attribute__((overloadable)) isfinite(float4);
04069 int8 const_func __attribute__((overloadable)) isfinite(float8);
04070 int16 const_func __attribute__((overloadable)) isfinite(float16);
04071 int const_func __attribute__((overloadable)) isfinite(double);
04072 long2 const_func __attribute__((overloadable)) isfinite(double2);
04073 long3 const_func __attribute__((overloadable)) isfinite(double3);
04074 long4 const_func __attribute__((overloadable)) isfinite(double4);
04075 long8 const_func __attribute__((overloadable)) isfinite(double8);
04076 long16 const_func __attribute__((overloadable)) isfinite(double16);
04077 //int const_func __attribute__((overloadable)) isfinite(half);
04078 //short2 const_func __attribute__((overloadable)) isfinite(half2);
04079 //short3 const_func __attribute__((overloadable)) isfinite(half3);
04080 //short4 const_func __attribute__((overloadable)) isfinite(half4);
04081 //short8 const_func __attribute__((overloadable)) isfinite(half8);
04082 //short16 const_func __attribute__((overloadable)) isfinite(half16);
04083
04084 /**
04085 * Test for infinity value (+ve or -ve) .
04086 */
04087 int const_func __attribute__((overloadable)) isinf(float);
04088 int2 const_func __attribute__((overloadable)) isinf(float2);
04089 int3 const_func __attribute__((overloadable)) isinf(float3);
04090 int4 const_func __attribute__((overloadable)) isinf(float4);
04091 int8 const_func __attribute__((overloadable)) isinf(float8);
04092 int16 const_func __attribute__((overloadable)) isinf(float16);
04093 int const_func __attribute__((overloadable)) isinf(double);
04094 long2 const_func __attribute__((overloadable)) isinf(double2);
04095 long3 const_func __attribute__((overloadable)) isinf(double3);
04096 long4 const_func __attribute__((overloadable)) isinf(double4);
04097 long8 const_func __attribute__((overloadable)) isinf(double8);
04098 long16 const_func __attribute__((overloadable)) isinf(double16);
04099 //int const_func __attribute__((overloadable)) isinf(half);
04100 //short2 const_func __attribute__((overloadable)) isinf(half2);
04101 //short3 const_func __attribute__((overloadable)) isinf(half3);
04102 //short4 const_func __attribute__((overloadable)) isinf(half4);
04103 //short8 const_func __attribute__((overloadable)) isinf(half8);
04104 //short16 const_func __attribute__((overloadable)) isinf(half16);
04105
04106 /**
04107 * Test for a NaN.
04108 */
04109 int const_func __attribute__((overloadable)) isnan(float);
04110 int2 const_func __attribute__((overloadable)) isnan(float2);
04111 int3 const_func __attribute__((overloadable)) isnan(float3);
04112 int4 const_func __attribute__((overloadable)) isnan(float4);
04113 int8 const_func __attribute__((overloadable)) isnan(float8);
04114 int16 const_func __attribute__((overloadable)) isnan(float16);
04115 int const_func __attribute__((overloadable)) isnan(double);
04116 long2 const_func __attribute__((overloadable)) isnan(double2);
04117 long3 const_func __attribute__((overloadable)) isnan(double3);
04118 long4 const_func __attribute__((overloadable)) isnan(double4);
04119 long8 const_func __attribute__((overloadable)) isnan(double8);
04120 long16 const_func __attribute__((overloadable)) isnan(double16);
04121 //int const_func __attribute__((overloadable)) isnan(half);
04122 //short2 const_func __attribute__((overloadable)) isnan(half2);
04123 //short3 const_func __attribute__((overloadable)) isnan(half3);
04124 //short4 const_func __attribute__((overloadable)) isnan(half4);
04125 //short8 const_func __attribute__((overloadable)) isnan(half8);
04126 //short16 const_func __attribute__((overloadable)) isnan(half16);
04127
04128 /**
04129 * Test for a normal value.
04130 */
04131 int const_func __attribute__((overloadable)) isnormal(float);
04132 int2 const_func __attribute__((overloadable)) isnormal(float2);
04133 int3 const_func __attribute__((overloadable)) isnormal(float3);
04134 int4 const_func __attribute__((overloadable)) isnormal(float4);
04135 int8 const_func __attribute__((overloadable)) isnormal(float8);
04136 int16 const_func __attribute__((overloadable)) isnormal(float16);
04137 int const_func __attribute__((overloadable)) isnormal(double);
04138 long2 const_func __attribute__((overloadable)) isnormal(double2);
04139 long3 const_func __attribute__((overloadable)) isnormal(double3);
04140 long4 const_func __attribute__((overloadable)) isnormal(double4);
```

```
04141 long8 const_func __attribute__((overloadable)) isnormal(double8);
04142 long16 const_func __attribute__((overloadable)) isnormal(double16);
04143 //int const_func __attribute__((overloadable)) isnormal(half);
04144 //short2 const_func __attribute__((overloadable)) isnormal(half2);
04145 //short3 const_func __attribute__((overloadable)) isnormal(half3);
04146 //short4 const_func __attribute__((overloadable)) isnormal(half4);
04147 //short8 const_func __attribute__((overloadable)) isnormal(half8);
04148 //short16 const_func __attribute__((overloadable)) isnormal(half16);
04149
04150 /**
04151 * Test if arguments are ordered.  isordered() takes
04152 * arguments x and y, and returns the result
04153 * isequal(x, x) && isequal(y, y).
04154 */
04155 int const_func __attribute__((overloadable)) isordered(float x, float y);
04156 int2 const_func __attribute__((overloadable)) isordered(float2 x, float2 y);
04157 int3 const_func __attribute__((overloadable)) isordered(float3 x, float3 y);
04158 int4 const_func __attribute__((overloadable)) isordered(float4 x, float4 y);
04159 int8 const_func __attribute__((overloadable)) isordered(float8 x, float8 y);
04160 int16 const_func __attribute__((overloadable)) isordered(float16 x, float16 y);
04161 int const_func __attribute__((overloadable)) isordered(double x, double y);
04162 long2 const_func __attribute__((overloadable)) isordered(double2 x, double2 y);
04163 long3 const_func __attribute__((overloadable)) isordered(double3 x, double3 y);
04164 long4 const_func __attribute__((overloadable)) isordered(double4 x, double4 y);
04165 long8 const_func __attribute__((overloadable)) isordered(double8 x, double8 y);
04166 long16 const_func __attribute__((overloadable)) isordered(double16 x, double16 y);
04167 //int const_func __attribute__((overloadable)) isordered(half x, half y);
04168 //short2 const_func __attribute__((overloadable)) isordered(half2 x, half2 y);
04169 //short3 const_func __attribute__((overloadable)) isordered(half3 x, half3 y);
04170 //short4 const_func __attribute__((overloadable)) isordered(half4 x, half4 y);
04171 //short8 const_func __attribute__((overloadable)) isordered(half8 x, half8 y);
04172 //short16 const_func __attribute__((overloadable)) isordered(half16 x, half16 y);
04173
04174 /**
04175 * Test if arguments are unordered.  isunordered()
04176 * takes arguments x and y, returning non-zero if x or y
04177 * is NaN, and zero otherwise.
04178 */
04179 int const_func __attribute__((overloadable)) isunordered(float x, float y);
04180 int2 const_func __attribute__((overloadable)) isunordered(float2 x, float2 y);
04181 int3 const_func __attribute__((overloadable)) isunordered(float3 x, float3 y);
04182 int4 const_func __attribute__((overloadable)) isunordered(float4 x, float4 y);
04183 int8 const_func __attribute__((overloadable)) isunordered(float8 x, float8 y);
04184 int16 const_func __attribute__((overloadable)) isunordered(float16 x, float16 y);
04185 int const_func __attribute__((overloadable)) isunordered(double x, double y);
04186 long2 const_func __attribute__((overloadable)) isunordered(double2 x, double2 y);
04187 long3 const_func __attribute__((overloadable)) isunordered(double3 x, double3 y);
04188 long4 const_func __attribute__((overloadable)) isunordered(double4 x, double4 y);
04189 long8 const_func __attribute__((overloadable)) isunordered(double8 x, double8 y);
04190 long16 const_func __attribute__((overloadable)) isunordered(double16 x, double16 y);
04191 //int const_func __attribute__((overloadable)) isunordered(half x, half y);
04192 //short2 const_func __attribute__((overloadable)) isunordered(half2 x, half2 y);
04193 //short3 const_func __attribute__((overloadable)) isunordered(half3 x, half3 y);
04194 //short4 const_func __attribute__((overloadable)) isunordered(half4 x, half4 y);
04195 //short8 const_func __attribute__((overloadable)) isunordered(half8 x, half8 y);
04196 //short16 const_func __attribute__((overloadable)) isunordered(half16 x, half16 y);
04197
04198 /**
04199 * Test for sign bit.  The scalar version of the function
04200 * returns a 1 if the sign bit in the float is set else returns
04201 * 0.  The vector version of the function returns the
04202 * following for each component in floatn:  a -1 if the
04203 * sign bit in the float is set else returns 0.
04204 */
04205 int const_func __attribute__((overloadable)) signbit(float);
04206 int2 const_func __attribute__((overloadable)) signbit(float2);
04207 int3 const_func __attribute__((overloadable)) signbit(float3);
04208 int4 const_func __attribute__((overloadable)) signbit(float4);
04209 int8 const_func __attribute__((overloadable)) signbit(float8);
04210 int16 const_func __attribute__((overloadable)) signbit(float16);
04211 int const_func __attribute__((overloadable)) signbit(double);
04212 long2 const_func __attribute__((overloadable)) signbit(double2);
04213 long3 const_func __attribute__((overloadable)) signbit(double3);
04214 long4 const_func __attribute__((overloadable)) signbit(double4);
04215 long8 const_func __attribute__((overloadable)) signbit(double8);
04216 long16 const_func __attribute__((overloadable)) signbit(double16);
04217 //int const_func __attribute__((overloadable)) signbit(half);
04218 //short2 const_func __attribute__((overloadable)) signbit(half2);
04219 //short3 const_func __attribute__((overloadable)) signbit(half3);
04220 //short4 const_func __attribute__((overloadable)) signbit(half4);
04221 //short8 const_func __attribute__((overloadable)) signbit(half8);
04222 //short16 const_func __attribute__((overloadable)) signbit(half16);
04223
04224 /**
04225 * Returns 1 if the most significant bit in any component
04226 * of x is set; otherwise returns 0.
04227 */
```

```
04228 int const_func __attribute__((overloadable)) any(char x);
04229 int const_func __attribute__((overloadable)) any(char2 x);
04230 int const_func __attribute__((overloadable)) any(char3 x);
04231 int const_func __attribute__((overloadable)) any(char4 x);
04232 int const_func __attribute__((overloadable)) any(char8 x);
04233 int const_func __attribute__((overloadable)) any(char16 x);
04234 int const_func __attribute__((overloadable)) any(short x);
04235 int const_func __attribute__((overloadable)) any(short2 x);
04236 int const_func __attribute__((overloadable)) any(short3 x);
04237 int const_func __attribute__((overloadable)) any(short4 x);
04238 int const_func __attribute__((overloadable)) any(short8 x);
04239 int const_func __attribute__((overloadable)) any(short16 x);
04240 int const_func __attribute__((overloadable)) any(int x);
04241 int const_func __attribute__((overloadable)) any(int2 x);
04242 int const_func __attribute__((overloadable)) any(int3 x);
04243 int const_func __attribute__((overloadable)) any(int4 x);
04244 int const_func __attribute__((overloadable)) any(int8 x);
04245 int const_func __attribute__((overloadable)) any(int16 x);
04246 int const_func __attribute__((overloadable)) any(long x);
04247 int const_func __attribute__((overloadable)) any(long2 x);
04248 int const_func __attribute__((overloadable)) any(long3 x);
04249 int const_func __attribute__((overloadable)) any(long4 x);
04250 int const_func __attribute__((overloadable)) any(long8 x);
04251 int const_func __attribute__((overloadable)) any(long16 x);
04252
04253 /**
04254 * Returns 1 if the most significant bit in all components
04255 * of x is set; otherwise returns 0.
04256 */
04257 int const_func __attribute__((overloadable)) all(char x);
04258 int const_func __attribute__((overloadable)) all(char2 x);
04259 int const_func __attribute__((overloadable)) all(char3 x);
04260 int const_func __attribute__((overloadable)) all(char4 x);
04261 int const_func __attribute__((overloadable)) all(char8 x);
04262 int const_func __attribute__((overloadable)) all(char16 x);
04263 int const_func __attribute__((overloadable)) all(short x);
04264 int const_func __attribute__((overloadable)) all(short2 x);
04265 int const_func __attribute__((overloadable)) all(short3 x);
04266 int const_func __attribute__((overloadable)) all(short4 x);
04267 int const_func __attribute__((overloadable)) all(short8 x);
04268 int const_func __attribute__((overloadable)) all(short16 x);
04269 int const_func __attribute__((overloadable)) all(int x);
04270 int const_func __attribute__((overloadable)) all(int2 x);
04271 int const_func __attribute__((overloadable)) all(int3 x);
04272 int const_func __attribute__((overloadable)) all(int4 x);
04273 int const_func __attribute__((overloadable)) all(int8 x);
04274 int const_func __attribute__((overloadable)) all(int16 x);
04275 int const_func __attribute__((overloadable)) all(long x);
04276 int const_func __attribute__((overloadable)) all(long2 x);
04277 int const_func __attribute__((overloadable)) all(long3 x);
04278 int const_func __attribute__((overloadable)) all(long4 x);
04279 int const_func __attribute__((overloadable)) all(long8 x);
04280 int const_func __attribute__((overloadable)) all(long16 x);
04281
04282 /**
04283 * Each bit of the result is the corresponding bit of a if
04284 * the corresponding bit of c is 0.  Otherwise it is the
04285 * corresponding bit of b.
04286 */
04287 char const_func __attribute__((overloadable)) bitselect(char a, char b, char c);
04288 uchar const_func __attribute__((overloadable)) bitselect(uchar a, uchar b, uchar c);
04289 char2 const_func __attribute__((overloadable)) bitselect(char2 a, char2 b, char2 c);
04290 uchar2 const_func __attribute__((overloadable)) bitselect(uchar2 a, uchar2 b, uchar2 c);
04291 char3 const_func __attribute__((overloadable)) bitselect(char3 a, char3 b, char3 c);
04292 uchar3 const_func __attribute__((overloadable)) bitselect(uchar3 a, uchar3 b, uchar3 c);
04293 char4 const_func __attribute__((overloadable)) bitselect(char4 a, char4 b, char4 c);
04294 uchar4 const_func __attribute__((overloadable)) bitselect(uchar4 a, uchar4 b, uchar4 c);
04295 char8 const_func __attribute__((overloadable)) bitselect(char8 a, char8 b, char8 c);
04296 uchar8 const_func __attribute__((overloadable)) bitselect(uchar8 a, uchar8 b, uchar8 c);
04297 char16 const_func __attribute__((overloadable)) bitselect(char16 a, char16 b, char16 c);
04298 uchar16 const_func __attribute__((overloadable)) bitselect(uchar16 a, uchar16 b, uchar16 c);
04299 short const_func __attribute__((overloadable)) bitselect(short a, short b, short c);
04300 ushort const_func __attribute__((overloadable)) bitselect(ushort a, ushort b, ushort c);
04301 short2 const_func __attribute__((overloadable)) bitselect(short2 a, short2 b, short2 c);
04302 ushort2 const_func __attribute__((overloadable)) bitselect(ushort2 a, ushort2 b, ushort2 c);
04303 short3 const_func __attribute__((overloadable)) bitselect(short3 a, short3 b, short3 c);
04304 ushort3 const_func __attribute__((overloadable)) bitselect(ushort3 a, ushort3 b, ushort3 c);
04305 short4 const_func __attribute__((overloadable)) bitselect(short4 a, short4 b, short4 c);
04306 ushort4 const_func __attribute__((overloadable)) bitselect(ushort4 a, ushort4 b, ushort4 c);
04307 short8 const_func __attribute__((overloadable)) bitselect(short8 a, short8 b, short8 c);
04308 ushort8 const_func __attribute__((overloadable)) bitselect(ushort8 a, ushort8 b, ushort8 c);
04309 short16 const_func __attribute__((overloadable)) bitselect(short16 a, short16 b, short16 c);
04310 ushort16 const_func __attribute__((overloadable)) bitselect(ushort16 a, ushort16 b, ushort16 c);
04311 int const_func __attribute__((overloadable)) bitselect(int a, int b, int c);
04312 uint const_func __attribute__((overloadable)) bitselect(uint a, uint b, uint c);
04313 int2 const_func __attribute__((overloadable)) bitselect(int2 a, int2 b, int2 c);
04314 uint2 const_func __attribute__((overloadable)) bitselect(uint2 a, uint2 b, uint2 c);
```

```
04315 int3 const_func __attribute__((overloadable)) bitselect(int3 a, int3 b, int3 c);
04316 uint3 const_func __attribute__((overloadable)) bitselect(uint3 a, uint3 b, uint3 c);
04317 int4 const_func __attribute__((overloadable)) bitselect(int4 a, int4 b, int4 c);
04318 uint4 const_func __attribute__((overloadable)) bitselect(uint4 a, uint4 b, uint4 c);
04319 int8 const_func __attribute__((overloadable)) bitselect(int8 a, int8 b, int8 c);
04320 uint8 const_func __attribute__((overloadable)) bitselect(uint8 a, uint8 b, uint8 c);
04321 int16 const_func __attribute__((overloadable)) bitselect(int16 a, int16 b, int16 c);
04322 uint16 const_func __attribute__((overloadable)) bitselect(uint16 a, uint16 b, uint16 c);
04323 long const_func __attribute__((overloadable)) bitselect(long a, long b, long c);
04324 ulong const_func __attribute__((overloadable)) bitselect(ulong a, ulong b, ulong c);
04325 long2 const_func __attribute__((overloadable)) bitselect(long2 a, long2 b, long2 c);
04326 ulong2 const_func __attribute__((overloadable)) bitselect(ulong2 a, ulong2 b, ulong2 c);
04327 long3 const_func __attribute__((overloadable)) bitselect(long3 a, long3 b, long3 c);
04328 ulong3 const_func __attribute__((overloadable)) bitselect(ulong3 a, ulong3 b, ulong3 c);
04329 long4 const_func __attribute__((overloadable)) bitselect(long4 a, long4 b, long4 c);
04330 ulong4 const_func __attribute__((overloadable)) bitselect(ulong4 a, ulong4 b, ulong4 c);
04331 long8 const_func __attribute__((overloadable)) bitselect(long8 a, long8 b, long8 c);
04332 ulong8 const_func __attribute__((overloadable)) bitselect(ulong8 a, ulong8 b, ulong8 c);
04333 long16 const_func __attribute__((overloadable)) bitselect(long16 a, long16 b, long16 c);
04334 ulong16 const_func __attribute__((overloadable)) bitselect(ulong16 a, ulong16 b, ulong16 c);
04335 float const_func __attribute__((overloadable)) bitselect(float a, float b, float c);
04336 float2 const_func __attribute__((overloadable)) bitselect(float2 a, float2 b, float2 c);
04337 float3 const_func __attribute__((overloadable)) bitselect(float3 a, float3 b, float3 c);
04338 float4 const_func __attribute__((overloadable)) bitselect(float4 a, float4 b, float4 c);
04339 float8 const_func __attribute__((overloadable)) bitselect(float8 a, float8 b, float8 c);
04340 float16 const_func __attribute__((overloadable)) bitselect(float16 a, float16 b, float16 c);
04341 double const_func __attribute__((overloadable)) bitselect(double a, double b, double c);
04342 double2 const_func __attribute__((overloadable)) bitselect(double2 a, double2 b, double2 c);
04343 double3 const_func __attribute__((overloadable)) bitselect(double3 a, double3 b, double3 c);
04344 double4 const_func __attribute__((overloadable)) bitselect(double4 a, double4 b, double4 c);
04345 double8 const_func __attribute__((overloadable)) bitselect(double8 a, double8 b, double8 c);
04346 double16 const_func __attribute__((overloadable)) bitselect(double16 a, double16 b, double16 c);
04347 //half const_func __attribute__((overloadable)) bitselect(half a, half b, half c);
04348 //half2 const_func __attribute__((overloadable)) bitselect(half2 a, half2 b, half2 c);
04349 //half3 const_func __attribute__((overloadable)) bitselect(half3 a, half3 b, half3 c);
04350 //half4 const_func __attribute__((overloadable)) bitselect(half4 a, half4 b, half4 c);
04351 //half8 const_func __attribute__((overloadable)) bitselect(half8 a, half8 b, half8 c);
04352 //half16 const_func __attribute__((overloadable)) bitselect(half16 a, half16 b, half16 c);
04353
04354 /**
04355 * For each component of a vector type,
04356 * result[i] = if MSB of c[i] is set ?  b[i] :  a[i].
04357 * For a scalar type, result = c ?  b :  a.
04358 */
04359 char const_func __attribute__((overloadable)) select(char a, char b, char c);
04360 uchar const_func __attribute__((overloadable)) select(uchar a, uchar b, char c);
04361 char2 const_func __attribute__((overloadable)) select(char2 a, char2 b, char2 c);
04362 uchar2 const_func __attribute__((overloadable)) select(uchar2 a, uchar2 b, char2 c);
04363 char3 const_func __attribute__((overloadable)) select(char3 a, char3 b, char3 c);
04364 uchar3 const_func __attribute__((overloadable)) select(uchar3 a, uchar3 b, char3 c);
04365 char4 const_func __attribute__((overloadable)) select(char4 a, char4 b, char4 c);
04366 uchar4 const_func __attribute__((overloadable)) select(uchar4 a, uchar4 b, char4 c);
04367 char8 const_func __attribute__((overloadable)) select(char8 a, char8 b, char8 c);
04368 uchar8 const_func __attribute__((overloadable)) select(uchar8 a, uchar8 b, char8 c);
04369 char16 const_func __attribute__((overloadable)) select(char16 a, char16 b, char16 c);
04370 uchar16 const_func __attribute__((overloadable)) select(uchar16 a, uchar16 b, char16 c);
04371 short const_func __attribute__((overloadable)) select(short a, short b, char c);
04372 ushort const_func __attribute__((overloadable)) select(ushort a, ushort b, char c);
04373 short2 const_func __attribute__((overloadable)) select(short2 a, short2 b, char2 c);
04374 ushort2 const_func __attribute__((overloadable)) select(ushort2 a, ushort2 b, char2 c);
04375 short3 const_func __attribute__((overloadable)) select(short3 a, short3 b, char3 c);
04376 ushort3 const_func __attribute__((overloadable)) select(ushort3 a, ushort3 b, char3 c);
04377 short4 const_func __attribute__((overloadable)) select(short4 a, short4 b, char4 c);
04378 ushort4 const_func __attribute__((overloadable)) select(ushort4 a, ushort4 b, char4 c);
04379 short8 const_func __attribute__((overloadable)) select(short8 a, short8 b, char8 c);
04380 ushort8 const_func __attribute__((overloadable)) select(ushort8 a, ushort8 b, char8 c);
04381 short16 const_func __attribute__((overloadable)) select(short16 a, short16 b, char16 c);
04382 ushort16 const_func __attribute__((overloadable)) select(ushort16 a, ushort16 b, char16 c);
04383 int const_func __attribute__((overloadable)) select(int a, int b, char c);
04384 uint const_func __attribute__((overloadable)) select(uint a, uint b, char c);
04385 int2 const_func __attribute__((overloadable)) select(int2 a, int2 b, char2 c);
04386 uint2 const_func __attribute__((overloadable)) select(uint2 a, uint2 b, char2 c);
04387 int3 const_func __attribute__((overloadable)) select(int3 a, int3 b, char3 c);
04388 uint3 const_func __attribute__((overloadable)) select(uint3 a, uint3 b, char3 c);
04389 int4 const_func __attribute__((overloadable)) select(int4 a, int4 b, char4 c);
04390 uint4 const_func __attribute__((overloadable)) select(uint4 a, uint4 b, char4 c);
04391 int8 const_func __attribute__((overloadable)) select(int8 a, int8 b, char8 c);
04392 uint8 const_func __attribute__((overloadable)) select(uint8 a, uint8 b, char8 c);
04393 int16 const_func __attribute__((overloadable)) select(int16 a, int16 b, char16 c);
04394 uint16 const_func __attribute__((overloadable)) select(uint16 a, uint16 b, char16 c);
04395 long const_func __attribute__((overloadable)) select(long a, long b, char c);
04396 ulong const_func __attribute__((overloadable)) select(ulong a, ulong b, char c);
04397 long2 const_func __attribute__((overloadable)) select(long2 a, long2 b, char2 c);
04398 ulong2 const_func __attribute__((overloadable)) select(ulong2 a, ulong2 b, char2 c);
04399 long3 const_func __attribute__((overloadable)) select(long3 a, long3 b, char3 c);
04400 ulong3 const_func __attribute__((overloadable)) select(ulong3 a, ulong3 b, char3 c);
04401 long4 const_func __attribute__((overloadable)) select(long4 a, long4 b, char4 c);
```

```
04402 ulong4 const_func __attribute__((overloadable)) select(ulong4 a, ulong4 b, char4 c);
04403 long8 const_func __attribute__((overloadable)) select(long8 a, long8 b, char8 c);
04404 ulong8 const_func __attribute__((overloadable)) select(ulong8 a, ulong8 b, char8 c);
04405 long16 const_func __attribute__((overloadable)) select(long16 a, long16 b, char16 c);
04406 ulong16 const_func __attribute__((overloadable)) select(ulong16 a, ulong16 b, char16 c);
04407 float const_func __attribute__((overloadable)) select(float a, float b, char c);
04408 float2 const_func __attribute__((overloadable)) select(float2 a, float2 b, char2 c);
04409 float3 const_func __attribute__((overloadable)) select(float3 a, float3 b, char3 c);
04410 float4 const_func __attribute__((overloadable)) select(float4 a, float4 b, char4 c);
04411 float8 const_func __attribute__((overloadable)) select(float8 a, float8 b, char8 c);
04412 float16 const_func __attribute__((overloadable)) select(float16 a, float16 b, char16 c);
04413 char const_func __attribute__((overloadable)) select(char a, char b, short c);
04414 uchar const_func __attribute__((overloadable)) select(uchar a, uchar b, short c);
04415 char2 const_func __attribute__((overloadable)) select(char2 a, char2 b, short2 c);
04416 uchar2 const_func __attribute__((overloadable)) select(uchar2 a, uchar2 b, short2 c);
04417 char3 const_func __attribute__((overloadable)) select(char3 a, char3 b, short3 c);
04418 uchar3 const_func __attribute__((overloadable)) select(uchar3 a, uchar3 b, short3 c);
04419 char4 const_func __attribute__((overloadable)) select(char4 a, char4 b, short4 c);
04420 uchar4 const_func __attribute__((overloadable)) select(uchar4 a, uchar4 b, short4 c);
04421 char8 const_func __attribute__((overloadable)) select(char8 a, char8 b, short8 c);
04422 uchar8 const_func __attribute__((overloadable)) select(uchar8 a, uchar8 b, short8 c);
04423 char16 const_func __attribute__((overloadable)) select(char16 a, char16 b, short16 c);
04424 uchar16 const_func __attribute__((overloadable)) select(uchar16 a, uchar16 b, short16 c);
04425 short const_func __attribute__((overloadable)) select(short a, short b, short c);
04426 ushort const_func __attribute__((overloadable)) select(ushort a, ushort b, short c);
04427 short2 const_func __attribute__((overloadable)) select(short2 a, short2 b, short2 c);
04428 ushort2 const_func __attribute__((overloadable)) select(ushort2 a, ushort2 b, short2 c);
04429 short3 const_func __attribute__((overloadable)) select(short3 a, short3 b, short3 c);
04430 ushort3 const_func __attribute__((overloadable)) select(ushort3 a, ushort3 b, short3 c);
04431 short4 const_func __attribute__((overloadable)) select(short4 a, short4 b, short4 c);
04432 ushort4 const_func __attribute__((overloadable)) select(ushort4 a, ushort4 b, short4 c);
04433 short8 const_func __attribute__((overloadable)) select(short8 a, short8 b, short8 c);
04434 ushort8 const_func __attribute__((overloadable)) select(ushort8 a, ushort8 b, short8 c);
04435 short16 const_func __attribute__((overloadable)) select(short16 a, short16 b, short16 c);
04436 ushort16 const_func __attribute__((overloadable)) select(ushort16 a, ushort16 b, short16 c);
04437 int const_func __attribute__((overloadable)) select(int a, int b, short c);
04438 uint const_func __attribute__((overloadable)) select(uint a, uint b, short c);
04439 int2 const_func __attribute__((overloadable)) select(int2 a, int2 b, short2 c);
04440 uint2 const_func __attribute__((overloadable)) select(uint2 a, uint2 b, short2 c);
04441 int3 const_func __attribute__((overloadable)) select(int3 a, int3 b, short3 c);
04442 uint3 const_func __attribute__((overloadable)) select(uint3 a, uint3 b, short3 c);
04443 int4 const_func __attribute__((overloadable)) select(int4 a, int4 b, short4 c);
04444 uint4 const_func __attribute__((overloadable)) select(uint4 a, uint4 b, short4 c);
04445 int8 const_func __attribute__((overloadable)) select(int8 a, int8 b, short8 c);
04446 uint8 const_func __attribute__((overloadable)) select(uint8 a, uint8 b, short8 c);
04447 int16 const_func __attribute__((overloadable)) select(int16 a, int16 b, short16 c);
04448 uint16 const_func __attribute__((overloadable)) select(uint16 a, uint16 b, short16 c);
04449 long const_func __attribute__((overloadable)) select(long a, long b, short c);
04450 ulong const_func __attribute__((overloadable)) select(ulong a, ulong b, short c);
04451 long2 const_func __attribute__((overloadable)) select(long2 a, long2 b, short2 c);
04452 ulong2 const_func __attribute__((overloadable)) select(ulong2 a, ulong2 b, short2 c);
04453 long3 const_func __attribute__((overloadable)) select(long3 a, long3 b, short3 c);
04454 ulong3 const_func __attribute__((overloadable)) select(ulong3 a, ulong3 b, short3 c);
04455 long4 const_func __attribute__((overloadable)) select(long4 a, long4 b, short4 c);
04456 ulong4 const_func __attribute__((overloadable)) select(ulong4 a, ulong4 b, short4 c);
04457 long8 const_func __attribute__((overloadable)) select(long8 a, long8 b, short8 c);
04458 ulong8 const_func __attribute__((overloadable)) select(ulong8 a, ulong8 b, short8 c);
04459 long16 const_func __attribute__((overloadable)) select(long16 a, long16 b, short16 c);
04460 ulong16 const_func __attribute__((overloadable)) select(ulong16 a, ulong16 b, short16 c);
04461 float const_func __attribute__((overloadable)) select(float a, float b, short c);
04462 float2 const_func __attribute__((overloadable)) select(float2 a, float2 b, short2 c);
04463 float3 const_func __attribute__((overloadable)) select(float3 a, float3 b, short3 c);
04464 float4 const_func __attribute__((overloadable)) select(float4 a, float4 b, short4 c);
04465 float8 const_func __attribute__((overloadable)) select(float8 a, float8 b, short8 c);
04466 float16 const_func __attribute__((overloadable)) select(float16 a, float16 b, short16 c);
04467 char const_func __attribute__((overloadable)) select(char a, char b, int c);
04468 uchar const_func __attribute__((overloadable)) select(uchar a, uchar b, int c);
04469 char2 const_func __attribute__((overloadable)) select(char2 a, char2 b, int2 c);
04470 uchar2 const_func __attribute__((overloadable)) select(uchar2 a, uchar2 b, int2 c);
04471 char3 const_func __attribute__((overloadable)) select(char3 a, char3 b, int3 c);
04472 uchar3 const_func __attribute__((overloadable)) select(uchar3 a, uchar3 b, int3 c);
04473 char4 const_func __attribute__((overloadable)) select(char4 a, char4 b, int4 c);
04474 uchar4 const_func __attribute__((overloadable)) select(uchar4 a, uchar4 b, int4 c);
04475 char8 const_func __attribute__((overloadable)) select(char8 a, char8 b, int8 c);
04476 uchar8 const_func __attribute__((overloadable)) select(uchar8 a, uchar8 b, int8 c);
04477 char16 const_func __attribute__((overloadable)) select(char16 a, char16 b, int16 c);
04478 uchar16 const_func __attribute__((overloadable)) select(uchar16 a, uchar16 b, int16 c);
04479 short const_func __attribute__((overloadable)) select(short a, short b, int c);
04480 ushort const_func __attribute__((overloadable)) select(ushort a, ushort b, int c);
04481 short2 const_func __attribute__((overloadable)) select(short2 a, short2 b, int2 c);
04482 ushort2 const_func __attribute__((overloadable)) select(ushort2 a, ushort2 b, int2 c);
04483 short3 const_func __attribute__((overloadable)) select(short3 a, short3 b, int3 c);
04484 ushort3 const_func __attribute__((overloadable)) select(ushort3 a, ushort3 b, int3 c);
04485 short4 const_func __attribute__((overloadable)) select(short4 a, short4 b, int4 c);
04486 ushort4 const_func __attribute__((overloadable)) select(ushort4 a, ushort4 b, int4 c);
04487 short8 const_func __attribute__((overloadable)) select(short8 a, short8 b, int8 c);
04488 ushort8 const_func __attribute__((overloadable)) select(ushort8 a, ushort8 b, int8 c);
```

```
04489 short16 const_func __attribute__((overloadable)) select(short16 a, short16 b, int16 c);
04490 ushort16 const_func __attribute__((overloadable)) select(ushort16 a, ushort16 b, int16 c);
04491 int const_func __attribute__((overloadable)) select(int a, int b, int c);
04492 uint const_func __attribute__((overloadable)) select(uint a, uint b, int c);
04493 int2 const_func __attribute__((overloadable)) select(int2 a, int2 b, int2 c);
04494 uint2 const_func __attribute__((overloadable)) select(uint2 a, uint2 b, int2 c);
04495 int3 const_func __attribute__((overloadable)) select(int3 a, int3 b, int3 c);
04496 uint3 const_func __attribute__((overloadable)) select(uint3 a, uint3 b, int3 c);
04497 int4 const_func __attribute__((overloadable)) select(int4 a, int4 b, int4 c);
04498 uint4 const_func __attribute__((overloadable)) select(uint4 a, uint4 b, int4 c);
04499 int8 const_func __attribute__((overloadable)) select(int8 a, int8 b, int8 c);
04500 uint8 const_func __attribute__((overloadable)) select(uint8 a, uint8 b, int8 c);
04501 int16 const_func __attribute__((overloadable)) select(int16 a, int16 b, int16 c);
04502 uint16 const_func __attribute__((overloadable)) select(uint16 a, uint16 b, int16 c);
04503 long const_func __attribute__((overloadable)) select(long a, long b, int c);
04504 ulong const_func __attribute__((overloadable)) select(ulong a, ulong b, int c);
04505 long2 const_func __attribute__((overloadable)) select(long2 a, long2 b, int2 c);
04506 ulong2 const_func __attribute__((overloadable)) select(ulong2 a, ulong2 b, int2 c);
04507 long3 const_func __attribute__((overloadable)) select(long3 a, long3 b, int3 c);
04508 ulong3 const_func __attribute__((overloadable)) select(ulong3 a, ulong3 b, int3 c);
04509 long4 const_func __attribute__((overloadable)) select(long4 a, long4 b, int4 c);
04510 ulong4 const_func __attribute__((overloadable)) select(ulong4 a, ulong4 b, int4 c);
04511 long8 const_func __attribute__((overloadable)) select(long8 a, long8 b, int8 c);
04512 ulong8 const_func __attribute__((overloadable)) select(ulong8 a, ulong8 b, int8 c);
04513 long16 const_func __attribute__((overloadable)) select(long16 a, long16 b, int16 c);
04514 ulong16 const_func __attribute__((overloadable)) select(ulong16 a, ulong16 b, int16 c);
04515 float const_func __attribute__((overloadable)) select(float a, float b, int c);
04516 float2 const_func __attribute__((overloadable)) select(float2 a, float2 b, int2 c);
04517 float3 const_func __attribute__((overloadable)) select(float3 a, float3 b, int3 c);
04518 float4 const_func __attribute__((overloadable)) select(float4 a, float4 b, int4 c);
04519 float8 const_func __attribute__((overloadable)) select(float8 a, float8 b, int8 c);
04520 float16 const_func __attribute__((overloadable)) select(float16 a, float16 b, int16 c);
04521 char const_func __attribute__((overloadable)) select(char a, char b, long c);
04522 uchar const_func __attribute__((overloadable)) select(uchar a, uchar b, long c);
04523 char2 const_func __attribute__((overloadable)) select(char2 a, char2 b, long2 c);
04524 uchar2 const_func __attribute__((overloadable)) select(uchar2 a, uchar2 b, long2 c);
04525 char3 const_func __attribute__((overloadable)) select(char3 a, char3 b, long3 c);
04526 uchar3 const_func __attribute__((overloadable)) select(uchar3 a, uchar3 b, long3 c);
04527 char4 const_func __attribute__((overloadable)) select(char4 a, char4 b, long4 c);
04528 uchar4 const_func __attribute__((overloadable)) select(uchar4 a, uchar4 b, long4 c);
04529 char8 const_func __attribute__((overloadable)) select(char8 a, char8 b, long8 c);
04530 uchar8 const_func __attribute__((overloadable)) select(uchar8 a, uchar8 b, long8 c);
04531 char16 const_func __attribute__((overloadable)) select(char16 a, char16 b, long16 c);
04532 uchar16 const_func __attribute__((overloadable)) select(uchar16 a, uchar16 b, long16 c);
04533 short const_func __attribute__((overloadable)) select(short a, short b, long c);
04534 ushort const_func __attribute__((overloadable)) select(ushort a, ushort b, long c);
04535 short2 const_func __attribute__((overloadable)) select(short2 a, short2 b, long2 c);
04536 ushort2 const_func __attribute__((overloadable)) select(ushort2 a, ushort2 b, long2 c);
04537 short3 const_func __attribute__((overloadable)) select(short3 a, short3 b, long3 c);
04538 ushort3 const_func __attribute__((overloadable)) select(ushort3 a, ushort3 b, long3 c);
04539 short4 const_func __attribute__((overloadable)) select(short4 a, short4 b, long4 c);
04540 ushort4 const_func __attribute__((overloadable)) select(ushort4 a, ushort4 b, long4 c);
04541 short8 const_func __attribute__((overloadable)) select(short8 a, short8 b, long8 c);
04542 ushort8 const_func __attribute__((overloadable)) select(ushort8 a, ushort8 b, long8 c);
04543 short16 const_func __attribute__((overloadable)) select(short16 a, short16 b, long16 c);
04544 ushort16 const_func __attribute__((overloadable)) select(ushort16 a, ushort16 b, long16 c);
04545 int const_func __attribute__((overloadable)) select(int a, int b, long c);
04546 uint const_func __attribute__((overloadable)) select(uint a, uint b, long c);
04547 int2 const_func __attribute__((overloadable)) select(int2 a, int2 b, long2 c);
04548 uint2 const_func __attribute__((overloadable)) select(uint2 a, uint2 b, long2 c);
04549 int3 const_func __attribute__((overloadable)) select(int3 a, int3 b, long3 c);
04550 uint3 const_func __attribute__((overloadable)) select(uint3 a, uint3 b, long3 c);
04551 int4 const_func __attribute__((overloadable)) select(int4 a, int4 b, long4 c);
04552 uint4 const_func __attribute__((overloadable)) select(uint4 a, uint4 b, long4 c);
04553 int8 const_func __attribute__((overloadable)) select(int8 a, int8 b, long8 c);
04554 uint8 const_func __attribute__((overloadable)) select(uint8 a, uint8 b, long8 c);
04555 int16 const_func __attribute__((overloadable)) select(int16 a, int16 b, long16 c);
04556 uint16 const_func __attribute__((overloadable)) select(uint16 a, uint16 b, long16 c);
04557 long const_func __attribute__((overloadable)) select(long a, long b, long c);
04558 ulong const_func __attribute__((overloadable)) select(ulong a, ulong b, long c);
04559 long2 const_func __attribute__((overloadable)) select(long2 a, long2 b, long2 c);
04560 ulong2 const_func __attribute__((overloadable)) select(ulong2 a, ulong2 b, long2 c);
04561 long3 const_func __attribute__((overloadable)) select(long3 a, long3 b, long3 c);
04562 ulong3 const_func __attribute__((overloadable)) select(ulong3 a, ulong3 b, long3 c);
04563 long4 const_func __attribute__((overloadable)) select(long4 a, long4 b, long4 c);
04564 ulong4 const_func __attribute__((overloadable)) select(ulong4 a, ulong4 b, long4 c);
04565 long8 const_func __attribute__((overloadable)) select(long8 a, long8 b, long8 c);
04566 ulong8 const_func __attribute__((overloadable)) select(ulong8 a, ulong8 b, long8 c);
04567 long16 const_func __attribute__((overloadable)) select(long16 a, long16 b, long16 c);
04568 ulong16 const_func __attribute__((overloadable)) select(ulong16 a, ulong16 b, long16 c);
04569 float const_func __attribute__((overloadable)) select(float a, float b, long c);
04570 float2 const_func __attribute__((overloadable)) select(float2 a, float2 b, long2 c);
04571 float3 const_func __attribute__((overloadable)) select(float3 a, float3 b, long3 c);
04572 float4 const_func __attribute__((overloadable)) select(float4 a, float4 b, long4 c);
04573 float8 const_func __attribute__((overloadable)) select(float8 a, float8 b, long8 c);
04574 float16 const_func __attribute__((overloadable)) select(float16 a, float16 b, long16 c);
04575 char const_func __attribute__((overloadable)) select(char a, char b, uchar c);
```

```
04576 uchar const_func __attribute__((overloadable)) select(uchar a, uchar b, uchar c);
04577 char2 const_func __attribute__((overloadable)) select(char2 a, char2 b, uchar2 c);
04578 uchar2 const_func __attribute__((overloadable)) select(uchar2 a, uchar2 b, uchar2 c);
04579 char3 const_func __attribute__((overloadable)) select(char3 a, char3 b, uchar3 c);
04580 uchar3 const_func __attribute__((overloadable)) select(uchar3 a, uchar3 b, uchar3 c);
04581 char4 const_func __attribute__((overloadable)) select(char4 a, char4 b, uchar4 c);
04582 uchar4 const_func __attribute__((overloadable)) select(uchar4 a, uchar4 b, uchar4 c);
04583 char8 const_func __attribute__((overloadable)) select(char8 a, char8 b, uchar8 c);
04584 uchar8 const_func __attribute__((overloadable)) select(uchar8 a, uchar8 b, uchar8 c);
04585 char16 const_func __attribute__((overloadable)) select(char16 a, char16 b, uchar16 c);
04586 uchar16 const_func __attribute__((overloadable)) select(uchar16 a, uchar16 b, uchar16 c);
04587 short const_func __attribute__((overloadable)) select(short a, short b, uchar c);
04588 ushort const_func __attribute__((overloadable)) select(ushort a, ushort b, uchar c);
04589 short2 const_func __attribute__((overloadable)) select(short2 a, short2 b, uchar2 c);
04590 ushort2 const_func __attribute__((overloadable)) select(ushort2 a, ushort2 b, uchar2 c);
04591 short3 const_func __attribute__((overloadable)) select(short3 a, short3 b, uchar3 c);
04592 ushort3 const_func __attribute__((overloadable)) select(ushort3 a, ushort3 b, uchar3 c);
04593 short4 const_func __attribute__((overloadable)) select(short4 a, short4 b, uchar4 c);
04594 ushort4 const_func __attribute__((overloadable)) select(ushort4 a, ushort4 b, uchar4 c);
04595 short8 const_func __attribute__((overloadable)) select(short8 a, short8 b, uchar8 c);
04596 ushort8 const_func __attribute__((overloadable)) select(ushort8 a, ushort8 b, uchar8 c);
04597 short16 const_func __attribute__((overloadable)) select(short16 a, short16 b, uchar16 c);
04598 ushort16 const_func __attribute__((overloadable)) select(ushort16 a, ushort16 b, uchar16 c);
04599 int const_func __attribute__((overloadable)) select(int a, int b, uchar c);
04600 uint const_func __attribute__((overloadable)) select(uint a, uint b, uchar c);
04601 int2 const_func __attribute__((overloadable)) select(int2 a, int2 b, uchar2 c);
04602 uint2 const_func __attribute__((overloadable)) select(uint2 a, uint2 b, uchar2 c);
04603 int3 const_func __attribute__((overloadable)) select(int3 a, int3 b, uchar3 c);
04604 uint3 const_func __attribute__((overloadable)) select(uint3 a, uint3 b, uchar3 c);
04605 int4 const_func __attribute__((overloadable)) select(int4 a, int4 b, uchar4 c);
04606 uint4 const_func __attribute__((overloadable)) select(uint4 a, uint4 b, uchar4 c);
04607 int8 const_func __attribute__((overloadable)) select(int8 a, int8 b, uchar8 c);
04608 uint8 const_func __attribute__((overloadable)) select(uint8 a, uint8 b, uchar8 c);
04609 int16 const_func __attribute__((overloadable)) select(int16 a, int16 b, uchar16 c);
04610 uint16 const_func __attribute__((overloadable)) select(uint16 a, uint16 b, uchar16 c);
04611 long const_func __attribute__((overloadable)) select(long a, long b, uchar c);
04612 ulong const_func __attribute__((overloadable)) select(ulong a, ulong b, uchar c);
04613 long2 const_func __attribute__((overloadable)) select(long2 a, long2 b, uchar2 c);
04614 ulong2 const_func __attribute__((overloadable)) select(ulong2 a, ulong2 b, uchar2 c);
04615 long3 const_func __attribute__((overloadable)) select(long3 a, long3 b, uchar3 c);
04616 ulong3 const_func __attribute__((overloadable)) select(ulong3 a, ulong3 b, uchar3 c);
04617 long4 const_func __attribute__((overloadable)) select(long4 a, long4 b, uchar4 c);
04618 ulong4 const_func __attribute__((overloadable)) select(ulong4 a, ulong4 b, uchar4 c);
04619 long8 const_func __attribute__((overloadable)) select(long8 a, long8 b, uchar8 c);
04620 ulong8 const_func __attribute__((overloadable)) select(ulong8 a, ulong8 b, uchar8 c);
04621 long16 const_func __attribute__((overloadable)) select(long16 a, long16 b, uchar16 c);
04622 ulong16 const_func __attribute__((overloadable)) select(ulong16 a, ulong16 b, uchar16 c);
04623 float const_func __attribute__((overloadable)) select(float a, float b, uchar c);
04624 float2 const_func __attribute__((overloadable)) select(float2 a, float2 b, uchar2 c);
04625 float3 const_func __attribute__((overloadable)) select(float3 a, float3 b, uchar3 c);
04626 float4 const_func __attribute__((overloadable)) select(float4 a, float4 b, uchar4 c);
04627 float8 const_func __attribute__((overloadable)) select(float8 a, float8 b, uchar8 c);
04628 float16 const_func __attribute__((overloadable)) select(float16 a, float16 b, uchar16 c);
04629 char const_func __attribute__((overloadable)) select(char a, char b, ushort c);
04630 uchar const_func __attribute__((overloadable)) select(uchar a, uchar b, ushort c);
04631 char2 const_func __attribute__((overloadable)) select(char2 a, char2 b, ushort2 c);
04632 uchar2 const_func __attribute__((overloadable)) select(uchar2 a, uchar2 b, ushort2 c);
04633 char3 const_func __attribute__((overloadable)) select(char3 a, char3 b, ushort3 c);
04634 uchar3 const_func __attribute__((overloadable)) select(uchar3 a, uchar3 b, ushort3 c);
04635 char4 const_func __attribute__((overloadable)) select(char4 a, char4 b, ushort4 c);
04636 uchar4 const_func __attribute__((overloadable)) select(uchar4 a, uchar4 b, ushort4 c);
04637 char8 const_func __attribute__((overloadable)) select(char8 a, char8 b, ushort8 c);
04638 uchar8 const_func __attribute__((overloadable)) select(uchar8 a, uchar8 b, ushort8 c);
04639 char16 const_func __attribute__((overloadable)) select(char16 a, char16 b, ushort16 c);
04640 uchar16 const_func __attribute__((overloadable)) select(uchar16 a, uchar16 b, ushort16 c);
04641 short const_func __attribute__((overloadable)) select(short a, short b, ushort c);
04642 ushort const_func __attribute__((overloadable)) select(ushort a, ushort b, ushort c);
04643 short2 const_func __attribute__((overloadable)) select(short2 a, short2 b, ushort2 c);
04644 ushort2 const_func __attribute__((overloadable)) select(ushort2 a, ushort2 b, ushort2 c);
04645 short3 const_func __attribute__((overloadable)) select(short3 a, short3 b, ushort3 c);
04646 ushort3 const_func __attribute__((overloadable)) select(ushort3 a, ushort3 b, ushort3 c);
04647 short4 const_func __attribute__((overloadable)) select(short4 a, short4 b, ushort4 c);
04648 ushort4 const_func __attribute__((overloadable)) select(ushort4 a, ushort4 b, ushort4 c);
04649 short8 const_func __attribute__((overloadable)) select(short8 a, short8 b, ushort8 c);
04650 ushort8 const_func __attribute__((overloadable)) select(ushort8 a, ushort8 b, ushort8 c);
04651 short16 const_func __attribute__((overloadable)) select(short16 a, short16 b, ushort16 c);
04652 ushort16 const_func __attribute__((overloadable)) select(ushort16 a, ushort16 b, ushort16 c);
04653 int const_func __attribute__((overloadable)) select(int a, int b, ushort c);
04654 uint const_func __attribute__((overloadable)) select(uint a, uint b, ushort c);
04655 int2 const_func __attribute__((overloadable)) select(int2 a, int2 b, ushort2 c);
04656 uint2 const_func __attribute__((overloadable)) select(uint2 a, uint2 b, ushort2 c);
04657 int3 const_func __attribute__((overloadable)) select(int3 a, int3 b, ushort3 c);
04658 uint3 const_func __attribute__((overloadable)) select(uint3 a, uint3 b, ushort3 c);
04659 int4 const_func __attribute__((overloadable)) select(int4 a, int4 b, ushort4 c);
04660 uint4 const_func __attribute__((overloadable)) select(uint4 a, uint4 b, ushort4 c);
04661 int8 const_func __attribute__((overloadable)) select(int8 a, int8 b, ushort8 c);
04662 uint8 const_func __attribute__((overloadable)) select(uint8 a, uint8 b, ushort8 c);
```

```
04663 int16 const_func __attribute__((overloadable)) select(int16 a, int16 b, ushort16 c);
04664 uint16 const_func __attribute__((overloadable)) select(uint16 a, uint16 b, ushort16 c);
04665 long const_func __attribute__((overloadable)) select(long a, long b, ushort c);
04666 ulong const_func __attribute__((overloadable)) select(ulong a, ulong b, ushort c);
04667 long2 const_func __attribute__((overloadable)) select(long2 a, long2 b, ushort2 c);
04668 ulong2 const_func __attribute__((overloadable)) select(ulong2 a, ulong2 b, ushort2 c);
04669 long3 const_func __attribute__((overloadable)) select(long3 a, long3 b, ushort3 c);
04670 ulong3 const_func __attribute__((overloadable)) select(ulong3 a, ulong3 b, ushort3 c);
04671 long4 const_func __attribute__((overloadable)) select(long4 a, long4 b, ushort4 c);
04672 ulong4 const_func __attribute__((overloadable)) select(ulong4 a, ulong4 b, ushort4 c);
04673 long8 const_func __attribute__((overloadable)) select(long8 a, long8 b, ushort8 c);
04674 ulong8 const_func __attribute__((overloadable)) select(ulong8 a, ulong8 b, ushort8 c);
04675 long16 const_func __attribute__((overloadable)) select(long16 a, long16 b, ushort16 c);
04676 ulong16 const_func __attribute__((overloadable)) select(ulong16 a, ulong16 b, ushort16 c);
04677 float const_func __attribute__((overloadable)) select(float a, float b, ushort c);
04678 float2 const_func __attribute__((overloadable)) select(float2 a, float2 b, ushort2 c);
04679 float3 const_func __attribute__((overloadable)) select(float3 a, float3 b, ushort3 c);
04680 float4 const_func __attribute__((overloadable)) select(float4 a, float4 b, ushort4 c);
04681 float8 const_func __attribute__((overloadable)) select(float8 a, float8 b, ushort8 c);
04682 float16 const_func __attribute__((overloadable)) select(float16 a, float16 b, ushort16 c);
04683 char const_func __attribute__((overloadable)) select(char a, char b, uint c);
04684 uchar const_func __attribute__((overloadable)) select(uchar a, uchar b, uint c);
04685 char2 const_func __attribute__((overloadable)) select(char2 a, char2 b, uint2 c);
04686 uchar2 const_func __attribute__((overloadable)) select(uchar2 a, uchar2 b, uint2 c);
04687 char3 const_func __attribute__((overloadable)) select(char3 a, char3 b, uint3 c);
04688 uchar3 const_func __attribute__((overloadable)) select(uchar3 a, uchar3 b, uint3 c);
04689 char4 const_func __attribute__((overloadable)) select(char4 a, char4 b, uint4 c);
04690 uchar4 const_func __attribute__((overloadable)) select(uchar4 a, uchar4 b, uint4 c);
04691 char8 const_func __attribute__((overloadable)) select(char8 a, char8 b, uint8 c);
04692 uchar8 const_func __attribute__((overloadable)) select(uchar8 a, uchar8 b, uint8 c);
04693 char16 const_func __attribute__((overloadable)) select(char16 a, char16 b, uint16 c);
04694 uchar16 const_func __attribute__((overloadable)) select(uchar16 a, uchar16 b, uint16 c);
04695 short const_func __attribute__((overloadable)) select(short a, short b, uint c);
04696 ushort const_func __attribute__((overloadable)) select(ushort a, ushort b, uint c);
04697 short2 const_func __attribute__((overloadable)) select(short2 a, short2 b, uint2 c);
04698 ushort2 const_func __attribute__((overloadable)) select(ushort2 a, ushort2 b, uint2 c);
04699 short3 const_func __attribute__((overloadable)) select(short3 a, short3 b, uint3 c);
04700 ushort3 const_func __attribute__((overloadable)) select(ushort3 a, ushort3 b, uint3 c);
04701 short4 const_func __attribute__((overloadable)) select(short4 a, short4 b, uint4 c);
04702 ushort4 const_func __attribute__((overloadable)) select(ushort4 a, ushort4 b, uint4 c);
04703 short8 const_func __attribute__((overloadable)) select(short8 a, short8 b, uint8 c);
04704 ushort8 const_func __attribute__((overloadable)) select(ushort8 a, ushort8 b, uint8 c);
04705 short16 const_func __attribute__((overloadable)) select(short16 a, short16 b, uint16 c);
04706 ushort16 const_func __attribute__((overloadable)) select(ushort16 a, ushort16 b, uint16 c);
04707 int const_func __attribute__((overloadable)) select(int a, int b, uint c);
04708 uint const_func __attribute__((overloadable)) select(uint a, uint b, uint c);
04709 int2 const_func __attribute__((overloadable)) select(int2 a, int2 b, uint2 c);
04710 uint2 const_func __attribute__((overloadable)) select(uint2 a, uint2 b, uint2 c);
04711 int3 const_func __attribute__((overloadable)) select(int3 a, int3 b, uint3 c);
04712 uint3 const_func __attribute__((overloadable)) select(uint3 a, uint3 b, uint3 c);
04713 int4 const_func __attribute__((overloadable)) select(int4 a, int4 b, uint4 c);
04714 uint4 const_func __attribute__((overloadable)) select(uint4 a, uint4 b, uint4 c);
04715 int8 const_func __attribute__((overloadable)) select(int8 a, int8 b, uint8 c);
04716 uint8 const_func __attribute__((overloadable)) select(uint8 a, uint8 b, uint8 c);
04717 int16 const_func __attribute__((overloadable)) select(int16 a, int16 b, uint16 c);
04718 uint16 const_func __attribute__((overloadable)) select(uint16 a, uint16 b, uint16 c);
04719 long const_func __attribute__((overloadable)) select(long a, long b, uint c);
04720 ulong const_func __attribute__((overloadable)) select(ulong a, ulong b, uint c);
04721 long2 const_func __attribute__((overloadable)) select(long2 a, long2 b, uint2 c);
04722 ulong2 const_func __attribute__((overloadable)) select(ulong2 a, ulong2 b, uint2 c);
04723 long3 const_func __attribute__((overloadable)) select(long3 a, long3 b, uint3 c);
04724 ulong3 const_func __attribute__((overloadable)) select(ulong3 a, ulong3 b, uint3 c);
04725 long4 const_func __attribute__((overloadable)) select(long4 a, long4 b, uint4 c);
04726 ulong4 const_func __attribute__((overloadable)) select(ulong4 a, ulong4 b, uint4 c);
04727 long8 const_func __attribute__((overloadable)) select(long8 a, long8 b, uint8 c);
04728 ulong8 const_func __attribute__((overloadable)) select(ulong8 a, ulong8 b, uint8 c);
04729 long16 const_func __attribute__((overloadable)) select(long16 a, long16 b, uint16 c);
04730 ulong16 const_func __attribute__((overloadable)) select(ulong16 a, ulong16 b, uint16 c);
04731 float const_func __attribute__((overloadable)) select(float a, float b, uint c);
04732 float2 const_func __attribute__((overloadable)) select(float2 a, float2 b, uint2 c);
04733 float3 const_func __attribute__((overloadable)) select(float3 a, float3 b, uint3 c);
04734 float4 const_func __attribute__((overloadable)) select(float4 a, float4 b, uint4 c);
04735 float8 const_func __attribute__((overloadable)) select(float8 a, float8 b, uint8 c);
04736 float16 const_func __attribute__((overloadable)) select(float16 a, float16 b, uint16 c);
04737 char const_func __attribute__((overloadable)) select(char a, char b, ulong c);
04738 uchar const_func __attribute__((overloadable)) select(uchar a, uchar b, ulong c);
04739 char2 const_func __attribute__((overloadable)) select(char2 a, char2 b, ulong2 c);
04740 uchar2 const_func __attribute__((overloadable)) select(uchar2 a, uchar2 b, ulong2 c);
04741 char3 const_func __attribute__((overloadable)) select(char3 a, char3 b, ulong3 c);
04742 uchar3 const_func __attribute__((overloadable)) select(uchar3 a, uchar3 b, ulong3 c);
04743 char4 const_func __attribute__((overloadable)) select(char4 a, char4 b, ulong4 c);
04744 uchar4 const_func __attribute__((overloadable)) select(uchar4 a, uchar4 b, ulong4 c);
04745 char8 const_func __attribute__((overloadable)) select(char8 a, char8 b, ulong8 c);
04746 uchar8 const_func __attribute__((overloadable)) select(uchar8 a, uchar8 b, ulong8 c);
04747 char16 const_func __attribute__((overloadable)) select(char16 a, char16 b, ulong16 c);
04748 uchar16 const_func __attribute__((overloadable)) select(uchar16 a, uchar16 b, ulong16 c);
04749 short const_func __attribute__((overloadable)) select(short a, short b, ulong c);
```

```
04750 ushort const_func __attribute__((overloadable)) select(ushort a, ushort b, ulong c);
04751 short2 const_func __attribute__((overloadable)) select(short2 a, short2 b, ulong2 c);
04752 ushort2 const_func __attribute__((overloadable)) select(ushort2 a, ushort2 b, ulong2 c);
04753 short3 const_func __attribute__((overloadable)) select(short3 a, short3 b, ulong3 c);
04754 ushort3 const_func __attribute__((overloadable)) select(ushort3 a, ushort3 b, ulong3 c);
04755 short4 const_func __attribute__((overloadable)) select(short4 a, short4 b, ulong4 c);
04756 ushort4 const_func __attribute__((overloadable)) select(ushort4 a, ushort4 b, ulong4 c);
04757 short8 const_func __attribute__((overloadable)) select(short8 a, short8 b, ulong8 c);
04758 ushort8 const_func __attribute__((overloadable)) select(ushort8 a, ushort8 b, ulong8 c);
04759 short16 const_func __attribute__((overloadable)) select(short16 a, short16 b, ulong16 c);
04760 ushort16 const_func __attribute__((overloadable)) select(ushort16 a, ushort16 b, ulong16 c);
04761 int const_func __attribute__((overloadable)) select(int a, int b, ulong c);
04762 uint const_func __attribute__((overloadable)) select(uint a, uint b, ulong c);
04763 int2 const_func __attribute__((overloadable)) select(int2 a, int2 b, ulong2 c);
04764 uint2 const_func __attribute__((overloadable)) select(uint2 a, uint2 b, ulong2 c);
04765 int3 const_func __attribute__((overloadable)) select(int3 a, int3 b, ulong3 c);
04766 uint3 const_func __attribute__((overloadable)) select(uint3 a, uint3 b, ulong3 c);
04767 int4 const_func __attribute__((overloadable)) select(int4 a, int4 b, ulong4 c);
04768 uint4 const_func __attribute__((overloadable)) select(uint4 a, uint4 b, ulong4 c);
04769 int8 const_func __attribute__((overloadable)) select(int8 a, int8 b, ulong8 c);
04770 uint8 const_func __attribute__((overloadable)) select(uint8 a, uint8 b, ulong8 c);
04771 int16 const_func __attribute__((overloadable)) select(int16 a, int16 b, ulong16 c);
04772 uint16 const_func __attribute__((overloadable)) select(uint16 a, uint16 b, ulong16 c);
04773 long const_func __attribute__((overloadable)) select(long a, long b, ulong c);
04774 ulong const_func __attribute__((overloadable)) select(ulong a, ulong b, ulong c);
04775 long2 const_func __attribute__((overloadable)) select(long2 a, long2 b, ulong2 c);
04776 ulong2 const_func __attribute__((overloadable)) select(ulong2 a, ulong2 b, ulong2 c);
04777 long3 const_func __attribute__((overloadable)) select(long3 a, long3 b, ulong3 c);
04778 ulong3 const_func __attribute__((overloadable)) select(ulong3 a, ulong3 b, ulong3 c);
04779 long4 const_func __attribute__((overloadable)) select(long4 a, long4 b, ulong4 c);
04780 ulong4 const_func __attribute__((overloadable)) select(ulong4 a, ulong4 b, ulong4 c);
04781 long8 const_func __attribute__((overloadable)) select(long8 a, long8 b, ulong8 c);
04782 ulong8 const_func __attribute__((overloadable)) select(ulong8 a, ulong8 b, ulong8 c);
04783 long16 const_func __attribute__((overloadable)) select(long16 a, long16 b, ulong16 c);
04784 ulong16 const_func __attribute__((overloadable)) select(ulong16 a, ulong16 b, ulong16 c);
04785 float const_func __attribute__((overloadable)) select(float a, float b, ulong c);
04786 float2 const_func __attribute__((overloadable)) select(float2 a, float2 b, ulong2 c);
04787 float3 const_func __attribute__((overloadable)) select(float3 a, float3 b, ulong3 c);
04788 float4 const_func __attribute__((overloadable)) select(float4 a, float4 b, ulong4 c);
04789 float8 const_func __attribute__((overloadable)) select(float8 a, float8 b, ulong8 c);
04790 float16 const_func __attribute__((overloadable)) select(float16 a, float16 b, ulong16 c);
04791 double const_func __attribute__((overloadable)) select(double a, double b, long c);
04792 double2 const_func __attribute__((overloadable)) select(double2 a, double2 b, long2 c);
04793 double3 const_func __attribute__((overloadable)) select(double3 a, double3 b, long3 c);
04794 double4 const_func __attribute__((overloadable)) select(double4 a, double4 b, long4 c);
04795 double8 const_func __attribute__((overloadable)) select(double8 a, double8 b, long8 c);
04796 double16 const_func __attribute__((overloadable)) select(double16 a, double16 b, long16 c);
04797 double const_func __attribute__((overloadable)) select(double a, double b, ulong c);
04798 double2 const_func __attribute__((overloadable)) select(double2 a, double2 b, ulong2 c);
04799 double3 const_func __attribute__((overloadable)) select(double3 a, double3 b, ulong3 c);
04800 double4 const_func __attribute__((overloadable)) select(double4 a, double4 b, ulong4 c);
04801 double8 const_func __attribute__((overloadable)) select(double8 a, double8 b, ulong8 c);
04802 double16 const_func __attribute__((overloadable)) select(double16 a, double16 b, ulong16 c);
04803 //half const_func __attribute__((overloadable)) select(half a, half b, short c);
04804 //half2 const_func __attribute__((overloadable)) select(half2 a, half2 b, short2 c);
04805 //half3 const_func __attribute__((overloadable)) select(half3 a, half3 b, short3 c);
04806 //half4 const_func __attribute__((overloadable)) select(half4 a, half4 b, short4 c);
04807 //half8 const_func __attribute__((overloadable)) select(half8 a, half8 b, short8 c);
04808 //half16 const_func __attribute__((overloadable)) select(half16 a, half16 b, short16 c);
04809 //half const_func __attribute__((overloadable)) select(half a, half b, ushort c);
04810 //half2 const_func __attribute__((overloadable)) select(half2 a, half2 b, ushort2 c);
04811 //half3 const_func __attribute__((overloadable)) select(half3 a, half3 b, ushort3 c);
04812 //half4 const_func __attribute__((overloadable)) select(half4 a, half4 b, ushort4 c);
04813 //half8 const_func __attribute__((overloadable)) select(half8 a, half8 b, ushort8 c);
04814 //half16 const_func __attribute__((overloadable)) select(half16 a, half16 b, ushort16 c);
04815
04816 // Vector data load and store functions
04817
04818 /**
04819 * Return sizeof (gentypen) bytes of data read
04820 * from address (p + (offset * n)).  The address
04821 * computed as (p + (offset * n)) must be 8-bit
04822 * aligned if gentype is char, uchar; 16-bit
04823 * aligned if gentype is short, ushort; 32-bit
04824 * aligned if gentype is int, uint, float; 64-bit
04825 * aligned if gentype is long, ulong.
04826 */
04827 char2 __attribute__((overloadable)) vload2(size_t offset, const __global char *p);
04828 uchar2 __attribute__((overloadable)) vload2(size_t offset, const __global uchar *p);
04829 short2 __attribute__((overloadable)) vload2(size_t offset, const __global short *p);
04830 ushort2 __attribute__((overloadable)) vload2(size_t offset, const __global ushort *p);
04831 int2 __attribute__((overloadable)) vload2(size_t offset, const __global int *p);
04832 uint2 __attribute__((overloadable)) vload2(size_t offset, const __global uint *p);
04833 long2 __attribute__((overloadable)) vload2(size_t offset, const __global long *p);
04834 ulong2 __attribute__((overloadable)) vload2(size_t offset, const __global ulong *p);
04835 float2 __attribute__((overloadable)) vload2(size_t offset, const __global float *p);
04836 char3 __attribute__((overloadable)) vload3(size_t offset, const __global char *p);
```

```
04837 uchar3 __attribute__((overloadable)) vload3(size_t offset, const __global uchar *p);
04838 short3 __attribute__((overloadable)) vload3(size_t offset, const __global short *p);
04839 ushort3 __attribute__((overloadable)) vload3(size_t offset, const __global ushort *p);
04840 int3 __attribute__((overloadable)) vload3(size_t offset, const __global int *p);
04841 uint3 __attribute__((overloadable)) vload3(size_t offset, const __global uint *p);
04842 long3 __attribute__((overloadable)) vload3(size_t offset, const __global long *p);
04843 ulong3 __attribute__((overloadable)) vload3(size_t offset, const __global ulong *p);
04844 float3 __attribute__((overloadable)) vload3(size_t offset, const __global float *p);
04845 char4 __attribute__((overloadable)) vload4(size_t offset, const __global char *p);
04846 uchar4 __attribute__((overloadable)) vload4(size_t offset, const __global uchar *p);
04847 short4 __attribute__((overloadable)) vload4(size_t offset, const __global short *p);
04848 ushort4 __attribute__((overloadable)) vload4(size_t offset, const __global ushort *p);
04849 int4 __attribute__((overloadable)) vload4(size_t offset, const __global int *p);
04850 uint4 __attribute__((overloadable)) vload4(size_t offset, const __global uint *p);
04851 long4 __attribute__((overloadable)) vload4(size_t offset, const __global long *p);
04852 ulong4 __attribute__((overloadable)) vload4(size_t offset, const __global ulong *p);
04853 float4 __attribute__((overloadable)) vload4(size_t offset, const __global float *p);
04854 char8 __attribute__((overloadable)) vload8(size_t offset, const __global char *p);
04855 uchar8 __attribute__((overloadable)) vload8(size_t offset, const __global uchar *p);
04856 short8 __attribute__((overloadable)) vload8(size_t offset, const __global short *p);
04857 ushort8 __attribute__((overloadable)) vload8(size_t offset, const __global ushort *p);
04858 int8 __attribute__((overloadable)) vload8(size_t offset, const __global int *p);
04859 uint8 __attribute__((overloadable)) vload8(size_t offset, const __global uint *p);
04860 long8 __attribute__((overloadable)) vload8(size_t offset, const __global long *p);
04861 ulong8 __attribute__((overloadable)) vload8(size_t offset, const __global ulong *p);
04862 float8 __attribute__((overloadable)) vload8(size_t offset, const __global float *p);
04863 char16 __attribute__((overloadable)) vload16(size_t offset, const __global char *p);
04864 uchar16 __attribute__((overloadable)) vload16(size_t offset, const __global uchar *p);
04865 short16 __attribute__((overloadable)) vload16(size_t offset, const __global short *p);
04866 ushort16 __attribute__((overloadable)) vload16(size_t offset, const __global ushort *p);
04867 int16 __attribute__((overloadable)) vload16(size_t offset, const __global int *p);
04868 uint16 __attribute__((overloadable)) vload16(size_t offset, const __global uint *p);
04869 long16 __attribute__((overloadable)) vload16(size_t offset, const __global long *p);
04870 ulong16 __attribute__((overloadable)) vload16(size_t offset, const __global ulong *p);
04871 float16 __attribute__((overloadable)) vload16(size_t offset, const __global float *p);
04872 char2 __attribute__((overloadable)) vload2(size_t offset, const __local char *p);
04873 uchar2 __attribute__((overloadable)) vload2(size_t offset, const __local uchar *p);
04874 short2 __attribute__((overloadable)) vload2(size_t offset, const __local short *p);
04875 ushort2 __attribute__((overloadable)) vload2(size_t offset, const __local ushort *p);
04876 int2 __attribute__((overloadable)) vload2(size_t offset, const __local int *p);
04877 uint2 __attribute__((overloadable)) vload2(size_t offset, const __local uint *p);
04878 long2 __attribute__((overloadable)) vload2(size_t offset, const __local long *p);
04879 ulong2 __attribute__((overloadable)) vload2(size_t offset, const __local ulong *p);
04880 float2 __attribute__((overloadable)) vload2(size_t offset, const __local float *p);
04881 char3 __attribute__((overloadable)) vload3(size_t offset, const __local char *p);
04882 uchar3 __attribute__((overloadable)) vload3(size_t offset, const __local uchar *p);
04883 short3 __attribute__((overloadable)) vload3(size_t offset, const __local short *p);
04884 ushort3 __attribute__((overloadable)) vload3(size_t offset, const __local ushort *p);
04885 int3 __attribute__((overloadable)) vload3(size_t offset, const __local int *p);
04886 uint3 __attribute__((overloadable)) vload3(size_t offset, const __local uint *p);
04887 long3 __attribute__((overloadable)) vload3(size_t offset, const __local long *p);
04888 ulong3 __attribute__((overloadable)) vload3(size_t offset, const __local ulong *p);
04889 float3 __attribute__((overloadable)) vload3(size_t offset, const __local float *p);
04890 char4 __attribute__((overloadable)) vload4(size_t offset, const __local char *p);
04891 uchar4 __attribute__((overloadable)) vload4(size_t offset, const __local uchar *p);
04892 short4 __attribute__((overloadable)) vload4(size_t offset, const __local short *p);
04893 ushort4 __attribute__((overloadable)) vload4(size_t offset, const __local ushort *p);
04894 int4 __attribute__((overloadable)) vload4(size_t offset, const __local int *p);
04895 uint4 __attribute__((overloadable)) vload4(size_t offset, const __local uint *p);
04896 long4 __attribute__((overloadable)) vload4(size_t offset, const __local long *p);
04897 ulong4 __attribute__((overloadable)) vload4(size_t offset, const __local ulong *p);
04898 float4 __attribute__((overloadable)) vload4(size_t offset, const __local float *p);
04899 char8 __attribute__((overloadable)) vload8(size_t offset, const __local char *p);
04900 uchar8 __attribute__((overloadable)) vload8(size_t offset, const __local uchar *p);
04901 short8 __attribute__((overloadable)) vload8(size_t offset, const __local short *p);
04902 ushort8 __attribute__((overloadable)) vload8(size_t offset, const __local ushort *p);
04903 int8 __attribute__((overloadable)) vload8(size_t offset, const __local int *p);
04904 uint8 __attribute__((overloadable)) vload8(size_t offset, const __local uint *p);
04905 long8 __attribute__((overloadable)) vload8(size_t offset, const __local long *p);
04906 ulong8 __attribute__((overloadable)) vload8(size_t offset, const __local ulong *p);
04907 float8 __attribute__((overloadable)) vload8(size_t offset, const __local float *p);
04908 char16 __attribute__((overloadable)) vload16(size_t offset, const __local char *p);
04909 uchar16 __attribute__((overloadable)) vload16(size_t offset, const __local uchar *p);
04910 short16 __attribute__((overloadable)) vload16(size_t offset, const __local short *p);
04911 ushort16 __attribute__((overloadable)) vload16(size_t offset, const __local ushort *p);
04912 int16 __attribute__((overloadable)) vload16(size_t offset, const __local int *p);
04913 uint16 __attribute__((overloadable)) vload16(size_t offset, const __local uint *p);
04914 long16 __attribute__((overloadable)) vload16(size_t offset, const __local long *p);
04915 ulong16 __attribute__((overloadable)) vload16(size_t offset, const __local ulong *p);
04916 float16 __attribute__((overloadable)) vload16(size_t offset, const __local float *p);
04917 char2 __attribute__((overloadable)) vload2(size_t offset, const __private char *p);
04918 uchar2 __attribute__((overloadable)) vload2(size_t offset, const __private uchar *p);
04919 short2 __attribute__((overloadable)) vload2(size_t offset, const __private short *p);
04920 ushort2 __attribute__((overloadable)) vload2(size_t offset, const __private ushort *p);
04921 int2 __attribute__((overloadable)) vload2(size_t offset, const __private int *p);
04922 uint2 __attribute__((overloadable)) vload2(size_t offset, const __private uint *p);
04923 long2 __attribute__((overloadable)) vload2(size_t offset, const __private long *p);
```

```
04924 ulong2 __attribute__((overloadable)) vload2(size_t offset, const __private ulong *p);
04925 float2 __attribute__((overloadable)) vload2(size_t offset, const __private float *p);
04926 char3 __attribute__((overloadable)) vload3(size_t offset, const __private char *p);
04927 uchar3 __attribute__((overloadable)) vload3(size_t offset, const __private uchar *p);
04928 short3 __attribute__((overloadable)) vload3(size_t offset, const __private short *p);
04929 ushort3 __attribute__((overloadable)) vload3(size_t offset, const __private ushort *p);
04930 int3 __attribute__((overloadable)) vload3(size_t offset, const __private int *p);
04931 uint3 __attribute__((overloadable)) vload3(size_t offset, const __private uint *p);
04932 long3 __attribute__((overloadable)) vload3(size_t offset, const __private long *p);
04933 ulong3 __attribute__((overloadable)) vload3(size_t offset, const __private ulong *p);
04934 float3 __attribute__((overloadable)) vload3(size_t offset, const __private float *p);
04935 char4 __attribute__((overloadable)) vload4(size_t offset, const __private char *p);
04936 uchar4 __attribute__((overloadable)) vload4(size_t offset, const __private uchar *p);
04937 short4 __attribute__((overloadable)) vload4(size_t offset, const __private short *p);
04938 ushort4 __attribute__((overloadable)) vload4(size_t offset, const __private ushort *p);
04939 int4 __attribute__((overloadable)) vload4(size_t offset, const __private int *p);
04940 uint4 __attribute__((overloadable)) vload4(size_t offset, const __private uint *p);
04941 long4 __attribute__((overloadable)) vload4(size_t offset, const __private long *p);
04942 ulong4 __attribute__((overloadable)) vload4(size_t offset, const __private ulong *p);
04943 float4 __attribute__((overloadable)) vload4(size_t offset, const __private float *p);
04944 char8 __attribute__((overloadable)) vload8(size_t offset, const __private char *p);
04945 uchar8 __attribute__((overloadable)) vload8(size_t offset, const __private uchar *p);
04946 short8 __attribute__((overloadable)) vload8(size_t offset, const __private short *p);
04947 ushort8 __attribute__((overloadable)) vload8(size_t offset, const __private ushort *p);
04948 int8 __attribute__((overloadable)) vload8(size_t offset, const __private int *p);
04949 uint8 __attribute__((overloadable)) vload8(size_t offset, const __private uint *p);
04950 long8 __attribute__((overloadable)) vload8(size_t offset, const __private long *p);
04951 ulong8 __attribute__((overloadable)) vload8(size_t offset, const __private ulong *p);
04952 float8 __attribute__((overloadable)) vload8(size_t offset, const __private float *p);
04953 char16 __attribute__((overloadable)) vload16(size_t offset, const __private char *p);
04954 uchar16 __attribute__((overloadable)) vload16(size_t offset, const __private uchar *p);
04955 short16 __attribute__((overloadable)) vload16(size_t offset, const __private short *p);
04956 ushort16 __attribute__((overloadable)) vload16(size_t offset, const __private ushort *p);
04957 int16 __attribute__((overloadable)) vload16(size_t offset, const __private int *p);
04958 uint16 __attribute__((overloadable)) vload16(size_t offset, const __private uint *p);
04959 long16 __attribute__((overloadable)) vload16(size_t offset, const __private long *p);
04960 ulong16 __attribute__((overloadable)) vload16(size_t offset, const __private ulong *p);
04961 float16 __attribute__((overloadable)) vload16(size_t offset, const __private float *p);
04962 double2 __attribute__((overloadable)) vload2(size_t offset, const __global double *p);
04963 double3 __attribute__((overloadable)) vload3(size_t offset, const __global double *p);
04964 double4 __attribute__((overloadable)) vload4(size_t offset, const __global double *p);
04965 double8 __attribute__((overloadable)) vload8(size_t offset, const __global double *p);
04966 double16 __attribute__((overloadable)) vload16(size_t offset, const __global double *p);
04967 double2 __attribute__((overloadable)) vload2(size_t offset, const __local double *p);
04968 double3 __attribute__((overloadable)) vload3(size_t offset, const __local double *p);
04969 double4 __attribute__((overloadable)) vload4(size_t offset, const __local double *p);
04970 double8 __attribute__((overloadable)) vload8(size_t offset, const __local double *p);
04971 double16 __attribute__((overloadable)) vload16(size_t offset, const __local double *p);
04972 double2 __attribute__((overloadable)) vload2(size_t offset, const __private double *p);
04973 double3 __attribute__((overloadable)) vload3(size_t offset, const __private double *p);
04974 double4 __attribute__((overloadable)) vload4(size_t offset, const __private double *p);
04975 double8 __attribute__((overloadable)) vload8(size_t offset, const __private double *p);
04976 double16 __attribute__((overloadable)) vload16(size_t offset, const __private double *p);
04977 //half __attribute__((overloadable)) vload(size_t offset, const __global half *p);
04978 //half2 __attribute__((overloadable)) vload2(size_t offset, const __global half *p);
04979 //half3 __attribute__((overloadable)) vload3(size_t offset, const __global half *p);
04980 //half4 __attribute__((overloadable)) vload4(size_t offset, const __global half *p);
04981 //half8 __attribute__((overloadable)) vload8(size_t offset, const __global half *p);
04982 //half16 __attribute__((overloadable)) vload16(size_t offset, const __global half *p);
04983 //half __attribute__((overloadable)) vload(size_t offset, const __local half *p);
04984 //half2 __attribute__((overloadable)) vload2(size_t offset, const __local half *p);
04985 //half3 __attribute__((overloadable)) vload3(size_t offset, const __local half *p);
04986 //half4 __attribute__((overloadable)) vload4(size_t offset, const __local half *p);
04987 //half8 __attribute__((overloadable)) vload8(size_t offset, const __local half *p);
04988 //half16 __attribute__((overloadable)) vload16(size_t offset, const __local half *p);
04989 //half __attribute__((overloadable)) vload(size_t offset, const __private half *p);
04990 //half2 __attribute__((overloadable)) vload2(size_t offset, const __private half *p);
04991 //half3 __attribute__((overloadable)) vload3(size_t offset, const __private half *p);
04992 //half4 __attribute__((overloadable)) vload4(size_t offset, const __private half *p);
04993 //half8 __attribute__((overloadable)) vload8(size_t offset, const __private half *p);
04994 //half16 __attribute__((overloadable)) vload16(size_t offset, const __private half *p);
04995
04996 char2 __attribute__((overloadable)) vload2(size_t offset, const __constant char *p);
04997 uchar2 __attribute__((overloadable)) vload2(size_t offset, const __constant uchar *p);
04998 short2 __attribute__((overloadable)) vload2(size_t offset, const __constant short *p);
04999 ushort2 __attribute__((overloadable)) vload2(size_t offset, const __constant ushort *p);
05000 int2 __attribute__((overloadable)) vload2(size_t offset, const __constant int *p);
05001 uint2 __attribute__((overloadable)) vload2(size_t offset, const __constant uint *p);
05002 long2 __attribute__((overloadable)) vload2(size_t offset, const __constant long *p);
05003 ulong2 __attribute__((overloadable)) vload2(size_t offset, const __constant ulong *p);
05004 float2 __attribute__((overloadable)) vload2(size_t offset, const __constant float *p);
05005 char3 __attribute__((overloadable)) vload3(size_t offset, const __constant char *p);
05006 uchar3 __attribute__((overloadable)) vload3(size_t offset, const __constant uchar *p);
05007 short3 __attribute__((overloadable)) vload3(size_t offset, const __constant short *p);
05008 ushort3 __attribute__((overloadable)) vload3(size_t offset, const __constant ushort *p);
05009 int3 __attribute__((overloadable)) vload3(size_t offset, const __constant int *p);
05010 uint3 __attribute__((overloadable)) vload3(size_t offset, const __constant uint *p);
```

```
05011 long3 __attribute__((overloadable)) vload3(size_t offset, const __constant long *p);
05012 ulong3 __attribute__((overloadable)) vload3(size_t offset, const __constant ulong *p);
05013 float3 __attribute__((overloadable)) vload3(size_t offset, const __constant float *p);
05014 char4 __attribute__((overloadable)) vload4(size_t offset, const __constant char *p);
05015 uchar4 __attribute__((overloadable)) vload4(size_t offset, const __constant uchar *p);
05016 short4 __attribute__((overloadable)) vload4(size_t offset, const __constant short *p);
05017 ushort4 __attribute__((overloadable)) vload4(size_t offset, const __constant ushort *p);
05018 int4 __attribute__((overloadable)) vload4(size_t offset, const __constant int *p);
05019 uint4 __attribute__((overloadable)) vload4(size_t offset, const __constant uint *p);
05020 long4 __attribute__((overloadable)) vload4(size_t offset, const __constant long *p);
05021 ulong4 __attribute__((overloadable)) vload4(size_t offset, const __constant ulong *p);
05022 float4 __attribute__((overloadable)) vload4(size_t offset, const __constant float *p);
05023 char8 __attribute__((overloadable)) vload8(size_t offset, const __constant char *p);
05024 uchar8 __attribute__((overloadable)) vload8(size_t offset, const __constant uchar *p);
05025 short8 __attribute__((overloadable)) vload8(size_t offset, const __constant short *p);
05026 ushort8 __attribute__((overloadable)) vload8(size_t offset, const __constant ushort *p);
05027 int8 __attribute__((overloadable)) vload8(size_t offset, const __constant int *p);
05028 uint8 __attribute__((overloadable)) vload8(size_t offset, const __constant uint *p);
05029 long8 __attribute__((overloadable)) vload8(size_t offset, const __constant long *p);
05030 ulong8 __attribute__((overloadable)) vload8(size_t offset, const __constant ulong *p);
05031 float8 __attribute__((overloadable)) vload8(size_t offset, const __constant float *p);
05032 char16 __attribute__((overloadable)) vload16(size_t offset, const __constant char *p);
05033 uchar16 __attribute__((overloadable)) vload16(size_t offset, const __constant uchar *p);
05034 short16 __attribute__((overloadable)) vload16(size_t offset, const __constant short *p);
05035 ushort16 __attribute__((overloadable)) vload16(size_t offset, const __constant ushort *p);
05036 int16 __attribute__((overloadable)) vload16(size_t offset, const __constant int *p);
05037 uint16 __attribute__((overloadable)) vload16(size_t offset, const __constant uint *p);
05038 long16 __attribute__((overloadable)) vload16(size_t offset, const __constant long *p);
05039 ulong16 __attribute__((overloadable)) vload16(size_t offset, const __constant ulong *p);
05040 float16 __attribute__((overloadable)) vload16(size_t offset, const __constant float *p);
05041 double2 __attribute__((overloadable)) vload2(size_t offset, const __constant double *p);
05042 double3 __attribute__((overloadable)) vload3(size_t offset, const __constant double *p);
05043 double4 __attribute__((overloadable)) vload4(size_t offset, const __constant double *p);
05044 double8 __attribute__((overloadable)) vload8(size_t offset, const __constant double *p);
05045 double16 __attribute__((overloadable)) vload16(size_t offset, const __constant double *p);
05046 //half __attribute__((overloadable)) vload(size_t offset, const __constant half *p);
05047 //half2 __attribute__((overloadable)) vload2(size_t offset, const __constant half *p);
05048 //half3 __attribute__((overloadable)) vload3(size_t offset, const __constant half *p);
05049 //half4 __attribute__((overloadable)) vload4(size_t offset, const __constant half *p);
05050 //half8 __attribute__((overloadable)) vload8(size_t offset, const __constant half *p);
05051 //half16 __attribute__((overloadable)) vload16(size_t offset, const __constant half *p);
05052
05053 /**
05054 * Write sizeof (gentypen) bytes given by data
05055 * to address (p + (offset * n)).  The address
05056 * computed as (p + (offset * n)) must be 8-bit
05057 * aligned if gentype is char, uchar; 16-bit
05058 * aligned if gentype is short, ushort; 32-bit
05059 * aligned if gentype is int, uint, float; 64-bit
05060 * aligned if gentype is long, ulong.
05061 */
05062 void __attribute__((overloadable)) vstore2(char2 data, size_t offset, __global char *p);
05063 void __attribute__((overloadable)) vstore2(uchar2 data, size_t offset, __global uchar *p);
05064 void __attribute__((overloadable)) vstore2(short2 data, size_t offset, __global short *p);
05065 void __attribute__((overloadable)) vstore2(ushort2 data, size_t offset, __global ushort *p);
05066 void __attribute__((overloadable)) vstore2(int2 data, size_t offset, __global int *p);
05067 void __attribute__((overloadable)) vstore2(uint2 data, size_t offset, __global uint *p);
05068 void __attribute__((overloadable)) vstore2(long2 data, size_t offset, __global long *p);
05069 void __attribute__((overloadable)) vstore2(ulong2 data, size_t offset, __global ulong *p);
05070 void __attribute__((overloadable)) vstore2(float2 data, size_t offset, __global float *p);
05071 void __attribute__((overloadable)) vstore3(char3 data, size_t offset, __global char *p);
05072 void __attribute__((overloadable)) vstore3(uchar3 data, size_t offset, __global uchar *p);
05073 void __attribute__((overloadable)) vstore3(short3 data, size_t offset, __global short *p);
05074 void __attribute__((overloadable)) vstore3(ushort3 data, size_t offset, __global ushort *p);
05075 void __attribute__((overloadable)) vstore3(int3 data, size_t offset, __global int *p);
05076 void __attribute__((overloadable)) vstore3(uint3 data, size_t offset, __global uint *p);
05077 void __attribute__((overloadable)) vstore3(long3 data, size_t offset, __global long *p);
05078 void __attribute__((overloadable)) vstore3(ulong3 data, size_t offset, __global ulong *p);
05079 void __attribute__((overloadable)) vstore3(float3 data, size_t offset, __global float *p);
05080 void __attribute__((overloadable)) vstore4(char4 data, size_t offset, __global char *p);
05081 void __attribute__((overloadable)) vstore4(uchar4 data, size_t offset, __global uchar *p);
05082 void __attribute__((overloadable)) vstore4(short4 data, size_t offset, __global short *p);
05083 void __attribute__((overloadable)) vstore4(ushort4 data, size_t offset, __global ushort *p);
05084 void __attribute__((overloadable)) vstore4(int4 data, size_t offset, __global int *p);
05085 void __attribute__((overloadable)) vstore4(uint4 data, size_t offset, __global uint *p);
05086 void __attribute__((overloadable)) vstore4(long4 data, size_t offset, __global long *p);
05087 void __attribute__((overloadable)) vstore4(ulong4 data, size_t offset, __global ulong *p);
05088 void __attribute__((overloadable)) vstore4(float4 data, size_t offset, __global float *p);
05089 void __attribute__((overloadable)) vstore8(char8 data, size_t offset, __global char *p);
05090 void __attribute__((overloadable)) vstore8(uchar8 data, size_t offset, __global uchar *p);
05091 void __attribute__((overloadable)) vstore8(short8 data, size_t offset, __global short *p);
05092 void __attribute__((overloadable)) vstore8(ushort8 data, size_t offset, __global ushort *p);
05093 void __attribute__((overloadable)) vstore8(int8 data, size_t offset, __global int *p);
05094 void __attribute__((overloadable)) vstore8(uint8 data, size_t offset, __global uint *p);
05095 void __attribute__((overloadable)) vstore8(long8 data, size_t offset, __global long *p);
05096 void __attribute__((overloadable)) vstore8(ulong8 data, size_t offset, __global ulong *p);
05097 void __attribute__((overloadable)) vstore8(float8 data, size_t offset, __global float *p);
```

```
05098 void __attribute__((overloadable)) vstore16(char16 data, size_t offset, __global char *p);
05099 void __attribute__((overloadable)) vstore16(uchar16 data, size_t offset, __global uchar *p);
05100 void __attribute__((overloadable)) vstore16(short16 data, size_t offset, __global short *p);
05101 void __attribute__((overloadable)) vstore16(ushort16 data, size_t offset, __global ushort *p);
05102 void __attribute__((overloadable)) vstore16(int16 data, size_t offset, __global int *p);
05103 void __attribute__((overloadable)) vstore16(uint16 data, size_t offset, __global uint *p);
05104 void __attribute__((overloadable)) vstore16(long16 data, size_t offset, __global long *p);
05105 void __attribute__((overloadable)) vstore16(ulong16 data, size_t offset, __global ulong *p);
05106 void __attribute__((overloadable)) vstore16(float16 data, size_t offset, __global float *p);
05107 void __attribute__((overloadable)) vstore2(char2 data, size_t offset, __local char *p);
05108 void __attribute__((overloadable)) vstore2(uchar2 data, size_t offset, __local uchar *p);
05109 void __attribute__((overloadable)) vstore2(short2 data, size_t offset, __local short *p);
05110 void __attribute__((overloadable)) vstore2(ushort2 data, size_t offset, __local ushort *p);
05111 void __attribute__((overloadable)) vstore2(int2 data, size_t offset, __local int *p);
05112 void __attribute__((overloadable)) vstore2(uint2 data, size_t offset, __local uint *p);
05113 void __attribute__((overloadable)) vstore2(long2 data, size_t offset, __local long *p);
05114 void __attribute__((overloadable)) vstore2(ulong2 data, size_t offset, __local ulong *p);
05115 void __attribute__((overloadable)) vstore2(float2 data, size_t offset, __local float *p);
05116 void __attribute__((overloadable)) vstore3(char3 data, size_t offset, __local char *p);
05117 void __attribute__((overloadable)) vstore3(uchar3 data, size_t offset, __local uchar *p);
05118 void __attribute__((overloadable)) vstore3(short3 data, size_t offset, __local short *p);
05119 void __attribute__((overloadable)) vstore3(ushort3 data, size_t offset, __local ushort *p);
05120 void __attribute__((overloadable)) vstore3(int3 data, size_t offset, __local int *p);
05121 void __attribute__((overloadable)) vstore3(uint3 data, size_t offset, __local uint *p);
05122 void __attribute__((overloadable)) vstore3(long3 data, size_t offset, __local long *p);
05123 void __attribute__((overloadable)) vstore3(ulong3 data, size_t offset, __local ulong *p);
05124 void __attribute__((overloadable)) vstore3(float3 data, size_t offset, __local float *p);
05125 void __attribute__((overloadable)) vstore4(char4 data, size_t offset, __local char *p);
05126 void __attribute__((overloadable)) vstore4(uchar4 data, size_t offset, __local uchar *p);
05127 void __attribute__((overloadable)) vstore4(short4 data, size_t offset, __local short *p);
05128 void __attribute__((overloadable)) vstore4(ushort4 data, size_t offset, __local ushort *p);
05129 void __attribute__((overloadable)) vstore4(int4 data, size_t offset, __local int *p);
05130 void __attribute__((overloadable)) vstore4(uint4 data, size_t offset, __local uint *p);
05131 void __attribute__((overloadable)) vstore4(long4 data, size_t offset, __local long *p);
05132 void __attribute__((overloadable)) vstore4(ulong4 data, size_t offset, __local ulong *p);
05133 void __attribute__((overloadable)) vstore4(float4 data, size_t offset, __local float *p);
05134 void __attribute__((overloadable)) vstore8(char8 data, size_t offset, __local char *p);
05135 void __attribute__((overloadable)) vstore8(uchar8 data, size_t offset, __local uchar *p);
05136 void __attribute__((overloadable)) vstore8(short8 data, size_t offset, __local short *p);
05137 void __attribute__((overloadable)) vstore8(ushort8 data, size_t offset, __local ushort *p);
05138 void __attribute__((overloadable)) vstore8(int8 data, size_t offset, __local int *p);
05139 void __attribute__((overloadable)) vstore8(uint8 data, size_t offset, __local uint *p);
05140 void __attribute__((overloadable)) vstore8(long8 data, size_t offset, __local long *p);
05141 void __attribute__((overloadable)) vstore8(ulong8 data, size_t offset, __local ulong *p);
05142 void __attribute__((overloadable)) vstore8(float8 data, size_t offset, __local float *p);
05143 void __attribute__((overloadable)) vstore16(char16 data, size_t offset, __local char *p);
05144 void __attribute__((overloadable)) vstore16(uchar16 data, size_t offset, __local uchar *p);
05145 void __attribute__((overloadable)) vstore16(short16 data, size_t offset, __local short *p);
05146 void __attribute__((overloadable)) vstore16(ushort16 data, size_t offset, __local ushort *p);
05147 void __attribute__((overloadable)) vstore16(int16 data, size_t offset, __local int *p);
05148 void __attribute__((overloadable)) vstore16(uint16 data, size_t offset, __local uint *p);
05149 void __attribute__((overloadable)) vstore16(long16 data, size_t offset, __local long *p);
05150 void __attribute__((overloadable)) vstore16(ulong16 data, size_t offset, __local ulong *p);
05151 void __attribute__((overloadable)) vstore16(float16 data, size_t offset, __local float *p);
05152 void __attribute__((overloadable)) vstore2(char2 data, size_t offset, __private char *p);
05153 void __attribute__((overloadable)) vstore2(uchar2 data, size_t offset, __private uchar *p);
05154 void __attribute__((overloadable)) vstore2(short2 data, size_t offset, __private short *p);
05155 void __attribute__((overloadable)) vstore2(ushort2 data, size_t offset, __private ushort *p);
05156 void __attribute__((overloadable)) vstore2(int2 data, size_t offset, __private int *p);
05157 void __attribute__((overloadable)) vstore2(uint2 data, size_t offset, __private uint *p);
05158 void __attribute__((overloadable)) vstore2(long2 data, size_t offset, __private long *p);
05159 void __attribute__((overloadable)) vstore2(ulong2 data, size_t offset, __private ulong *p);
05160 void __attribute__((overloadable)) vstore2(float2 data, size_t offset, __private float *p);
05161 void __attribute__((overloadable)) vstore3(char3 data, size_t offset, __private char *p);
05162 void __attribute__((overloadable)) vstore3(uchar3 data, size_t offset, __private uchar *p);
05163 void __attribute__((overloadable)) vstore3(short3 data, size_t offset, __private short *p);
05164 void __attribute__((overloadable)) vstore3(ushort3 data, size_t offset, __private ushort *p);
05165 void __attribute__((overloadable)) vstore3(int3 data, size_t offset, __private int *p);
05166 void __attribute__((overloadable)) vstore3(uint3 data, size_t offset, __private uint *p);
05167 void __attribute__((overloadable)) vstore3(long3 data, size_t offset, __private long *p);
05168 void __attribute__((overloadable)) vstore3(ulong3 data, size_t offset, __private ulong *p);
05169 void __attribute__((overloadable)) vstore3(float3 data, size_t offset, __private float *p);
05170 void __attribute__((overloadable)) vstore4(char4 data, size_t offset, __private char *p);
05171 void __attribute__((overloadable)) vstore4(uchar4 data, size_t offset, __private uchar *p);
05172 void __attribute__((overloadable)) vstore4(short4 data, size_t offset, __private short *p);
05173 void __attribute__((overloadable)) vstore4(ushort4 data, size_t offset, __private ushort *p);
05174 void __attribute__((overloadable)) vstore4(int4 data, size_t offset, __private int *p);
05175 void __attribute__((overloadable)) vstore4(uint4 data, size_t offset, __private uint *p);
05176 void __attribute__((overloadable)) vstore4(long4 data, size_t offset, __private long *p);
05177 void __attribute__((overloadable)) vstore4(ulong4 data, size_t offset, __private ulong *p);
05178 void __attribute__((overloadable)) vstore4(float4 data, size_t offset, __private float *p);
05179 void __attribute__((overloadable)) vstore8(char8 data, size_t offset, __private char *p);
05180 void __attribute__((overloadable)) vstore8(uchar8 data, size_t offset, __private uchar *p);
05181 void __attribute__((overloadable)) vstore8(short8 data, size_t offset, __private short *p);
05182 void __attribute__((overloadable)) vstore8(ushort8 data, size_t offset, __private ushort *p);
05183 void __attribute__((overloadable)) vstore8(int8 data, size_t offset, __private int *p);
05184 void __attribute__((overloadable)) vstore8(uint8 data, size_t offset, __private uint *p);
```

```
05185 void __attribute__((overloadable)) vstore8(long8 data, size_t offset, __private long *p);
05186 void __attribute__((overloadable)) vstore8(ulong8 data, size_t offset, __private ulong *p);
05187 void __attribute__((overloadable)) vstore8(float8 data, size_t offset, __private float *p);
05188 void __attribute__((overloadable)) vstore16(char16 data, size_t offset, __private char *p);
05189 void __attribute__((overloadable)) vstore16(uchar16 data, size_t offset, __private uchar *p);
05190 void __attribute__((overloadable)) vstore16(short16 data, size_t offset, __private short *p);
05191 void __attribute__((overloadable)) vstore16(ushort16 data, size_t offset, __private ushort *p);
05192 void __attribute__((overloadable)) vstore16(int16 data, size_t offset, __private int *p);
05193 void __attribute__((overloadable)) vstore16(uint16 data, size_t offset, __private uint *p);
05194 void __attribute__((overloadable)) vstore16(long16 data, size_t offset, __private long *p);
05195 void __attribute__((overloadable)) vstore16(ulong16 data, size_t offset, __private ulong *p);
05196 void __attribute__((overloadable)) vstore16(float16 data, size_t offset, __private float *p);
05197 void __attribute__((overloadable)) vstore2(double2 data, size_t offset, __global double *p);
05198 void __attribute__((overloadable)) vstore3(double3 data, size_t offset, __global double *p);
05199 void __attribute__((overloadable)) vstore4(double4 data, size_t offset, __global double *p);
05200 void __attribute__((overloadable)) vstore8(double8 data, size_t offset, __global double *p);
05201 void __attribute__((overloadable)) vstore16(double16 data, size_t offset, __global double *p);
05202 void __attribute__((overloadable)) vstore2(double2 data, size_t offset, __local double *p);
05203 void __attribute__((overloadable)) vstore3(double3 data, size_t offset, __local double *p);
05204 void __attribute__((overloadable)) vstore4(double4 data, size_t offset, __local double *p);
05205 void __attribute__((overloadable)) vstore8(double8 data, size_t offset, __local double *p);
05206 void __attribute__((overloadable)) vstore16(double16 data, size_t offset, __local double *p);
05207 void __attribute__((overloadable)) vstore2(double2 data, size_t offset, __private double *p);
05208 void __attribute__((overloadable)) vstore3(double3 data, size_t offset, __private double *p);
05209 void __attribute__((overloadable)) vstore4(double4 data, size_t offset, __private double *p);
05210 void __attribute__((overloadable)) vstore8(double8 data, size_t offset, __private double *p);
05211 void __attribute__((overloadable)) vstore16(double16 data, size_t offset, __private double *p);
05212 //void __attribute__((overloadable)) vstore(half data, size_t offset, __global half *p);
05213 //void __attribute__((overloadable)) vstore2(half2 data, size_t offset, __global half *p);
05214 //void __attribute__((overloadable)) vstore3(half3 data, size_t offset, __global half *p);
05215 //void __attribute__((overloadable)) vstore4(half4 data, size_t offset, __global half *p);
05216 //void __attribute__((overloadable)) vstore8(half8 data, size_t offset, __global half *p);
05217 //void __attribute__((overloadable)) vstore16(half16 data, size_t offset, __global half *p);
05218 //void __attribute__((overloadable)) vstore(half data, size_t offset, __local half *p);
05219 //void __attribute__((overloadable)) vstore2(half2 data, size_t offset, __local half *p);
05220 //void __attribute__((overloadable)) vstore3(half3 data, size_t offset, __local half *p);
05221 //void __attribute__((overloadable)) vstore4(half4 data, size_t offset, __local half *p);
05222 //void __attribute__((overloadable)) vstore8(half8 data, size_t offset, __local half *p);
05223 //void __attribute__((overloadable)) vstore16(half16 data, size_t offset, __local half *p);
05224 //void __attribute__((overloadable)) vstore(half data, size_t offset, __private half *p);
05225 //void __attribute__((overloadable)) vstore2(half2 data, size_t offset, __private half *p);
05226 //void __attribute__((overloadable)) vstore3(half3 data, size_t offset, __private half *p);
05227 //void __attribute__((overloadable)) vstore4(half4 data, size_t offset, __private half *p);
05228 //void __attribute__((overloadable)) vstore8(half8 data, size_t offset, __private half *p);
05229 //void __attribute__((overloadable)) vstore16(half16 data, size_t offset, __private half *p);
05230
05231
05232 /**
05233 * Read sizeof (half) bytes of data from address
05234 * (p + offset).  The data read is interpreted as a
05235 * half value.  The half value is converted to a
05236 * float value and the float value is returned.
05237 * The read address computed as (p + offset)
05238 * must be 16-bit aligned.
05239 */
05240 float __attribute__((overloadable)) vload_half(size_t offset, const __global half *p);
05241 float __attribute__((overloadable)) vload_half(size_t offset, const __local half *p);
05242 float __attribute__((overloadable)) vload_half(size_t offset, const __private half *p);
05243 float __attribute__((overloadable)) vload_half(size_t offset, const __constant half *p);
05244
05245 /**
05246 * Read sizeof (halfn) bytes of data from address
05247 * (p + (offset * n)).  The data read is interpreted
05248 * as a halfn value.  The halfn value read is
05249 * converted to a floatn value and the floatn
05250 * value is returned.  The read address computed
05251 * as (p + (offset * n)) must be 16-bit aligned.
05252 */
05253 float2 __attribute__((overloadable)) vload_half2(size_t offset, const __global half *p);
05254 float3 __attribute__((overloadable)) vload_half3(size_t offset, const __global half *p);
05255 float4 __attribute__((overloadable)) vload_half4(size_t offset, const __global half *p);
05256 float8 __attribute__((overloadable)) vload_half8(size_t offset, const __global half *p);
05257 float16 __attribute__((overloadable)) vload_half16(size_t offset, const __global half *p);
05258 float2 __attribute__((overloadable)) vload_half2(size_t offset, const __local half *p);
05259 float3 __attribute__((overloadable)) vload_half3(size_t offset, const __local half *p);
05260 float4 __attribute__((overloadable)) vload_half4(size_t offset, const __local half *p);
05261 float8 __attribute__((overloadable)) vload_half8(size_t offset, const __local half *p);
05262 float16 __attribute__((overloadable)) vload_half16(size_t offset, const __local half *p);
05263 float2 __attribute__((overloadable)) vload_half2(size_t offset, const __private half *p);
05264 float3 __attribute__((overloadable)) vload_half3(size_t offset, const __private half *p);
05265 float4 __attribute__((overloadable)) vload_half4(size_t offset, const __private half *p);
05266 float8 __attribute__((overloadable)) vload_half8(size_t offset, const __private half *p);
05267 float16 __attribute__((overloadable)) vload_half16(size_t offset, const __private half *p);
05268 float2 __attribute__((overloadable)) vload_half2(size_t offset, const __constant half *p);
05269 float3 __attribute__((overloadable)) vload_half3(size_t offset, const __constant half *p);
05270 float4 __attribute__((overloadable)) vload_half4(size_t offset, const __constant half *p);
05271 float8 __attribute__((overloadable)) vload_half8(size_t offset, const __constant half *p);
```

```
05272 float16 __attribute__((overloadable)) vload_half16(size_t offset, const __constant half *p);
05273
05274 /**
05275 * The float value given by data is first
05276 * converted to a half value using the appropriate
05277 * rounding mode.  The half value is then written
05278 * to address computed as (p + offset).  The
05279 * address computed as (p + offset) must be 16-
05280 * bit aligned.
05281 * vstore_half use the current rounding mode.
05282 * The default current rounding mode is round to
05283 * nearest even.
05284 */
05285 void __attribute__((overloadable)) vstore_half(float data, size_t offset, __global half *p);
05286 void __attribute__((overloadable)) vstore_half_rte(float data, size_t offset, __global half *p);
05287 void __attribute__((overloadable)) vstore_half_rtz(float data, size_t offset, __global half *p);
05288 void __attribute__((overloadable)) vstore_half_rtp(float data, size_t offset, __global half *p);
05289 void __attribute__((overloadable)) vstore_half_rtn(float data, size_t offset, __global half *p);
05290 void __attribute__((overloadable)) vstore_half(float data, size_t offset, __local half *p);
05291 void __attribute__((overloadable)) vstore_half_rte(float data, size_t offset, __local half *p);
05292 void __attribute__((overloadable)) vstore_half_rtz(float data, size_t offset, __local half *p);
05293 void __attribute__((overloadable)) vstore_half_rtp(float data, size_t offset, __local half *p);
05294 void __attribute__((overloadable)) vstore_half_rtn(float data, size_t offset, __local half *p);
05295 void __attribute__((overloadable)) vstore_half(float data, size_t offset, __private half *p);
05296 void __attribute__((overloadable)) vstore_half_rte(float data, size_t offset, __private half *p);
05297 void __attribute__((overloadable)) vstore_half_rtz(float data, size_t offset, __private half *p);
05298 void __attribute__((overloadable)) vstore_half_rtp(float data, size_t offset, __private half *p);
05299 void __attribute__((overloadable)) vstore_half_rtn(float data, size_t offset, __private half *p);
05300 void __attribute__((overloadable)) vstore_half(double data, size_t offset, __global half *p);
05301 void __attribute__((overloadable)) vstore_half_rte(double data, size_t offset, __global half *p);
05302 void __attribute__((overloadable)) vstore_half_rtz(double data, size_t offset, __global half *p);
05303 void __attribute__((overloadable)) vstore_half_rtp(double data, size_t offset, __global half *p);
05304 void __attribute__((overloadable)) vstore_half_rtn(double data, size_t offset, __global half *p);
05305 void __attribute__((overloadable)) vstore_half(double data, size_t offset, __local half *p);
05306 void __attribute__((overloadable)) vstore_half_rte(double data, size_t offset, __local half *p);
05307 void __attribute__((overloadable)) vstore_half_rtz(double data, size_t offset, __local half *p);
05308 void __attribute__((overloadable)) vstore_half_rtp(double data, size_t offset, __local half *p);
05309 void __attribute__((overloadable)) vstore_half_rtn(double data, size_t offset, __local half *p);
05310 void __attribute__((overloadable)) vstore_half(double data, size_t offset, __private half *p);
05311 void __attribute__((overloadable)) vstore_half_rte(double data, size_t offset, __private half *p);
05312 void __attribute__((overloadable)) vstore_half_rtz(double data, size_t offset, __private half *p);
05313 void __attribute__((overloadable)) vstore_half_rtp(double data, size_t offset, __private half *p);
05314 void __attribute__((overloadable)) vstore_half_rtn(double data, size_t offset, __private half *p);
05315
05316 /**
05317 * The floatn value given by data is converted to
05318 * a halfn value using the appropriate rounding
05319 * mode.  The halfn value is then written to
05320 * address computed as (p + (offset * n)).  The
05321 * address computed as (p + (offset * n)) must be
05322 * 16-bit aligned.
05323 * vstore_halfn uses the current rounding mode.
05324 * The default current rounding mode is round to
05325 * nearest even.
05326 */
05327 void __attribute__((overloadable)) vstore_half2(float2 data, size_t offset, __global half *p);
05328 void __attribute__((overloadable)) vstore_half3(float3 data, size_t offset, __global half *p);
05329 void __attribute__((overloadable)) vstore_half4(float4 data, size_t offset, __global half *p);
05330 void __attribute__((overloadable)) vstore_half8(float8 data, size_t offset, __global half *p);
05331 void __attribute__((overloadable)) vstore_half16(float16 data, size_t offset, __global half *p);
05332 void __attribute__((overloadable)) vstore_half2_rte(float2 data, size_t offset, __global half *p);
05333 void __attribute__((overloadable)) vstore_half3_rte(float3 data, size_t offset, __global half *p);
05334 void __attribute__((overloadable)) vstore_half4_rte(float4 data, size_t offset, __global half *p);
05335 void __attribute__((overloadable)) vstore_half8_rte(float8 data, size_t offset, __global half *p);
05336 void __attribute__((overloadable)) vstore_half16_rte(float16 data, size_t offset, __global half *p);
05337 void __attribute__((overloadable)) vstore_half2_rtz(float2 data, size_t offset, __global half *p);
05338 void __attribute__((overloadable)) vstore_half3_rtz(float3 data, size_t offset, __global half *p);
05339 void __attribute__((overloadable)) vstore_half4_rtz(float4 data, size_t offset, __global half *p);
05340 void __attribute__((overloadable)) vstore_half8_rtz(float8 data, size_t offset, __global half *p);
05341 void __attribute__((overloadable)) vstore_half16_rtz(float16 data, size_t offset, __global half *p);
05342 void __attribute__((overloadable)) vstore_half2_rtp(float2 data, size_t offset, __global half *p);
05343 void __attribute__((overloadable)) vstore_half3_rtp(float3 data, size_t offset, __global half *p);
05344 void __attribute__((overloadable)) vstore_half4_rtp(float4 data, size_t offset, __global half *p);
05345 void __attribute__((overloadable)) vstore_half8_rtp(float8 data, size_t offset, __global half *p);
05346 void __attribute__((overloadable)) vstore_half16_rtp(float16 data, size_t offset, __global half *p);
05347 void __attribute__((overloadable)) vstore_half2_rtn(float2 data, size_t offset, __global half *p);
05348 void __attribute__((overloadable)) vstore_half3_rtn(float3 data, size_t offset, __global half *p);
05349 void __attribute__((overloadable)) vstore_half4_rtn(float4 data, size_t offset, __global half *p);
05350 void __attribute__((overloadable)) vstore_half8_rtn(float8 data, size_t offset, __global half *p);
05351 void __attribute__((overloadable)) vstore_half16_rtn(float16 data, size_t offset, __global half *p);
05352 void __attribute__((overloadable)) vstore_half2(float2 data, size_t offset, __local half *p);
05353 void __attribute__((overloadable)) vstore_half3(float3 data, size_t offset, __local half *p);
05354 void __attribute__((overloadable)) vstore_half4(float4 data, size_t offset, __local half *p);
05355 void __attribute__((overloadable)) vstore_half8(float8 data, size_t offset, __local half *p);
05356 void __attribute__((overloadable)) vstore_half16(float16 data, size_t offset, __local half *p);
05357 void __attribute__((overloadable)) vstore_half2_rte(float2 data, size_t offset, __local half *p);
05358 void __attribute__((overloadable)) vstore_half3_rte(float3 data, size_t offset, __local half *p);
```

```
05359 void __attribute__((overloadable)) vstore_half4_rte(float4 data, size_t offset, __local half *p);
05360 void __attribute__((overloadable)) vstore_half8_rte(float8 data, size_t offset, __local half *p);
05361 void __attribute__((overloadable)) vstore_half16_rte(float16 data, size_t offset, __local half *p);
05362 void __attribute__((overloadable)) vstore_half2_rtz(float2 data, size_t offset, __local half *p);
05363 void __attribute__((overloadable)) vstore_half3_rtz(float3 data, size_t offset, __local half *p);
05364 void __attribute__((overloadable)) vstore_half4_rtz(float4 data, size_t offset, __local half *p);
05365 void __attribute__((overloadable)) vstore_half8_rtz(float8 data, size_t offset, __local half *p);
05366 void __attribute__((overloadable)) vstore_half16_rtz(float16 data, size_t offset, __local half *p);
05367 void __attribute__((overloadable)) vstore_half2_rtp(float2 data, size_t offset, __local half *p);
05368 void __attribute__((overloadable)) vstore_half3_rtp(float3 data, size_t offset, __local half *p);
05369 void __attribute__((overloadable)) vstore_half4_rtp(float4 data, size_t offset, __local half *p);
05370 void __attribute__((overloadable)) vstore_half8_rtp(float8 data, size_t offset, __local half *p);
05371 void __attribute__((overloadable)) vstore_half16_rtp(float16 data, size_t offset, __local half *p);
05372 void __attribute__((overloadable)) vstore_half2_rtn(float2 data, size_t offset, __local half *p);
05373 void __attribute__((overloadable)) vstore_half3_rtn(float3 data, size_t offset, __local half *p);
05374 void __attribute__((overloadable)) vstore_half4_rtn(float4 data, size_t offset, __local half *p);
05375 void __attribute__((overloadable)) vstore_half8_rtn(float8 data, size_t offset, __local half *p);
05376 void __attribute__((overloadable)) vstore_half16_rtn(float16 data, size_t offset, __local half *p);
05377 void __attribute__((overloadable)) vstore_half2(float2 data, size_t offset, __private half *p);
05378 void __attribute__((overloadable)) vstore_half3(float3 data, size_t offset, __private half *p);
05379 void __attribute__((overloadable)) vstore_half4(float4 data, size_t offset, __private half *p);
05380 void __attribute__((overloadable)) vstore_half8(float8 data, size_t offset, __private half *p);
05381 void __attribute__((overloadable)) vstore_half16(float16 data, size_t offset, __private half *p);
05382 void __attribute__((overloadable)) vstore_half2_rte(float2 data, size_t offset, __private half *p);
05383 void __attribute__((overloadable)) vstore_half3_rte(float3 data, size_t offset, __private half *p);
05384 void __attribute__((overloadable)) vstore_half4_rte(float4 data, size_t offset, __private half *p);
05385 void __attribute__((overloadable)) vstore_half8_rte(float8 data, size_t offset, __private half *p);
05386 void __attribute__((overloadable)) vstore_half16_rte(float16 data, size_t offset, __private half *p);
05387 void __attribute__((overloadable)) vstore_half2_rtz(float2 data, size_t offset, __private half *p);
05388 void __attribute__((overloadable)) vstore_half3_rtz(float3 data, size_t offset, __private half *p);
05389 void __attribute__((overloadable)) vstore_half4_rtz(float4 data, size_t offset, __private half *p);
05390 void __attribute__((overloadable)) vstore_half8_rtz(float8 data, size_t offset, __private half *p);
05391 void __attribute__((overloadable)) vstore_half16_rtz(float16 data, size_t offset, __private half *p);
05392 void __attribute__((overloadable)) vstore_half2_rtp(float2 data, size_t offset, __private half *p);
05393 void __attribute__((overloadable)) vstore_half3_rtp(float3 data, size_t offset, __private half *p);
05394 void __attribute__((overloadable)) vstore_half4_rtp(float4 data, size_t offset, __private half *p);
05395 void __attribute__((overloadable)) vstore_half8_rtp(float8 data, size_t offset, __private half *p);
05396 void __attribute__((overloadable)) vstore_half16_rtp(float16 data, size_t offset, __private half *p);
05397 void __attribute__((overloadable)) vstore_half2_rtn(float2 data, size_t offset, __private half *p);
05398 void __attribute__((overloadable)) vstore_half3_rtn(float3 data, size_t offset, __private half *p);
05399 void __attribute__((overloadable)) vstore_half4_rtn(float4 data, size_t offset, __private half *p);
05400 void __attribute__((overloadable)) vstore_half8_rtn(float8 data, size_t offset, __private half *p);
05401 void __attribute__((overloadable)) vstore_half16_rtn(float16 data, size_t offset, __private half *p);
05402 void __attribute__((overloadable)) vstore_half2(double2 data, size_t offset, __global half *p);
05403 void __attribute__((overloadable)) vstore_half3(double3 data, size_t offset, __global half *p);
05404 void __attribute__((overloadable)) vstore_half4(double4 data, size_t offset, __global half *p);
05405 void __attribute__((overloadable)) vstore_half8(double8 data, size_t offset, __global half *p);
05406 void __attribute__((overloadable)) vstore_half16(double16 data, size_t offset, __global half *p);
05407 void __attribute__((overloadable)) vstore_half2_rte(double2 data, size_t offset, __global half *p);
05408 void __attribute__((overloadable)) vstore_half3_rte(double3 data, size_t offset, __global half *p);
05409 void __attribute__((overloadable)) vstore_half4_rte(double4 data, size_t offset, __global half *p);
05410 void __attribute__((overloadable)) vstore_half8_rte(double8 data, size_t offset, __global half *p);
05411 void __attribute__((overloadable)) vstore_half16_rte(double16 data, size_t offset, __global half *p);
05412 void __attribute__((overloadable)) vstore_half2_rtz(double2 data, size_t offset, __global half *p);
05413 void __attribute__((overloadable)) vstore_half3_rtz(double3 data, size_t offset, __global half *p);
05414 void __attribute__((overloadable)) vstore_half4_rtz(double4 data, size_t offset, __global half *p);
05415 void __attribute__((overloadable)) vstore_half8_rtz(double8 data, size_t offset, __global half *p);
05416 void __attribute__((overloadable)) vstore_half16_rtz(double16 data, size_t offset, __global half *p);
05417 void __attribute__((overloadable)) vstore_half2_rtp(double2 data, size_t offset, __global half *p);
05418 void __attribute__((overloadable)) vstore_half3_rtp(double3 data, size_t offset, __global half *p);
05419 void __attribute__((overloadable)) vstore_half4_rtp(double4 data, size_t offset, __global half *p);
05420 void __attribute__((overloadable)) vstore_half8_rtp(double8 data, size_t offset, __global half *p);
05421 void __attribute__((overloadable)) vstore_half16_rtp(double16 data, size_t offset, __global half *p);
05422 void __attribute__((overloadable)) vstore_half2_rtn(double2 data, size_t offset, __global half *p);
05423 void __attribute__((overloadable)) vstore_half3_rtn(double3 data, size_t offset, __global half *p);
05424 void __attribute__((overloadable)) vstore_half4_rtn(double4 data, size_t offset, __global half *p);
05425 void __attribute__((overloadable)) vstore_half8_rtn(double8 data, size_t offset, __global half *p);
05426 void __attribute__((overloadable)) vstore_half16_rtn(double16 data, size_t offset, __global half *p);
05427 void __attribute__((overloadable)) vstore_half2(double2 data, size_t offset, __local half *p);
05428 void __attribute__((overloadable)) vstore_half3(double3 data, size_t offset, __local half *p);
05429 void __attribute__((overloadable)) vstore_half4(double4 data, size_t offset, __local half *p);
05430 void __attribute__((overloadable)) vstore_half8(double8 data, size_t offset, __local half *p);
05431 void __attribute__((overloadable)) vstore_half16(double16 data, size_t offset, __local half *p);
05432 void __attribute__((overloadable)) vstore_half2_rte(double2 data, size_t offset, __local half *p);
05433 void __attribute__((overloadable)) vstore_half3_rte(double3 data, size_t offset, __local half *p);
05434 void __attribute__((overloadable)) vstore_half4_rte(double4 data, size_t offset, __local half *p);
05435 void __attribute__((overloadable)) vstore_half8_rte(double8 data, size_t offset, __local half *p);
05436 void __attribute__((overloadable)) vstore_half16_rte(double16 data, size_t offset, __local half *p);
05437 void __attribute__((overloadable)) vstore_half2_rtz(double2 data, size_t offset, __local half *p);
05438 void __attribute__((overloadable)) vstore_half3_rtz(double3 data, size_t offset, __local half *p);
05439 void __attribute__((overloadable)) vstore_half4_rtz(double4 data, size_t offset, __local half *p);
05440 void __attribute__((overloadable)) vstore_half8_rtz(double8 data, size_t offset, __local half *p);
05441 void __attribute__((overloadable)) vstore_half16_rtz(double16 data, size_t offset, __local half *p);
05442 void __attribute__((overloadable)) vstore_half2_rtp(double2 data, size_t offset, __local half *p);
05443 void __attribute__((overloadable)) vstore_half3_rtp(double3 data, size_t offset, __local half *p);
05444 void __attribute__((overloadable)) vstore_half4_rtp(double4 data, size_t offset, __local half *p);
05445 void __attribute__((overloadable)) vstore_half8_rtp(double8 data, size_t offset, __local half *p);
```

```
05446 void __attribute__((overloadable)) vstore_half16_rtp(double16 data, size_t offset, __local half *p);
05447 void __attribute__((overloadable)) vstore_half2_rtn(double2 data, size_t offset, __local half *p);
05448 void __attribute__((overloadable)) vstore_half3_rtn(double3 data, size_t offset, __local half *p);
05449 void __attribute__((overloadable)) vstore_half4_rtn(double4 data, size_t offset, __local half *p);
05450 void __attribute__((overloadable)) vstore_half8_rtn(double8 data, size_t offset, __local half *p);
05451 void __attribute__((overloadable)) vstore_half16_rtn(double16 data, size_t offset, __local half *p);
05452 void __attribute__((overloadable)) vstore_half2(double2 data, size_t offset, __private half *p);
05453 void __attribute__((overloadable)) vstore_half3(double3 data, size_t offset, __private half *p);
05454 void __attribute__((overloadable)) vstore_half4(double4 data, size_t offset, __private half *p);
05455 void __attribute__((overloadable)) vstore_half8(double8 data, size_t offset, __private half *p);
05456 void __attribute__((overloadable)) vstore_half16(double16 data, size_t offset, __private half *p);
05457 void __attribute__((overloadable)) vstore_half2_rte(double2 data, size_t offset, __private half *p);
05458 void __attribute__((overloadable)) vstore_half3_rte(double3 data, size_t offset, __private half *p);
05459 void __attribute__((overloadable)) vstore_half4_rte(double4 data, size_t offset, __private half *p);
05460 void __attribute__((overloadable)) vstore_half8_rte(double8 data, size_t offset, __private half *p);
05461 void __attribute__((overloadable)) vstore_half16_rte(double16 data, size_t offset, __private half *p);
05462 void __attribute__((overloadable)) vstore_half2_rtz(double2 data, size_t offset, __private half *p);
05463 void __attribute__((overloadable)) vstore_half3_rtz(double3 data, size_t offset, __private half *p);
05464 void __attribute__((overloadable)) vstore_half4_rtz(double4 data, size_t offset, __private half *p);
05465 void __attribute__((overloadable)) vstore_half8_rtz(double8 data, size_t offset, __private half *p);
05466 void __attribute__((overloadable)) vstore_half16_rtz(double16 data, size_t offset, __private half *p);
05467 void __attribute__((overloadable)) vstore_half2_rtp(double2 data, size_t offset, __private half *p);
05468 void __attribute__((overloadable)) vstore_half3_rtp(double3 data, size_t offset, __private half *p);
05469 void __attribute__((overloadable)) vstore_half4_rtp(double4 data, size_t offset, __private half *p);
05470 void __attribute__((overloadable)) vstore_half8_rtp(double8 data, size_t offset, __private half *p);
05471 void __attribute__((overloadable)) vstore_half16_rtp(double16 data, size_t offset, __private half *p);
05472 void __attribute__((overloadable)) vstore_half2_rtn(double2 data, size_t offset, __private half *p);
05473 void __attribute__((overloadable)) vstore_half3_rtn(double3 data, size_t offset, __private half *p);
05474 void __attribute__((overloadable)) vstore_half4_rtn(double4 data, size_t offset, __private half *p);
05475 void __attribute__((overloadable)) vstore_half8_rtn(double8 data, size_t offset, __private half *p);
05476 void __attribute__((overloadable)) vstore_half16_rtn(double16 data, size_t offset, __private half *p);
05477
05478 /**
05479 * For n = 1, 2, 4, 8 and 16 read sizeof (halfn)
05480 * bytes of data from address (p + (offset * n)).
05481 * The data read is interpreted as a halfn value.
05482 * The halfn value read is converted to a floatn
05483 * value and the floatn value is returned.
05484 * The address computed as (p + (offset * n))
05485 * must be aligned to sizeof (halfn) bytes.
05486 * For n = 3, vloada_half3 reads a half3 from
05487 * address (p + (offset * 4)) and returns a float3.
05488 * The address computed as (p + (offset * 4))
05489 * must be aligned to sizeof (half) * 4 bytes.
05490 */
05491 float __attribute__((overloadable)) vloada_half(size_t offset, const __global half *p);
05492 float2 __attribute__((overloadable)) vloada_half2(size_t offset, const __global half *p);
05493 float3 __attribute__((overloadable)) vloada_half3(size_t offset, const __global half *p);
05494 float4 __attribute__((overloadable)) vloada_half4(size_t offset, const __global half *p);
05495 float8 __attribute__((overloadable)) vloada_half8(size_t offset, const __global half *p);
05496 float16 __attribute__((overloadable)) vloada_half16(size_t offset, const __global half *p);
05497 float __attribute__((overloadable)) vloada_half(size_t offset, const __local half *p);
05498 float2 __attribute__((overloadable)) vloada_half2(size_t offset, const __local half *p);
05499 float3 __attribute__((overloadable)) vloada_half3(size_t offset, const __local half *p);
05500 float4 __attribute__((overloadable)) vloada_half4(size_t offset, const __local half *p);
05501 float8 __attribute__((overloadable)) vloada_half8(size_t offset, const __local half *p);
05502 float16 __attribute__((overloadable)) vloada_half16(size_t offset, const __local half *p);
05503 float __attribute__((overloadable)) vloada_half(size_t offset, const __private half *p);
05504 float2 __attribute__((overloadable)) vloada_half2(size_t offset, const __private half *p);
05505 float3 __attribute__((overloadable)) vloada_half3(size_t offset, const __private half *p);
05506 float4 __attribute__((overloadable)) vloada_half4(size_t offset, const __private half *p);
05507 float8 __attribute__((overloadable)) vloada_half8(size_t offset, const __private half *p);
05508 float16 __attribute__((overloadable)) vloada_half16(size_t offset, const __private half *p);
05509 float __attribute__((overloadable)) vloada_half(size_t offset, const __constant half *p);
05510 float2 __attribute__((overloadable)) vloada_half2(size_t offset, const __constant half *p);
05511 float3 __attribute__((overloadable)) vloada_half3(size_t offset, const __constant half *p);
05512 float4 __attribute__((overloadable)) vloada_half4(size_t offset, const __constant half *p);
05513 float8 __attribute__((overloadable)) vloada_half8(size_t offset, const __constant half *p);
05514 float16 __attribute__((overloadable)) vloada_half16(size_t offset, const __constant half *p);
05515
05516 /**
05517 * The floatn value given by data is converted to
05518 * a halfn value using the appropriate rounding
05519 * mode.
05520 * For n = 1, 2, 4, 8 and 16, the halfn value is
05521 * written to the address computed as (p + (offset
05522 * * n)).  The address computed as (p + (offset *
05523 * n)) must be aligned to sizeof (halfn) bytes.
05524 * For n = 3, the half3 value is written to the
05525 * address computed as (p + (offset * 4)).  The
05526 * address computed as (p + (offset * 4)) must be
05527 * aligned to sizeof (half) * 4 bytes.
05528 * vstorea_halfn uses the current rounding
05529 * mode.  The default current rounding mode is
05530 * round to nearest even.
05531 */
05532 void __attribute__((overloadable)) vstorea_half(float data, size_t offset, __global half *p);
```

```
05533 void __attribute__((overloadable)) vstorea_half2(float2 data, size_t offset, __global half *p);
05534 void __attribute__((overloadable)) vstorea_half3(float3 data, size_t offset, __global half *p);
05535 void __attribute__((overloadable)) vstorea_half4(float4 data, size_t offset, __global half *p);
05536 void __attribute__((overloadable)) vstorea_half8(float8 data, size_t offset, __global half *p);
05537 void __attribute__((overloadable)) vstorea_half16(float16 data, size_t offset, __global half *p);
05538
05539 void __attribute__((overloadable)) vstorea_half_rte(float data, size_t offset, __global half *p);
05540 void __attribute__((overloadable)) vstorea_half2_rte(float2 data, size_t offset, __global half *p);
05541 void __attribute__((overloadable)) vstorea_half3_rte(float3 data, size_t offset, __global half *p);
05542 void __attribute__((overloadable)) vstorea_half4_rte(float4 data, size_t offset, __global half *p);
05543 void __attribute__((overloadable)) vstorea_half8_rte(float8 data, size_t offset, __global half *p);
05544 void __attribute__((overloadable)) vstorea_half16_rte(float16 data, size_t offset, __global half *p);
05545
05546 void __attribute__((overloadable)) vstorea_half_rtz(float data, size_t offset, __global half *p);
05547 void __attribute__((overloadable)) vstorea_half2_rtz(float2 data, size_t offset, __global half *p);
05548 void __attribute__((overloadable)) vstorea_half3_rtz(float3 data, size_t offset, __global half *p);
05549 void __attribute__((overloadable)) vstorea_half4_rtz(float4 data, size_t offset, __global half *p);
05550 void __attribute__((overloadable)) vstorea_half8_rtz(float8 data, size_t offset, __global half *p);
05551 void __attribute__((overloadable)) vstorea_half16_rtz(float16 data, size_t offset, __global half *p);
05552
05553 void __attribute__((overloadable)) vstorea_half_rtp(float data, size_t offset, __global half *p);
05554 void __attribute__((overloadable)) vstorea_half2_rtp(float2 data, size_t offset, __global half *p);
05555 void __attribute__((overloadable)) vstorea_half3_rtp(float3 data, size_t offset, __global half *p);
05556 void __attribute__((overloadable)) vstorea_half4_rtp(float4 data, size_t offset, __global half *p);
05557 void __attribute__((overloadable)) vstorea_half8_rtp(float8 data, size_t offset, __global half *p);
05558 void __attribute__((overloadable)) vstorea_half16_rtp(float16 data, size_t offset, __global half *p);
05559
05560 void __attribute__((overloadable)) vstorea_half_rtn(float data, size_t offset, __global half *p);
05561 void __attribute__((overloadable)) vstorea_half2_rtn(float2 data, size_t offset, __global half *p);
05562 void __attribute__((overloadable)) vstorea_half3_rtn(float3 data, size_t offset, __global half *p);
05563 void __attribute__((overloadable)) vstorea_half4_rtn(float4 data, size_t offset, __global half *p);
05564 void __attribute__((overloadable)) vstorea_half8_rtn(float8 data, size_t offset, __global half *p);
05565 void __attribute__((overloadable)) vstorea_half16_rtn(float16 data, size_t offset, __global half *p);
05566
05567 void __attribute__((overloadable)) vstorea_half(float data, size_t offset, __local half *p);
05568 void __attribute__((overloadable)) vstorea_half2(float2 data, size_t offset, __local half *p);
05569 void __attribute__((overloadable)) vstorea_half3(float3 data, size_t offset, __local half *p);
05570 void __attribute__((overloadable)) vstorea_half4(float4 data, size_t offset, __local half *p);
05571 void __attribute__((overloadable)) vstorea_half8(float8 data, size_t offset, __local half *p);
05572 void __attribute__((overloadable)) vstorea_half16(float16 data, size_t offset, __local half *p);
05573
05574 void __attribute__((overloadable)) vstorea_half_rte(float data, size_t offset, __local half *p);
05575 void __attribute__((overloadable)) vstorea_half2_rte(float2 data, size_t offset, __local half *p);
05576 void __attribute__((overloadable)) vstorea_half3_rte(float3 data, size_t offset, __local half *p);
05577 void __attribute__((overloadable)) vstorea_half4_rte(float4 data, size_t offset, __local half *p);
05578 void __attribute__((overloadable)) vstorea_half8_rte(float8 data, size_t offset, __local half *p);
05579 void __attribute__((overloadable)) vstorea_half16_rte(float16 data, size_t offset, __local half *p);
05580
05581 void __attribute__((overloadable)) vstorea_half_rtz(float data, size_t offset, __local half *p);
05582 void __attribute__((overloadable)) vstorea_half2_rtz(float2 data, size_t offset, __local half *p);
05583 void __attribute__((overloadable)) vstorea_half3_rtz(float3 data, size_t offset, __local half *p);
05584 void __attribute__((overloadable)) vstorea_half4_rtz(float4 data, size_t offset, __local half *p);
05585 void __attribute__((overloadable)) vstorea_half8_rtz(float8 data, size_t offset, __local half *p);
05586 void __attribute__((overloadable)) vstorea_half16_rtz(float16 data, size_t offset, __local half *p);
05587
05588 void __attribute__((overloadable)) vstorea_half_rtp(float data, size_t offset, __local half *p);
05589 void __attribute__((overloadable)) vstorea_half2_rtp(float2 data, size_t offset, __local half *p);
05590 void __attribute__((overloadable)) vstorea_half3_rtp(float3 data, size_t offset, __local half *p);
05591 void __attribute__((overloadable)) vstorea_half4_rtp(float4 data, size_t offset, __local half *p);
05592 void __attribute__((overloadable)) vstorea_half8_rtp(float8 data, size_t offset, __local half *p);
05593 void __attribute__((overloadable)) vstorea_half16_rtp(float16 data, size_t offset, __local half *p);
05594
05595 void __attribute__((overloadable)) vstorea_half_rtn(float data, size_t offset, __local half *p);
05596 void __attribute__((overloadable)) vstorea_half2_rtn(float2 data, size_t offset, __local half *p);
05597 void __attribute__((overloadable)) vstorea_half3_rtn(float3 data, size_t offset, __local half *p);
05598 void __attribute__((overloadable)) vstorea_half4_rtn(float4 data, size_t offset, __local half *p);
05599 void __attribute__((overloadable)) vstorea_half8_rtn(float8 data, size_t offset, __local half *p);
05600 void __attribute__((overloadable)) vstorea_half16_rtn(float16 data, size_t offset, __local half *p);
05601
05602 void __attribute__((overloadable)) vstorea_half(float data, size_t offset, __private half *p);
05603 void __attribute__((overloadable)) vstorea_half2(float2 data, size_t offset, __private half *p);
05604 void __attribute__((overloadable)) vstorea_half3(float3 data, size_t offset, __private half *p);
05605 void __attribute__((overloadable)) vstorea_half4(float4 data, size_t offset, __private half *p);
05606 void __attribute__((overloadable)) vstorea_half8(float8 data, size_t offset, __private half *p);
05607 void __attribute__((overloadable)) vstorea_half16(float16 data, size_t offset, __private half *p);
05608
05609 void __attribute__((overloadable)) vstorea_half_rte(float data, size_t offset, __private half *p);
05610 void __attribute__((overloadable)) vstorea_half2_rte(float2 data, size_t offset, __private half *p);
05611 void __attribute__((overloadable)) vstorea_half3_rte(float3 data, size_t offset, __private half *p);
05612 void __attribute__((overloadable)) vstorea_half4_rte(float4 data, size_t offset, __private half *p);
05613 void __attribute__((overloadable)) vstorea_half8_rte(float8 data, size_t offset, __private half *p);
05614 void __attribute__((overloadable)) vstorea_half16_rte(float16 data, size_t offset, __private half *p);
05615
05616 void __attribute__((overloadable)) vstorea_half_rtz(float data, size_t offset, __private half *p);
05617 void __attribute__((overloadable)) vstorea_half2_rtz(float2 data, size_t offset, __private half *p);
05618 void __attribute__((overloadable)) vstorea_half3_rtz(float3 data, size_t offset, __private half *p);
05619 void __attribute__((overloadable)) vstorea_half4_rtz(float4 data, size_t offset, __private half *p);
```

```
05620 void __attribute__((overloadable)) vstorea_half8_rtz(float8 data, size_t offset, __private half *p);
05621 void __attribute__((overloadable)) vstorea_half16_rtz(float16 data, size_t offset, __private half *p);
05622
05623 void __attribute__((overloadable)) vstorea_half_rtp(float data, size_t offset, __private half *p);
05624 void __attribute__((overloadable)) vstorea_half2_rtp(float2 data, size_t offset, __private half *p);
05625 void __attribute__((overloadable)) vstorea_half3_rtp(float3 data, size_t offset, __private half *p);
05626 void __attribute__((overloadable)) vstorea_half4_rtp(float4 data, size_t offset, __private half *p);
05627 void __attribute__((overloadable)) vstorea_half8_rtp(float8 data, size_t offset, __private half *p);
05628 void __attribute__((overloadable)) vstorea_half16_rtp(float16 data, size_t offset, __private half *p);
05629
05630 void __attribute__((overloadable)) vstorea_half_rtn(float data, size_t offset, __private half *p);
05631 void __attribute__((overloadable)) vstorea_half2_rtn(float2 data, size_t offset, __private half *p);
05632 void __attribute__((overloadable)) vstorea_half3_rtn(float3 data, size_t offset, __private half *p);
05633 void __attribute__((overloadable)) vstorea_half4_rtn(float4 data, size_t offset, __private half *p);
05634 void __attribute__((overloadable)) vstorea_half8_rtn(float8 data, size_t offset, __private half *p);
05635 void __attribute__((overloadable)) vstorea_half16_rtn(float16 data, size_t offset, __private half *p);
05636
05637 void __attribute__((overloadable)) vstorea_half(double data, size_t offset, __global half *p);
05638 void __attribute__((overloadable)) vstorea_half2(double2 data, size_t offset, __global half *p);
05639 void __attribute__((overloadable)) vstorea_half3(double3 data, size_t offset, __global half *p);
05640 void __attribute__((overloadable)) vstorea_half4(double4 data, size_t offset, __global half *p);
05641 void __attribute__((overloadable)) vstorea_half8(double8 data, size_t offset, __global half *p);
05642 void __attribute__((overloadable)) vstorea_half16(double16 data, size_t offset, __global half *p);
05643
05644 void __attribute__((overloadable)) vstorea_half_rte(double data, size_t offset, __global half *p);
05645 void __attribute__((overloadable)) vstorea_half2_rte(double2 data, size_t offset, __global half *p);
05646 void __attribute__((overloadable)) vstorea_half3_rte(double3 data, size_t offset, __global half *p);
05647 void __attribute__((overloadable)) vstorea_half4_rte(double4 data, size_t offset, __global half *p);
05648 void __attribute__((overloadable)) vstorea_half8_rte(double8 data, size_t offset, __global half *p);
05649 void __attribute__((overloadable)) vstorea_half16_rte(double16 data, size_t offset, __global half *p);
05650
05651 void __attribute__((overloadable)) vstorea_half_rtz(double data, size_t offset, __global half *p);
05652 void __attribute__((overloadable)) vstorea_half2_rtz(double2 data, size_t offset, __global half *p);
05653 void __attribute__((overloadable)) vstorea_half3_rtz(double3 data, size_t offset, __global half *p);
05654 void __attribute__((overloadable)) vstorea_half4_rtz(double4 data, size_t offset, __global half *p);
05655 void __attribute__((overloadable)) vstorea_half8_rtz(double8 data, size_t offset, __global half *p);
05656 void __attribute__((overloadable)) vstorea_half16_rtz(double16 data, size_t offset, __global half *p);
05657
05658 void __attribute__((overloadable)) vstorea_half_rtp(double data, size_t offset, __global half *p);
05659 void __attribute__((overloadable)) vstorea_half2_rtp(double2 data, size_t offset, __global half *p);
05660 void __attribute__((overloadable)) vstorea_half3_rtp(double3 data, size_t offset, __global half *p);
05661 void __attribute__((overloadable)) vstorea_half4_rtp(double4 data, size_t offset, __global half *p);
05662 void __attribute__((overloadable)) vstorea_half8_rtp(double8 data, size_t offset, __global half *p);
05663 void __attribute__((overloadable)) vstorea_half16_rtp(double16 data, size_t offset, __global half *p);
05664
05665 void __attribute__((overloadable)) vstorea_half_rtn(double data, size_t offset, __global half *p);
05666 void __attribute__((overloadable)) vstorea_half2_rtn(double2 data, size_t offset, __global half *p);
05667 void __attribute__((overloadable)) vstorea_half3_rtn(double3 data, size_t offset, __global half *p);
05668 void __attribute__((overloadable)) vstorea_half4_rtn(double4 data, size_t offset, __global half *p);
05669 void __attribute__((overloadable)) vstorea_half8_rtn(double8 data, size_t offset, __global half *p);
05670 void __attribute__((overloadable)) vstorea_half16_rtn(double16 data, size_t offset, __global half *p);
05671
05672 void __attribute__((overloadable)) vstorea_half(double data, size_t offset, __local half *p);
05673 void __attribute__((overloadable)) vstorea_half2(double2 data, size_t offset, __local half *p);
05674 void __attribute__((overloadable)) vstorea_half3(double3 data, size_t offset, __local half *p);
05675 void __attribute__((overloadable)) vstorea_half4(double4 data, size_t offset, __local half *p);
05676 void __attribute__((overloadable)) vstorea_half8(double8 data, size_t offset, __local half *p);
05677 void __attribute__((overloadable)) vstorea_half16(double16 data, size_t offset, __local half *p);
05678
05679 void __attribute__((overloadable)) vstorea_half_rte(double data, size_t offset, __local half *p);
05680 void __attribute__((overloadable)) vstorea_half2_rte(double2 data, size_t offset, __local half *p);
05681 void __attribute__((overloadable)) vstorea_half3_rte(double3 data, size_t offset, __local half *p);
05682 void __attribute__((overloadable)) vstorea_half4_rte(double4 data, size_t offset, __local half *p);
05683 void __attribute__((overloadable)) vstorea_half8_rte(double8 data, size_t offset, __local half *p);
05684 void __attribute__((overloadable)) vstorea_half16_rte(double16 data, size_t offset, __local half *p);
05685
05686 void __attribute__((overloadable)) vstorea_half_rtz(double data, size_t offset, __local half *p);
05687 void __attribute__((overloadable)) vstorea_half2_rtz(double2 data, size_t offset, __local half *p);
05688 void __attribute__((overloadable)) vstorea_half3_rtz(double3 data, size_t offset, __local half *p);
05689 void __attribute__((overloadable)) vstorea_half4_rtz(double4 data, size_t offset, __local half *p);
05690 void __attribute__((overloadable)) vstorea_half8_rtz(double8 data, size_t offset, __local half *p);
05691 void __attribute__((overloadable)) vstorea_half16_rtz(double16 data, size_t offset, __local half *p);
05692
05693 void __attribute__((overloadable)) vstorea_half_rtp(double data, size_t offset, __local half *p);
05694 void __attribute__((overloadable)) vstorea_half2_rtp(double2 data, size_t offset, __local half *p);
05695 void __attribute__((overloadable)) vstorea_half3_rtp(double3 data, size_t offset, __local half *p);
05696 void __attribute__((overloadable)) vstorea_half4_rtp(double4 data, size_t offset, __local half *p);
05697 void __attribute__((overloadable)) vstorea_half8_rtp(double8 data, size_t offset, __local half *p);
05698 void __attribute__((overloadable)) vstorea_half16_rtp(double16 data, size_t offset, __local half *p);
05699
05700 void __attribute__((overloadable)) vstorea_half_rtn(double data, size_t offset, __local half *p);
05701 void __attribute__((overloadable)) vstorea_half2_rtn(double2 data, size_t offset, __local half *p);
05702 void __attribute__((overloadable)) vstorea_half3_rtn(double3 data, size_t offset, __local half *p);
05703 void __attribute__((overloadable)) vstorea_half4_rtn(double4 data, size_t offset, __local half *p);
05704 void __attribute__((overloadable)) vstorea_half8_rtn(double8 data, size_t offset, __local half *p);
05705 void __attribute__((overloadable)) vstorea_half16_rtn(double16 data, size_t offset, __local half *p);
05706
```

```
05707 void __attribute__((overloadable)) vstorea_half(double data, size_t offset, __private half *p);
05708 void __attribute__((overloadable)) vstorea_half2(double2 data, size_t offset, __private half *p);
05709 void __attribute__((overloadable)) vstorea_half3(double3 data, size_t offset, __private half *p);
05710 void __attribute__((overloadable)) vstorea_half4(double4 data, size_t offset, __private half *p);
05711 void __attribute__((overloadable)) vstorea_half8(double8 data, size_t offset, __private half *p);
05712 void __attribute__((overloadable)) vstorea_half16(double16 data, size_t offset, __private half *p);
05713
05714 void __attribute__((overloadable)) vstorea_half_rte(double data, size_t offset, __private half *p);
05715 void __attribute__((overloadable)) vstorea_half2_rte(double2 data, size_t offset, __private half *p);
05716 void __attribute__((overloadable)) vstorea_half3_rte(double3 data, size_t offset, __private half *p);
05717 void __attribute__((overloadable)) vstorea_half4_rte(double4 data, size_t offset, __private half *p);
05718 void __attribute__((overloadable)) vstorea_half8_rte(double8 data, size_t offset, __private half *p);
05719 void __attribute__((overloadable)) vstorea_half16_rte(double16 data, size_t offset, __private half
      *p);
05720
05721 void __attribute__((overloadable)) vstorea_half_rtz(double data, size_t offset, __private half *p);
05722 void __attribute__((overloadable)) vstorea_half2_rtz(double2 data, size_t offset, __private half *p);
05723 void __attribute__((overloadable)) vstorea_half3_rtz(double3 data, size_t offset, __private half *p);
05724 void __attribute__((overloadable)) vstorea_half4_rtz(double4 data, size_t offset, __private half *p);
05725 void __attribute__((overloadable)) vstorea_half8_rtz(double8 data, size_t offset, __private half *p);
05726 void __attribute__((overloadable)) vstorea_half16_rtz(double16 data, size_t offset, __private half
      *p);
05727
05728 void __attribute__((overloadable)) vstorea_half_rtp(double data, size_t offset, __private half *p);
05729 void __attribute__((overloadable)) vstorea_half2_rtp(double2 data, size_t offset, __private half *p);
05730 void __attribute__((overloadable)) vstorea_half3_rtp(double3 data, size_t offset, __private half *p);
05731 void __attribute__((overloadable)) vstorea_half4_rtp(double4 data, size_t offset, __private half *p);
05732 void __attribute__((overloadable)) vstorea_half8_rtp(double8 data, size_t offset, __private half *p);
05733 void __attribute__((overloadable)) vstorea_half16_rtp(double16 data, size_t offset, __private half
      *p);
05734
05735 void __attribute__((overloadable)) vstorea_half_rtn(double data, size_t offset, __private half *p);
05736 void __attribute__((overloadable)) vstorea_half2_rtn(double2 data,size_t offset, __private half *p);
05737 void __attribute__((overloadable)) vstorea_half3_rtn(double3 data,size_t offset, __private half *p);
05738 void __attribute__((overloadable)) vstorea_half4_rtn(double4 data,size_t offset, __private half *p);
05739 void __attribute__((overloadable)) vstorea_half8_rtn(double8 data,size_t offset, __private half *p);
05740 void __attribute__((overloadable)) vstorea_half16_rtn(double16 data,size_t offset, __private half *p);
05741
05742 // Synchronization functions
05743
05744 /**
05745 * All work-items in a work-group executing the kernel
05746 * on a processor must execute this function before any
05747 * are allowed to continue execution beyond the barrier.
05748 * This function must be encountered by all work-items in
05749 * a work-group executing the kernel.
05750 * If barrier is inside a conditional statement, then all
05751 * work-items must enter the conditional if any work-item
05752 * enters the conditional statement and executes the
05753 * barrier.
05754 * If barrer is inside a loop, all work-items must execute
05755 * the barrier for each iteration of the loop before any are
05756 * allowed to continue execution beyond the barrier.
05757 * The barrier function also queues a memory fence
05758 * (reads and writes) to ensure correct ordering of
05759 * memory operations to local or global memory.
05760 * The flags argument specifies the memory address space
05761 * and can be set to a combination of the following literal
05762 * values.
05763 * CLK_LOCAL_MEM_FENCE - The barrier function
05764 * will either flush any variables stored in local memory
05765 * or queue a memory fence to ensure correct ordering of
05766 * memory operations to local memory.
05767 * CLK_GLOBAL_MEM_FENCE - The barrier function
05768 * will queue a memory fence to ensure correct ordering
05769 * of memory operations to global memory.  This can be
05770 * useful when work-items, for example, write to buffer or
05771 * image objects and then want to read the updated data.
05772 */
05773
05774 typedef uint cl_mem_fence_flags;
05775
05776 void __attribute__((overloadable)) barrier(cl_mem_fence_flags flags);
05777
05778 // Explicit memory fence functions
05779
05780 /**
05781 * Orders loads and stores of a work-item
05782 * executing a kernel.  This means that loads
05783 * and stores preceding the mem_fence will
05784 * be committed to memory before any loads
05785 * and stores following the mem_fence.
05786 * The flags argument specifies the memory
05787 * address space and can be set to a
05788 * combination of the following literal
05789 * values:
05790 * CLK_LOCAL_MEM_FENCE
```

```
05791 * CLK_GLOBAL_MEM_FENCE.
05792 */
05793 void __attribute__((overloadable)) mem_fence(cl_mem_fence_flags flags);
05794
05795 /**
05796 * Read memory barrier that orders only
05797 * loads.
05798 * The flags argument specifies the memory
05799 * address space and can be set to to a
05800 * combination of the following literal
05801 * values:
05802 * CLK_LOCAL_MEM_FENCE
05803 * CLK_GLOBAL_MEM_FENCE.
05804 */
05805 void __attribute__((overloadable)) read_mem_fence(cl_mem_fence_flags flags);
05806
05807 /**
05808 * Write memory barrier that orders only
05809 * stores.
05810 * The flags argument specifies the memory
05811 * address space and can be set to to a
05812 * combination of the following literal
05813 * values:
05814 * CLK_LOCAL_MEM_FENCE
05815 * CLK_GLOBAL_MEM_FENCE.
05816 */
05817 void __attribute__((overloadable)) write_mem_fence(cl_mem_fence_flags flags);
05818
05819 // Flag values for barrier, mem_fence, read_mem_fence, write_mem_fence
05820
05821 /**
05822 * Queue a memory fence to ensure correct
05823 * ordering of memory operations to local memory
05824 */
05825 #define CLK_LOCAL_MEM_FENCE    0x1
05826
05827 /**
05828 * Queue a memory fence to ensure correct
05829 * ordering of memory operations to global memory
05830 */
05831 #define CLK_GLOBAL_MEM_FENCE   0x2
05832
05833
05834 // Async copies from global to local memory, local to global memory, and prefetch
05835
05836 /**
05837 * event_t async_work_group_copy (
05838 * __global gentype *dst,
05839 * const __local gentype *src,
05840 * size_t num_elements,
05841 * event_t event)
05842 * Perform an async copy of num_elements
05843 * gentype elements from src to dst.  The async
05844 * copy is performed by all work-items in a workgroup
05845 * and this built-in function must therefore
05846 * be encountered by all work-items in a workgroup
05847 * executing the kernel with the same
05848 * argument values; otherwise the results are
05849 * undefined.
05850 * Returns an event object that can be used by
05851 * wait_group_events to wait for the async copy
05852 * to finish.  The event argument can also be used
05853 * to associate the async_work_group_copy with
05854 * a previous async copy allowing an event to be
05855 * shared by multiple async copies; otherwise event
05856 * should be zero.
05857 * If event argument is non-zero, the event object
05858 * supplied in event argument will be returned.
05859 * This function does not perform any implicit
05860 * synchronization of source data such as using a
05861 * barrier before performing the copy.
05862 */
05863 event_t __attribute__((overloadable)) async_work_group_copy(__local char *dst, const __global char
      *src, size_t num_elements, event_t event);
05864 event_t __attribute__((overloadable)) async_work_group_copy(__local uchar *dst, const __global uchar
      *src, size_t num_elements, event_t event);
05865 event_t __attribute__((overloadable)) async_work_group_copy(__local short *dst, const __global short
      *src, size_t num_elements, event_t event);
05866 event_t __attribute__((overloadable)) async_work_group_copy(__local ushort *dst, const __global ushort
      *src, size_t num_elements, event_t event);
05867 event_t __attribute__((overloadable)) async_work_group_copy(__local int *dst, const __global int *src,
      size_t num_elements, event_t event);
05868 event_t __attribute__((overloadable)) async_work_group_copy(__local uint *dst, const __global uint
      *src, size_t num_elements, event_t event);
05869 event_t __attribute__((overloadable)) async_work_group_copy(__local long *dst, const __global long
      *src, size_t num_elements, event_t event);
05870 event_t __attribute__((overloadable)) async_work_group_copy(__local ulong *dst, const __global ulong
```

```
      *src, size_t num_elements, event_t event);
05871 event_t __attribute__((overloadable)) async_work_group_copy(__local float *dst, const __global float
      *src, size_t num_elements, event_t event);
05872 event_t __attribute__((overloadable)) async_work_group_copy(__local char2 *dst, const __global char2
      *src, size_t num_elements, event_t event);
05873 event_t __attribute__((overloadable)) async_work_group_copy(__local uchar2 *dst, const __global uchar2
      *src, size_t num_elements, event_t event);
05874 event_t __attribute__((overloadable)) async_work_group_copy(__local short2 *dst, const __global short2
      *src, size_t num_elements, event_t event);
05875 event_t __attribute__((overloadable)) async_work_group_copy(__local ushort2 *dst, const __global
      ushort2 *src, size_t num_elements, event_t event);
05876 event_t __attribute__((overloadable)) async_work_group_copy(__local int2 *dst, const __global int2
      *src, size_t num_elements, event_t event);
05877 event_t __attribute__((overloadable)) async_work_group_copy(__local uint2 *dst, const __global uint2
      *src, size_t num_elements, event_t event);
05878 event_t __attribute__((overloadable)) async_work_group_copy(__local long2 *dst, const __global long2
      *src, size_t num_elements, event_t event);
05879 event_t __attribute__((overloadable)) async_work_group_copy(__local ulong2 *dst, const __global ulong2
      *src, size_t num_elements, event_t event);
05880 event_t __attribute__((overloadable)) async_work_group_copy(__local float2 *dst, const __global float2
      *src, size_t num_elements, event_t event);
05881 event_t __attribute__((overloadable)) async_work_group_copy(__local char3 *dst, const __global char3
      *src, size_t num_elements, event_t event);
05882 event_t __attribute__((overloadable)) async_work_group_copy(__local uchar3 *dst, const __global uchar3
      *src, size_t num_elements, event_t event);
05883 event_t __attribute__((overloadable)) async_work_group_copy(__local short3 *dst, const __global short3
      *src, size_t num_elements, event_t event);
05884 event_t __attribute__((overloadable)) async_work_group_copy(__local ushort3 *dst, const __global
      ushort3 *src, size_t num_elements, event_t event);
05885 event_t __attribute__((overloadable)) async_work_group_copy(__local int3 *dst, const __global int3
      *src, size_t num_elements, event_t event);
05886 event_t __attribute__((overloadable)) async_work_group_copy(__local uint3 *dst, const __global uint3
      *src, size_t num_elements, event_t event);
05887 event_t __attribute__((overloadable)) async_work_group_copy(__local long3 *dst, const __global long3
      *src, size_t num_elements, event_t event);
05888 event_t __attribute__((overloadable)) async_work_group_copy(__local ulong3 *dst, const __global ulong3
      *src, size_t num_elements, event_t event);
05889 event_t __attribute__((overloadable)) async_work_group_copy(__local float3 *dst, const __global float3
      *src, size_t num_elements, event_t event);
05890 event_t __attribute__((overloadable)) async_work_group_copy(__local char4 *dst, const __global char4
      *src, size_t num_elements, event_t event);
05891 event_t __attribute__((overloadable)) async_work_group_copy(__local uchar4 *dst, const __global uchar4
      *src, size_t num_elements, event_t event);
05892 event_t __attribute__((overloadable)) async_work_group_copy(__local short4 *dst, const __global short4
      *src, size_t num_elements, event_t event);
05893 event_t __attribute__((overloadable)) async_work_group_copy(__local ushort4 *dst, const __global
      ushort4 *src, size_t num_elements, event_t event);
05894 event_t __attribute__((overloadable)) async_work_group_copy(__local int4 *dst, const __global int4
      *src, size_t num_elements, event_t event);
05895 event_t __attribute__((overloadable)) async_work_group_copy(__local uint4 *dst, const __global uint4
      *src, size_t num_elements, event_t event);
05896 event_t __attribute__((overloadable)) async_work_group_copy(__local long4 *dst, const __global long4
      *src, size_t num_elements, event_t event);
05897 event_t __attribute__((overloadable)) async_work_group_copy(__local ulong4 *dst, const __global ulong4
      *src, size_t num_elements, event_t event);
05898 event_t __attribute__((overloadable)) async_work_group_copy(__local float4 *dst, const __global float4
      *src, size_t num_elements, event_t event);
05899 event_t __attribute__((overloadable)) async_work_group_copy(__local char8 *dst, const __global char8
      *src, size_t num_elements, event_t event);
05900 event_t __attribute__((overloadable)) async_work_group_copy(__local uchar8 *dst, const __global uchar8
      *src, size_t num_elements, event_t event);
05901 event_t __attribute__((overloadable)) async_work_group_copy(__local short8 *dst, const __global short8
      *src, size_t num_elements, event_t event);
05902 event_t __attribute__((overloadable)) async_work_group_copy(__local ushort8 *dst, const __global
      ushort8 *src, size_t num_elements, event_t event);
05903 event_t __attribute__((overloadable)) async_work_group_copy(__local int8 *dst, const __global int8
      *src, size_t num_elements, event_t event);
05904 event_t __attribute__((overloadable)) async_work_group_copy(__local uint8 *dst, const __global uint8
      *src, size_t num_elements, event_t event);
05905 event_t __attribute__((overloadable)) async_work_group_copy(__local long8 *dst, const __global long8
      *src, size_t num_elements, event_t event);
05906 event_t __attribute__((overloadable)) async_work_group_copy(__local ulong8 *dst, const __global ulong8
      *src, size_t num_elements, event_t event);
05907 event_t __attribute__((overloadable)) async_work_group_copy(__local float8 *dst, const __global float8
      *src, size_t num_elements, event_t event);
05908 event_t __attribute__((overloadable)) async_work_group_copy(__local char16 *dst, const __global char16
      *src, size_t num_elements, event_t event);
05909 event_t __attribute__((overloadable)) async_work_group_copy(__local uchar16 *dst, const __global
      uchar16 *src, size_t num_elements, event_t event);
05910 event_t __attribute__((overloadable)) async_work_group_copy(__local short16 *dst, const __global
      short16 *src, size_t num_elements, event_t event);
05911 event_t __attribute__((overloadable)) async_work_group_copy(__local ushort16 *dst, const __global
      ushort16 *src, size_t num_elements, event_t event);
05912 event_t __attribute__((overloadable)) async_work_group_copy(__local int16 *dst, const __global int16
      *src, size_t num_elements, event_t event);
05913 event_t __attribute__((overloadable)) async_work_group_copy(__local uint16 *dst, const __global uint16
      *src, size_t num_elements, event_t event);
```

```
05914 event_t __attribute__((overloadable)) async_work_group_copy(__local long16 *dst, const __global long16
      *src, size_t num_elements, event_t event);
05915 event_t __attribute__((overloadable)) async_work_group_copy(__local ulong16 *dst, const __global
      ulong16 *src, size_t num_elements, event_t event);
05916 event_t __attribute__((overloadable)) async_work_group_copy(__local float16 *dst, const __global
      float16 *src, size_t num_elements, event_t event);
05917 event_t __attribute__((overloadable)) async_work_group_copy(__global char *dst, const __local char
      *src, size_t num_elements, event_t event);
05918 event_t __attribute__((overloadable)) async_work_group_copy(__global uchar *dst, const __local uchar
      *src, size_t num_elements, event_t event);
05919 event_t __attribute__((overloadable)) async_work_group_copy(__global short *dst, const __local short
      *src, size_t num_elements, event_t event);
05920 event_t __attribute__((overloadable)) async_work_group_copy(__global ushort *dst, const __local ushort
      *src, size_t num_elements, event_t event);
05921 event_t __attribute__((overloadable)) async_work_group_copy(__global int *dst, const __local int *src,
      size_t num_elements, event_t event);
05922 event_t __attribute__((overloadable)) async_work_group_copy(__global uint *dst, const __local uint
      *src, size_t num_elements, event_t event);
05923 event_t __attribute__((overloadable)) async_work_group_copy(__global long *dst, const __local long
      *src, size_t num_elements, event_t event);
05924 event_t __attribute__((overloadable)) async_work_group_copy(__global ulong *dst, const __local ulong
      *src, size_t num_elements, event_t event);
05925 event_t __attribute__((overloadable)) async_work_group_copy(__global float *dst, const __local float
      *src, size_t num_elements, event_t event);
05926 event_t __attribute__((overloadable)) async_work_group_copy(__global char2 *dst, const __local char2
      *src, size_t num_elements, event_t event);
05927 event_t __attribute__((overloadable)) async_work_group_copy(__global uchar2 *dst, const __local uchar2
      *src, size_t num_elements, event_t event);
05928 event_t __attribute__((overloadable)) async_work_group_copy(__global short2 *dst, const __local short2
      *src, size_t num_elements, event_t event);
05929 event_t __attribute__((overloadable)) async_work_group_copy(__global ushort2 *dst, const __local
      ushort2 *src, size_t num_elements, event_t event);
05930 event_t __attribute__((overloadable)) async_work_group_copy(__global int2 *dst, const __local int2
      *src, size_t num_elements, event_t event);
05931 event_t __attribute__((overloadable)) async_work_group_copy(__global uint2 *dst, const __local uint2
      *src, size_t num_elements, event_t event);
05932 event_t __attribute__((overloadable)) async_work_group_copy(__global long2 *dst, const __local long2
      *src, size_t num_elements, event_t event);
05933 event_t __attribute__((overloadable)) async_work_group_copy(__global ulong2 *dst, const __local ulong2
      *src, size_t num_elements, event_t event);
05934 event_t __attribute__((overloadable)) async_work_group_copy(__global float2 *dst, const __local float2
      *src, size_t num_elements, event_t event);
05935 event_t __attribute__((overloadable)) async_work_group_copy(__global char3 *dst, const __local char3
      *src, size_t num_elements, event_t event);
05936 event_t __attribute__((overloadable)) async_work_group_copy(__global uchar3 *dst, const __local uchar3
      *src, size_t num_elements, event_t event);
05937 event_t __attribute__((overloadable)) async_work_group_copy(__global short3 *dst, const __local short3
      *src, size_t num_elements, event_t event);
05938 event_t __attribute__((overloadable)) async_work_group_copy(__global ushort3 *dst, const __local
      ushort3 *src, size_t num_elements, event_t event);
05939 event_t __attribute__((overloadable)) async_work_group_copy(__global int3 *dst, const __local int3
      *src, size_t num_elements, event_t event);
05940 event_t __attribute__((overloadable)) async_work_group_copy(__global uint3 *dst, const __local uint3
      *src, size_t num_elements, event_t event);
05941 event_t __attribute__((overloadable)) async_work_group_copy(__global long3 *dst, const __local long3
      *src, size_t num_elements, event_t event);
05942 event_t __attribute__((overloadable)) async_work_group_copy(__global ulong3 *dst, const __local ulong3
      *src, size_t num_elements, event_t event);
05943 event_t __attribute__((overloadable)) async_work_group_copy(__global float3 *dst, const __local float3
      *src, size_t num_elements, event_t event);
05944 event_t __attribute__((overloadable)) async_work_group_copy(__global char4 *dst, const __local char4
      *src, size_t num_elements, event_t event);
05945 event_t __attribute__((overloadable)) async_work_group_copy(__global uchar4 *dst, const __local uchar4
      *src, size_t num_elements, event_t event);
05946 event_t __attribute__((overloadable)) async_work_group_copy(__global short4 *dst, const __local short4
      *src, size_t num_elements, event_t event);
05947 event_t __attribute__((overloadable)) async_work_group_copy(__global ushort4 *dst, const __local
      ushort4 *src, size_t num_elements, event_t event);
05948 event_t __attribute__((overloadable)) async_work_group_copy(__global int4 *dst, const __local int4
      *src, size_t num_elements, event_t event);
05949 event_t __attribute__((overloadable)) async_work_group_copy(__global uint4 *dst, const __local uint4
      *src, size_t num_elements, event_t event);
05950 event_t __attribute__((overloadable)) async_work_group_copy(__global long4 *dst, const __local long4
      *src, size_t num_elements, event_t event);
05951 event_t __attribute__((overloadable)) async_work_group_copy(__global ulong4 *dst, const __local ulong4
      *src, size_t num_elements, event_t event);
05952 event_t __attribute__((overloadable)) async_work_group_copy(__global float4 *dst, const __local float4
      *src, size_t num_elements, event_t event);
05953 event_t __attribute__((overloadable)) async_work_group_copy(__global char8 *dst, const __local char8
      *src, size_t num_elements, event_t event);
05954 event_t __attribute__((overloadable)) async_work_group_copy(__global uchar8 *dst, const __local uchar8
      *src, size_t num_elements, event_t event);
05955 event_t __attribute__((overloadable)) async_work_group_copy(__global short8 *dst, const __local short8
      *src, size_t num_elements, event_t event);
05956 event_t __attribute__((overloadable)) async_work_group_copy(__global ushort8 *dst, const __local
      ushort8 *src, size_t num_elements, event_t event);
05957 event_t __attribute__((overloadable)) async_work_group_copy(__global int8 *dst, const __local int8
```

```
      *src, size_t num_elements, event_t event);
05958 event_t __attribute__((overloadable)) async_work_group_copy(__global uint8 *dst, const __local uint8
      *src, size_t num_elements, event_t event);
05959 event_t __attribute__((overloadable)) async_work_group_copy(__global long8 *dst, const __local long8
      *src, size_t num_elements, event_t event);
05960 event_t __attribute__((overloadable)) async_work_group_copy(__global ulong8 *dst, const __local ulong8
      *src, size_t num_elements, event_t event);
05961 event_t __attribute__((overloadable)) async_work_group_copy(__global float8 *dst, const __local float8
      *src, size_t num_elements, event_t event);
05962 event_t __attribute__((overloadable)) async_work_group_copy(__global char16 *dst, const __local char16
      *src, size_t num_elements, event_t event);
05963 event_t __attribute__((overloadable)) async_work_group_copy(__global uchar16 *dst, const __local
      uchar16 *src, size_t num_elements, event_t event);
05964 event_t __attribute__((overloadable)) async_work_group_copy(__global short16 *dst, const __local
      short16 *src, size_t num_elements, event_t event);
05965 event_t __attribute__((overloadable)) async_work_group_copy(__global ushort16 *dst, const __local
      ushort16 *src, size_t num_elements, event_t event);
05966 event_t __attribute__((overloadable)) async_work_group_copy(__global int16 *dst, const __local int16
      *src, size_t num_elements, event_t event);
05967 event_t __attribute__((overloadable)) async_work_group_copy(__global uint16 *dst, const __local uint16
      *src, size_t num_elements, event_t event);
05968 event_t __attribute__((overloadable)) async_work_group_copy(__global long16 *dst, const __local long16
      *src, size_t num_elements, event_t event);
05969 event_t __attribute__((overloadable)) async_work_group_copy(__global ulong16 *dst, const __local
      ulong16 *src, size_t num_elements, event_t event);
05970 event_t __attribute__((overloadable)) async_work_group_copy(__global float16 *dst, const __local
      float16 *src, size_t num_elements, event_t event);
05971 event_t __attribute__((overloadable)) async_work_group_copy(__local double *dst, const __global double
      *src, size_t num_elements, event_t event);
05972 event_t __attribute__((overloadable)) async_work_group_copy(__local double2 *dst, const __global
      double2 *src, size_t num_elements, event_t event);
05973 event_t __attribute__((overloadable)) async_work_group_copy(__local double3 *dst, const __global
      double3 *src, size_t num_elements, event_t event);
05974 event_t __attribute__((overloadable)) async_work_group_copy(__local double4 *dst, const __global
      double4 *src, size_t num_elements, event_t event);
05975 event_t __attribute__((overloadable)) async_work_group_copy(__local double8 *dst, const __global
      double8 *src, size_t num_elements, event_t event);
05976 event_t __attribute__((overloadable)) async_work_group_copy(__local double16 *dst, const __global
      double16 *src, size_t num_elements, event_t event);
05977 event_t __attribute__((overloadable)) async_work_group_copy(__global double *dst, const __local double
      *src, size_t num_elements, event_t event);
05978 event_t __attribute__((overloadable)) async_work_group_copy(__global double2 *dst, const __local
      double2 *src, size_t num_elements, event_t event);
05979 event_t __attribute__((overloadable)) async_work_group_copy(__global double3 *dst, const __local
      double3 *src, size_t num_elements, event_t event);
05980 event_t __attribute__((overloadable)) async_work_group_copy(__global double4 *dst, const __local
      double4 *src, size_t num_elements, event_t event);
05981 event_t __attribute__((overloadable)) async_work_group_copy(__global double8 *dst, const __local
      double8 *src, size_t num_elements, event_t event);
05982 event_t __attribute__((overloadable)) async_work_group_copy(__global double16 *dst, const __local
      double16 *src, size_t num_elements, event_t event);
05983 event_t __attribute__((overloadable)) async_work_group_copy(__local half *dst, const __global half
      *src, size_t num_elements, event_t event);
05984 event_t __attribute__((overloadable)) async_work_group_copy(__local half2 *dst, const __global half2
      *src, size_t num_elements, event_t event);
05985 event_t __attribute__((overloadable)) async_work_group_copy(__local half4 *dst, const __global half4
      *src, size_t num_elements, event_t event);
05986 event_t __attribute__((overloadable)) async_work_group_copy(__local half8 *dst, const __global half8
      *src, size_t num_elements, event_t event);
05987 event_t __attribute__((overloadable)) async_work_group_copy(__local half16 *dst, const __global half16
      *src, size_t num_elements, event_t event);
05988 event_t __attribute__((overloadable)) async_work_group_copy(__global half *dst, const __local half
      *src, size_t num_elements, event_t event);
05989 event_t __attribute__((overloadable)) async_work_group_copy(__global half2 *dst, const __local half2
      *src, size_t num_elements, event_t event);
05990 event_t __attribute__((overloadable)) async_work_group_copy(__global half4 *dst, const __local half4
      *src, size_t num_elements, event_t event);
05991 event_t __attribute__((overloadable)) async_work_group_copy(__global half8 *dst, const __local half8
      *src, size_t num_elements, event_t event);
05992 event_t __attribute__((overloadable)) async_work_group_copy(__global half16 *dst, const __local half16
      *src, size_t num_elements, event_t event);
05993
05994 /**
05995 * Perform an async gather of num_elements
05996 * gentype elements from src to dst.  The
05997 * src_stride is the stride in elements for each
05998 * gentype element read from src.  The dst_stride
05999 * is the stride in elements for each gentype
06000 * element written to dst.  The async gather is
06001 * performed by all work-items in a work-group.
06002 * This built-in function must therefore be
06003 * encountered by all work-items in a work-group
06004 * executing the kernel with the same argument
06005 * values; otherwise the results are undefined.
06006 * Returns an event object that can be used by
06007 * wait_group_events to wait for the async copy
06008 * to finish.  The event argument can also be used
```

```
06009 * to associate the
06010 * async_work_group_strided_copy with a
06011 * previous async copy allowing an event to be
06012 * shared by multiple async copies; otherwise event
06013 * should be zero.
06014 * If event argument is non-zero, the event object
06015 * supplied in event argument will be returned.
06016 * This function does not perform any implicit
06017 * synchronization of source data such as using a
06018 * barrier before performing the copy.
06019 */
06020 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local char *dst, const __global
      char *src, size_t num_elements, size_t src_stride, event_t event);
06021 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local uchar *dst, const __global
      uchar *src, size_t num_elements, size_t src_stride, event_t event);
06022 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local short *dst, const __global
      short *src, size_t num_elements, size_t src_stride, event_t event);
06023 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local ushort *dst, const
      __global ushort *src, size_t num_elements, size_t src_stride, event_t event);
06024 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local int *dst, const __global
      int *src, size_t num_elements, size_t src_stride, event_t event);
06025 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local uint *dst, const __global
      uint *src, size_t num_elements, size_t src_stride, event_t event);
06026 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local long *dst, const __global
      long *src, size_t num_elements, size_t src_stride, event_t event);
06027 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local ulong *dst, const __global
      ulong *src, size_t num_elements, size_t src_stride, event_t event);
06028 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local float *dst, const __global
      float *src, size_t num_elements, size_t src_stride, event_t event);
06029 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local char2 *dst, const __global
      char2 *src, size_t num_elements, size_t src_stride, event_t event);
06030 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local uchar2 *dst, const
      __global uchar2 *src, size_t num_elements, size_t src_stride, event_t event);
06031 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local short2 *dst, const
      __global short2 *src, size_t num_elements, size_t src_stride, event_t event);
06032 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local ushort2 *dst, const
      __global ushort2 *src, size_t num_elements, size_t src_stride, event_t event);
06033 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local int2 *dst, const __global
      int2 *src, size_t num_elements, size_t src_stride, event_t event);
06034 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local uint2 *dst, const __global
      uint2 *src, size_t num_elements, size_t src_stride, event_t event);
06035 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local long2 *dst, const __global
      long2 *src, size_t num_elements, size_t src_stride, event_t event);
06036 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local ulong2 *dst, const
      __global ulong2 *src, size_t num_elements, size_t src_stride, event_t event);
06037 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local float2 *dst, const
      __global float2 *src, size_t num_elements, size_t src_stride, event_t event);
06038 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local char3 *dst, const __global
      char3 *src, size_t num_elements, size_t src_stride, event_t event);
06039 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local uchar3 *dst, const
      __global uchar3 *src, size_t num_elements, size_t src_stride, event_t event);
06040 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local short3 *dst, const
      __global short3 *src, size_t num_elements, size_t src_stride, event_t event);
06041 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local ushort3 *dst, const
      __global ushort3 *src, size_t num_elements, size_t src_stride, event_t event);
06042 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local int3 *dst, const __global
      int3 *src, size_t num_elements, size_t src_stride, event_t event);
06043 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local uint3 *dst, const __global
      uint3 *src, size_t num_elements, size_t src_stride, event_t event);
06044 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local long3 *dst, const __global
      long3 *src, size_t num_elements, size_t src_stride, event_t event);
06045 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local ulong3 *dst, const
      __global ulong3 *src, size_t num_elements, size_t src_stride, event_t event);
06046 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local float3 *dst, const
      __global float3 *src, size_t num_elements, size_t src_stride, event_t event);
06047 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local char4 *dst, const __global
      char4 *src, size_t num_elements, size_t src_stride, event_t event);
06048 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local uchar4 *dst, const
      __global uchar4 *src, size_t num_elements, size_t src_stride, event_t event);
06049 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local short4 *dst, const
      __global short4 *src, size_t num_elements, size_t src_stride, event_t event);
06050 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local ushort4 *dst, const
      __global ushort4 *src, size_t num_elements, size_t src_stride, event_t event);
06051 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local int4 *dst, const __global
      int4 *src, size_t num_elements, size_t src_stride, event_t event);
06052 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local uint4 *dst, const __global
      uint4 *src, size_t num_elements, size_t src_stride, event_t event);
06053 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local long4 *dst, const __global
      long4 *src, size_t num_elements, size_t src_stride, event_t event);
06054 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local ulong4 *dst, const
      __global ulong4 *src, size_t num_elements, size_t src_stride, event_t event);
06055 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local float4 *dst, const
      __global float4 *src, size_t num_elements, size_t src_stride, event_t event);
06056 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local char8 *dst, const __global
      char8 *src, size_t num_elements, size_t src_stride, event_t event);
06057 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local uchar8 *dst, const
      __global uchar8 *src, size_t num_elements, size_t src_stride, event_t event);
```

```
06058 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local short8 *dst, const
      __global short8 *src, size_t num_elements, size_t src_stride, event_t event);
06059 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local ushort8 *dst, const
      __global ushort8 *src, size_t num_elements, size_t src_stride, event_t event);
06060 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local int8 *dst, const __global
      int8 *src, size_t num_elements, size_t src_stride, event_t event);
06061 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local uint8 *dst, const __global
      uint8 *src, size_t num_elements, size_t src_stride, event_t event);
06062 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local long8 *dst, const __global
      long8 *src, size_t num_elements, size_t src_stride, event_t event);
06063 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local ulong8 *dst, const
      __global ulong8 *src, size_t num_elements, size_t src_stride, event_t event);
06064 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local float8 *dst, const
      __global float8 *src, size_t num_elements, size_t src_stride, event_t event);
06065 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local char16 *dst, const
      __global char16 *src, size_t num_elements, size_t src_stride, event_t event);
06066 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local uchar16 *dst, const
      __global uchar16 *src, size_t num_elements, size_t src_stride, event_t event);
06067 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local short16 *dst, const
      __global short16 *src, size_t num_elements, size_t src_stride, event_t event);
06068 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local ushort16 *dst, const
      __global ushort16 *src, size_t num_elements, size_t src_stride, event_t event);
06069 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local int16 *dst, const __global
      int16 *src, size_t num_elements, size_t src_stride, event_t event);
06070 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local uint16 *dst, const
      __global uint16 *src, size_t num_elements, size_t src_stride, event_t event);
06071 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local long16 *dst, const
      __global long16 *src, size_t num_elements, size_t src_stride, event_t event);
06072 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local ulong16 *dst, const
      __global ulong16 *src, size_t num_elements, size_t src_stride, event_t event);
06073 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local float16 *dst, const
      __global float16 *src, size_t num_elements, size_t src_stride, event_t event);
06074 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global char *dst, const __local
      char *src, size_t num_elements, size_t dst_stride, event_t event);
06075 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global uchar *dst, const __local
      uchar *src, size_t num_elements, size_t dst_stride, event_t event);
06076 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global short *dst, const __local
      short *src, size_t num_elements, size_t dst_stride, event_t event);
06077 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global ushort *dst, const
      __local ushort *src, size_t num_elements, size_t dst_stride, event_t event);
06078 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global int *dst, const __local
      int *src, size_t num_elements, size_t dst_stride, event_t event);
06079 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global uint *dst, const __local
      uint *src, size_t num_elements, size_t dst_stride, event_t event);
06080 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global long *dst, const __local
      long *src, size_t num_elements, size_t dst_stride, event_t event);
06081 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global ulong *dst, const __local
      ulong *src, size_t num_elements, size_t dst_stride, event_t event);
06082 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global float *dst, const __local
      float *src, size_t num_elements, size_t dst_stride, event_t event);
06083 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global char2 *dst, const __local
      char2 *src, size_t num_elements, size_t dst_stride, event_t event);
06084 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global uchar2 *dst, const
      __local uchar2 *src, size_t num_elements, size_t dst_stride, event_t event);
06085 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global short2 *dst, const
      __local short2 *src, size_t num_elements, size_t dst_stride, event_t event);
06086 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global ushort2 *dst, const
      __local ushort2 *src, size_t num_elements, size_t dst_stride, event_t event);
06087 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global int2 *dst, const __local
      int2 *src, size_t num_elements, size_t dst_stride, event_t event);
06088 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global uint2 *dst, const __local
      uint2 *src, size_t num_elements, size_t dst_stride, event_t event);
06089 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global long2 *dst, const __local
      long2 *src, size_t num_elements, size_t dst_stride, event_t event);
06090 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global ulong2 *dst, const
      __local ulong2 *src, size_t num_elements, size_t dst_stride, event_t event);
06091 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global float2 *dst, const
      __local float2 *src, size_t num_elements, size_t dst_stride, event_t event);
06092 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global char3 *dst, const __local
      char3 *src, size_t num_elements, size_t dst_stride, event_t event);
06093 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global uchar3 *dst, const
      __local uchar3 *src, size_t num_elements, size_t dst_stride, event_t event);
06094 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global short3 *dst, const
      __local short3 *src, size_t num_elements, size_t dst_stride, event_t event);
06095 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global ushort3 *dst, const
      __local ushort3 *src, size_t num_elements, size_t dst_stride, event_t event);
06096 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global int3 *dst, const __local
      int3 *src, size_t num_elements, size_t dst_stride, event_t event);
06097 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global uint3 *dst, const __local
      uint3 *src, size_t num_elements, size_t dst_stride, event_t event);
06098 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global long3 *dst, const __local
      long3 *src, size_t num_elements, size_t dst_stride, event_t event);
06099 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global ulong3 *dst, const
      __local ulong3 *src, size_t num_elements, size_t dst_stride, event_t event);
06100 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global float3 *dst, const
      __local float3 *src, size_t num_elements, size_t dst_stride, event_t event);
06101 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global char4 *dst, const __local
```

```
      char4 *src, size_t num_elements, size_t dst_stride, event_t event);
06102 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global uchar4 *dst, const
      __local uchar4 *src, size_t num_elements, size_t dst_stride, event_t event);
06103 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global short4 *dst, const
      __local short4 *src, size_t num_elements, size_t dst_stride, event_t event);
06104 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global ushort4 *dst, const
      __local ushort4 *src, size_t num_elements, size_t dst_stride, event_t event);
06105 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global int4 *dst, const __local
      int4 *src, size_t num_elements, size_t dst_stride, event_t event);
06106 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global uint4 *dst, const __local
      uint4 *src, size_t num_elements, size_t dst_stride, event_t event);
06107 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global long4 *dst, const __local
      long4 *src, size_t num_elements, size_t dst_stride, event_t event);
06108 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global ulong4 *dst, const
      __local ulong4 *src, size_t num_elements, size_t dst_stride, event_t event);
06109 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global float4 *dst, const
      __local float4 *src, size_t num_elements, size_t dst_stride, event_t event);
06110 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global char8 *dst, const __local
      char8 *src, size_t num_elements, size_t dst_stride, event_t event);
06111 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global uchar8 *dst, const
      __local uchar8 *src, size_t num_elements, size_t dst_stride, event_t event);
06112 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global short8 *dst, const
      __local short8 *src, size_t num_elements, size_t dst_stride, event_t event);
06113 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global ushort8 *dst, const
      __local ushort8 *src, size_t num_elements, size_t dst_stride, event_t event);
06114 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global int8 *dst, const __local
      int8 *src, size_t num_elements, size_t dst_stride, event_t event);
06115 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global uint8 *dst, const __local
      uint8 *src, size_t num_elements, size_t dst_stride, event_t event);
06116 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global long8 *dst, const __local
      long8 *src, size_t num_elements, size_t dst_stride, event_t event);
06117 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global ulong8 *dst, const
      __local ulong8 *src, size_t num_elements, size_t dst_stride, event_t event);
06118 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global float8 *dst, const
      __local float8 *src, size_t num_elements, size_t dst_stride, event_t event);
06119 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global char16 *dst, const
      __local char16 *src, size_t num_elements, size_t dst_stride, event_t event);
06120 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global uchar16 *dst, const
      __local uchar16 *src, size_t num_elements, size_t dst_stride, event_t event);
06121 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global short16 *dst, const
      __local short16 *src, size_t num_elements, size_t dst_stride, event_t event);
06122 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global ushort16 *dst, const
      __local ushort16 *src, size_t num_elements, size_t dst_stride, event_t event);
06123 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global int16 *dst, const __local
      int16 *src, size_t num_elements, size_t dst_stride, event_t event);
06124 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global uint16 *dst, const
      __local uint16 *src, size_t num_elements, size_t dst_stride, event_t event);
06125 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global long16 *dst, const
      __local long16 *src, size_t num_elements, size_t dst_stride, event_t event);
06126 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global ulong16 *dst, const
      __local ulong16 *src, size_t num_elements, size_t dst_stride, event_t event);
06127 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global float16 *dst, const
      __local float16 *src, size_t num_elements, size_t dst_stride, event_t event);
06128 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local double *dst, const
      __global double *src, size_t num_elements, size_t src_stride, event_t event);
06129 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local double2 *dst, const
      __global double2 *src, size_t num_elements, size_t src_stride, event_t event);
06130 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local double3 *dst, const
      __global double3 *src, size_t num_elements, size_t src_stride, event_t event);
06131 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local double4 *dst, const
      __global double4 *src, size_t num_elements, size_t src_stride, event_t event);
06132 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local double8 *dst, const
      __global double8 *src, size_t num_elements, size_t src_stride, event_t event);
06133 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local double16 *dst, const
      __global double16 *src, size_t num_elements, size_t src_stride, event_t event);
06134 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global double *dst, const
      __local double *src, size_t num_elements, size_t dst_stride, event_t event);
06135 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global double2 *dst, const
      __local double2 *src, size_t num_elements, size_t dst_stride, event_t event);
06136 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global double3 *dst, const
      __local double3 *src, size_t num_elements, size_t dst_stride, event_t event);
06137 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global double4 *dst, const
      __local double4 *src, size_t num_elements, size_t dst_stride, event_t event);
06138 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global double8 *dst, const
      __local double8 *src, size_t num_elements, size_t dst_stride, event_t event);
06139 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global double16 *dst, const
      __local double16 *src, size_t num_elements, size_t dst_stride, event_t event);
06140 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local half *dst, const __global
      half *src, size_t num_elements, size_t src_stride, event_t event);
06141 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local half2 *dst, const __global
      half2 *src, size_t num_elements, size_t src_stride, event_t event);
06142 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local half4 *dst, const __global
      half4 *src, size_t num_elements, size_t src_stride, event_t event);
06143 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local half8 *dst, const __global
      half8 *src, size_t num_elements, size_t src_stride, event_t event);
06144 event_t __attribute__((overloadable)) async_work_group_strided_copy(__local half16 *dst, const
      __global half16 *src, size_t num_elements, size_t src_stride, event_t event);
```

```
06145 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global half *dst, const __local
      half *src, size_t num_elements, size_t dst_stride, event_t event);
06146 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global half2 *dst, const __local
      half2 *src, size_t num_elements, size_t dst_stride, event_t event);
06147 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global half4 *dst, const __local
      half4 *src, size_t num_elements, size_t dst_stride, event_t event);
06148 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global half8 *dst, const __local
      half8 *src, size_t num_elements, size_t dst_stride, event_t event);
06149 event_t __attribute__((overloadable)) async_work_group_strided_copy(__global half16 *dst, const
      __local half16 *src, size_t num_elements, size_t dst_stride, event_t event);
06150
06151 /**
06152 * Wait for events that identify the
06153 * async_work_group_copy operations to
06154 * complete.  The event objects specified in
06155 * event_list will be released after the wait is
06156 * performed.
06157 * This function must be encountered by all workitems
06158 * in a work-group executing the kernel with
06159 * the same num_events and event objects specified
06160 * in event_list; otherwise the results are undefined.
06161 */
06162 void __attribute__((overloadable)) wait_group_events(int num_events, event_t *event_list);
06163
06164 /**
06165 * Prefetch num_elements * sizeof(gentype)
06166 * bytes into the global cache.  The prefetch
06167 * instruction is applied to a work-item in a workgroup
06168 * and does not affect the functional
06169 * behavior of the kernel.
06170 */
06171 void __attribute__((overloadable)) prefetch(const __global char *p, size_t num_elements);
06172 void __attribute__((overloadable)) prefetch(const __global uchar *p, size_t num_elements);
06173 void __attribute__((overloadable)) prefetch(const __global short *p, size_t num_elements);
06174 void __attribute__((overloadable)) prefetch(const __global ushort *p, size_t num_elements);
06175 void __attribute__((overloadable)) prefetch(const __global int *p, size_t num_elements);
06176 void __attribute__((overloadable)) prefetch(const __global uint *p, size_t num_elements);
06177 void __attribute__((overloadable)) prefetch(const __global long *p, size_t num_elements);
06178 void __attribute__((overloadable)) prefetch(const __global ulong *p, size_t num_elements);
06179 void __attribute__((overloadable)) prefetch(const __global float *p, size_t num_elements);
06180 void __attribute__((overloadable)) prefetch(const __global char2 *p, size_t num_elements);
06181 void __attribute__((overloadable)) prefetch(const __global uchar2 *p, size_t num_elements);
06182 void __attribute__((overloadable)) prefetch(const __global short2 *p, size_t num_elements);
06183 void __attribute__((overloadable)) prefetch(const __global ushort2 *p, size_t num_elements);
06184 void __attribute__((overloadable)) prefetch(const __global int2 *p, size_t num_elements);
06185 void __attribute__((overloadable)) prefetch(const __global uint2 *p, size_t num_elements);
06186 void __attribute__((overloadable)) prefetch(const __global long2 *p, size_t num_elements);
06187 void __attribute__((overloadable)) prefetch(const __global ulong2 *p, size_t num_elements);
06188 void __attribute__((overloadable)) prefetch(const __global float2 *p, size_t num_elements);
06189 void __attribute__((overloadable)) prefetch(const __global char3 *p, size_t num_elements);
06190 void __attribute__((overloadable)) prefetch(const __global uchar3 *p, size_t num_elements);
06191 void __attribute__((overloadable)) prefetch(const __global short3 *p, size_t num_elements);
06192 void __attribute__((overloadable)) prefetch(const __global ushort3 *p, size_t num_elements);
06193 void __attribute__((overloadable)) prefetch(const __global int3 *p, size_t num_elements);
06194 void __attribute__((overloadable)) prefetch(const __global uint3 *p, size_t num_elements);
06195 void __attribute__((overloadable)) prefetch(const __global long3 *p, size_t num_elements);
06196 void __attribute__((overloadable)) prefetch(const __global ulong3 *p, size_t num_elements);
06197 void __attribute__((overloadable)) prefetch(const __global float3 *p, size_t num_elements);
06198 void __attribute__((overloadable)) prefetch(const __global char4 *p, size_t num_elements);
06199 void __attribute__((overloadable)) prefetch(const __global uchar4 *p, size_t num_elements);
06200 void __attribute__((overloadable)) prefetch(const __global short4 *p, size_t num_elements);
06201 void __attribute__((overloadable)) prefetch(const __global ushort4 *p, size_t num_elements);
06202 void __attribute__((overloadable)) prefetch(const __global int4 *p, size_t num_elements);
06203 void __attribute__((overloadable)) prefetch(const __global uint4 *p, size_t num_elements);
06204 void __attribute__((overloadable)) prefetch(const __global long4 *p, size_t num_elements);
06205 void __attribute__((overloadable)) prefetch(const __global ulong4 *p, size_t num_elements);
06206 void __attribute__((overloadable)) prefetch(const __global float4 *p, size_t num_elements);
06207 void __attribute__((overloadable)) prefetch(const __global char8 *p, size_t num_elements);
06208 void __attribute__((overloadable)) prefetch(const __global uchar8 *p, size_t num_elements);
06209 void __attribute__((overloadable)) prefetch(const __global short8 *p, size_t num_elements);
06210 void __attribute__((overloadable)) prefetch(const __global ushort8 *p, size_t num_elements);
06211 void __attribute__((overloadable)) prefetch(const __global int8 *p, size_t num_elements);
06212 void __attribute__((overloadable)) prefetch(const __global uint8 *p, size_t num_elements);
06213 void __attribute__((overloadable)) prefetch(const __global long8 *p, size_t num_elements);
06214 void __attribute__((overloadable)) prefetch(const __global ulong8 *p, size_t num_elements);
06215 void __attribute__((overloadable)) prefetch(const __global float8 *p, size_t num_elements);
06216 void __attribute__((overloadable)) prefetch(const __global char16 *p, size_t num_elements);
06217 void __attribute__((overloadable)) prefetch(const __global uchar16 *p, size_t num_elements);
06218 void __attribute__((overloadable)) prefetch(const __global short16 *p, size_t num_elements);
06219 void __attribute__((overloadable)) prefetch(const __global ushort16 *p, size_t num_elements);
06220 void __attribute__((overloadable)) prefetch(const __global int16 *p, size_t num_elements);
06221 void __attribute__((overloadable)) prefetch(const __global uint16 *p, size_t num_elements);
06222 void __attribute__((overloadable)) prefetch(const __global long16 *p, size_t num_elements);
06223 void __attribute__((overloadable)) prefetch(const __global ulong16 *p, size_t num_elements);
06224 void __attribute__((overloadable)) prefetch(const __global float16 *p, size_t num_elements);
06225 void __attribute__((overloadable)) prefetch(const __global double *p, size_t num_elements);
06226 void __attribute__((overloadable)) prefetch(const __global double2 *p, size_t num_elements);
```

```
06227 void __attribute__((overloadable)) prefetch(const __global double3 *p, size_t num_elements);
06228 void __attribute__((overloadable)) prefetch(const __global double4 *p, size_t num_elements);
06229 void __attribute__((overloadable)) prefetch(const __global double8 *p, size_t num_elements);
06230 void __attribute__((overloadable)) prefetch(const __global double16 *p, size_t num_elements);
06231 void __attribute__((overloadable)) prefetch(const __global half *p, size_t num_elements);
06232 void __attribute__((overloadable)) prefetch(const __global half2 *p, size_t num_elements);
06233 void __attribute__((overloadable)) prefetch(const __global half4 *p, size_t num_elements);
06234 void __attribute__((overloadable)) prefetch(const __global half8 *p, size_t num_elements);
06235 void __attribute__((overloadable)) prefetch(const __global half16 *p, size_t num_elements);
06236
06237 // Atomic functions
06238
06239 /**
06240 * Read the 32-bit value (referred to as old)
06241 * stored at location pointed by p.  Compute
06242 * (old + val) and store result at location
06243 * pointed by p.  The function returns old.
06244 */
06245 int __attribute__((overloadable)) atomic_add(volatile __global int *p, int val);
06246 unsigned int __attribute__((overloadable)) atomic_add(volatile __global unsigned int *p, unsigned int
      val);
06247 int __attribute__((overloadable)) atomic_add(volatile __local int *p, int val);
06248 unsigned int __attribute__((overloadable)) atomic_add(volatile __local unsigned int *p, unsigned int
      val);
06249
06250 int __attribute__((overloadable)) atom_add(__global int *p, int val);
06251 unsigned int __attribute__((overloadable)) atom_add(__global unsigned int *p, unsigned int val);
06252 int __attribute__((overloadable)) atom_add(__local int *p, int val);
06253 unsigned int __attribute__((overloadable)) atom_add(__local unsigned int *p, unsigned int val);
06254
06255 int __attribute__((overloadable)) atom_add(volatile __global int *p, int val);
06256 unsigned int __attribute__((overloadable)) atom_add(volatile __global unsigned int *p, unsigned int
      val);
06257 int __attribute__((overloadable)) atom_add(volatile __local int *p, int val);
06258 unsigned int __attribute__((overloadable)) atom_add(volatile __local unsigned int *p, unsigned int
      val);
06259
06260 /**
06261 * unsigned int atomic_sub (
06262 * volatile __global unsigned int *p,
06263 * unsigned int val)
06264 * int atomic_sub (volatile __local int *p, int val)
06265 * unsigned int atomic_sub (
06266 * volatile __local unsigned int *p,
06267 * unsigned int val)
06268 * Read the 32-bit value (referred to as old)
06269 * stored at location pointed by p.  Compute
06270 * (old - val) and store result at location
06271 * pointed by p.  The function returns old.
06272 */
06273 int __attribute__((overloadable)) atomic_sub(volatile __global int *p, int val);
06274 unsigned int __attribute__((overloadable)) atomic_sub(volatile __global unsigned int *p, unsigned int
      val);
06275 int __attribute__((overloadable)) atomic_sub(volatile __local int *p, int val);
06276 unsigned int __attribute__((overloadable)) atomic_sub(volatile __local unsigned int *p, unsigned int
      val);
06277
06278 int __attribute__((overloadable)) atom_sub(__global int *p, int val);
06279 unsigned int __attribute__((overloadable)) atom_sub(__global unsigned int *p, unsigned int val);
06280 int __attribute__((overloadable)) atom_sub(__local int *p, int val);
06281 unsigned int __attribute__((overloadable)) atom_sub(__local unsigned int *p, unsigned int val);
06282
06283 int __attribute__((overloadable)) atom_sub(volatile __global int *p, int val);
06284 unsigned int __attribute__((overloadable)) atom_sub(volatile __global unsigned int *p, unsigned int
      val);
06285 int __attribute__((overloadable)) atom_sub(volatile __local int *p, int val);
06286 unsigned int __attribute__((overloadable)) atom_sub(volatile __local unsigned int *p, unsigned int
      val);
06287
06288 /**
06289 * Swaps the old value stored at location p
06290 * with new value given by val.  Returns old
06291 * value.
06292 */
06293 int __attribute__((overloadable)) atomic_xchg(volatile __global int *p, int val);
06294 unsigned int __attribute__((overloadable)) atomic_xchg(volatile __global unsigned int *p, unsigned int
      val);
06295 float __attribute__((overloadable)) atomic_xchg(volatile __global float *p, float val);
06296 int __attribute__((overloadable)) atomic_xchg(volatile __local int *p, int val);
06297 unsigned int __attribute__((overloadable)) atomic_xchg(volatile __local unsigned int *p, unsigned int
      val);
06298 float __attribute__((overloadable)) atomic_xchg(volatile __local float *p, float val);
06299
06300 int __attribute__((overloadable)) atom_xchg(__global int *p, int val);
06301 unsigned int __attribute__((overloadable)) atom_xchg(__global unsigned int *p, unsigned int val);
06302 int __attribute__((overloadable)) atom_xchg(__local int *p, int val);
06303 unsigned int __attribute__((overloadable)) atom_xchg(__local unsigned int *p, unsigned int val);
```

```
06304
06305 int __attribute__((overloadable)) atom_xchg(volatile __global int *p, int val);
06306 unsigned int __attribute__((overloadable)) atom_xchg(volatile __global unsigned int *p, unsigned int
      val);
06307 int __attribute__((overloadable)) atom_xchg(volatile __local int *p, int val);
06308 unsigned int __attribute__((overloadable)) atom_xchg(volatile __local unsigned int *p, unsigned int
      val);
06309
06310 /**
06311 * Read the 32-bit value (referred to as old)
06312 * stored at location pointed by p.  Compute
06313 * (old + 1) and store result at location
06314 * pointed by p.  The function returns old.
06315 */
06316 int __attribute__((overloadable)) atomic_inc(volatile __global int *p);
06317 unsigned int __attribute__((overloadable)) atomic_inc(volatile __global unsigned int *p);
06318 int __attribute__((overloadable)) atomic_inc(volatile __local int *p);
06319 unsigned int __attribute__((overloadable)) atomic_inc(volatile __local unsigned int *p);
06320
06321 int __attribute__((overloadable)) atom_inc(__global int *p);
06322 unsigned int __attribute__((overloadable)) atom_inc(__global unsigned int *p);
06323 int __attribute__((overloadable)) atom_inc(__local int *p);
06324 unsigned int __attribute__((overloadable)) atom_inc(__local unsigned int *p);
06325
06326 int __attribute__((overloadable)) atom_inc(volatile __global int *p);
06327 unsigned int __attribute__((overloadable)) atom_inc(volatile __global unsigned int *p);
06328 int __attribute__((overloadable)) atom_inc(volatile __local int *p);
06329 unsigned int __attribute__((overloadable)) atom_inc(volatile __local unsigned int *p);
06330
06331 /**
06332 * Read the 32-bit value (referred to as old)
06333 * stored at location pointed by p.  Compute
06334 * (old - 1) and store result at location
06335 * pointed by p.  The function returns old.
06336 */
06337 int __attribute__((overloadable)) atomic_dec(volatile __global int *p);
06338 unsigned int __attribute__((overloadable)) atomic_dec(volatile __global unsigned int *p);
06339 int __attribute__((overloadable)) atomic_dec(volatile __local int *p);
06340 unsigned int __attribute__((overloadable)) atomic_dec(volatile __local unsigned int *p);
06341
06342 int __attribute__((overloadable)) atom_dec(__global int *p);
06343 unsigned int __attribute__((overloadable)) atom_dec(__global unsigned int *p);
06344 int __attribute__((overloadable)) atom_dec(__local int *p);
06345 unsigned int __attribute__((overloadable)) atom_dec(__local unsigned int *p);
06346
06347 int __attribute__((overloadable)) atom_dec(volatile __global int *p);
06348 unsigned int __attribute__((overloadable)) atom_dec(volatile __global unsigned int *p);
06349 int __attribute__((overloadable)) atom_dec(volatile __local int *p);
06350 unsigned int __attribute__((overloadable)) atom_dec(volatile __local unsigned int *p);
06351
06352 /**
06353 * Read the 32-bit value (referred to as old)
06354 * stored at location pointed by p.  Compute
06355 * (old == cmp) ?  val :  old and store result at
06356 * location pointed by p.  The function
06357 * returns old.
06358 */
06359 int __attribute__((overloadable)) atomic_cmpxchg(volatile __global int *p, int cmp, int val);
06360 unsigned int __attribute__((overloadable)) atomic_cmpxchg(volatile __global unsigned int *p, unsigned
      int cmp, unsigned int val);
06361 int __attribute__((overloadable)) atomic_cmpxchg(volatile __local int *p, int cmp, int val);
06362 unsigned int __attribute__((overloadable)) atomic_cmpxchg(volatile __local unsigned int *p, unsigned
      int cmp, unsigned int val);
06363
06364 int __attribute__((overloadable)) atom_cmpxchg(__global int *p, int cmp, int val);
06365 unsigned int __attribute__((overloadable)) atom_cmpxchg(__global unsigned int *p, unsigned int cmp,
      unsigned int val);
06366 int __attribute__((overloadable)) atom_cmpxchg(__local int *p, int cmp, int val);
06367 unsigned int __attribute__((overloadable)) atom_cmpxchg(__local unsigned int *p, unsigned int cmp,
      unsigned int val);
06368
06369 int __attribute__((overloadable)) atom_cmpxchg(volatile __global int *p, int cmp, int val);
06370 unsigned int __attribute__((overloadable)) atom_cmpxchg(volatile __global unsigned int *p, unsigned
      int cmp, unsigned int val);
06371 int __attribute__((overloadable)) atom_cmpxchg(volatile __local int *p, int cmp, int val);
06372 unsigned int __attribute__((overloadable)) atom_cmpxchg(volatile __local unsigned int *p, unsigned int
      cmp, unsigned int val);
06373
06374 /**
06375 * Read the 32-bit value (referred to as old)
06376 * stored at location pointed by p.  Compute
06377 * min(old, val) and store minimum value at
06378 * location pointed by p.  The function
06379 * returns old.
06380 */
06381 int __attribute__((overloadable)) atomic_min(volatile __global int *p, int val);
06382 unsigned int __attribute__((overloadable)) atomic_min(volatile __global unsigned int *p, unsigned int
```

```
      val);
06383 int __attribute__((overloadable)) atomic_min(volatile __local int *p, int val);
06384 unsigned int __attribute__((overloadable)) atomic_min(volatile __local unsigned int *p, unsigned int
      val);
06385
06386 int __attribute__((overloadable)) atom_min(__global int *p, int val);
06387 unsigned int __attribute__((overloadable)) atom_min(__global unsigned int *p, unsigned int val);
06388 int __attribute__((overloadable)) atom_min(__local int *p, int val);
06389 unsigned int __attribute__((overloadable)) atom_min(__local unsigned int *p, unsigned int val);
06390
06391 int __attribute__((overloadable)) atom_min(volatile __global int *p, int val);
06392 unsigned int __attribute__((overloadable)) atom_min(volatile __global unsigned int *p, unsigned int
      val);
06393 int __attribute__((overloadable)) atom_min(volatile __local int *p, int val);
06394 unsigned int __attribute__((overloadable)) atom_min(volatile __local unsigned int *p, unsigned int
      val);
06395
06396 /**
06397 * Read the 32-bit value (referred to as old)
06398 * stored at location pointed by p.  Compute
06399 * max(old, val) and store maximum value at
06400 * location pointed by p.  The function
06401 * returns old.
06402 */
06403 int __attribute__((overloadable)) atomic_max(volatile __global int *p, int val);
06404 unsigned int __attribute__((overloadable)) atomic_max(volatile __global unsigned int *p, unsigned int
      val);
06405 int __attribute__((overloadable)) atomic_max(volatile __local int *p, int val);
06406 unsigned int __attribute__((overloadable)) atomic_max(volatile __local unsigned int *p, unsigned int
      val);
06407
06408 int __attribute__((overloadable)) atom_max(__global int *p, int val);
06409 unsigned int __attribute__((overloadable)) atom_max(__global unsigned int *p, unsigned int val);
06410 int __attribute__((overloadable)) atom_max(__local int *p, int val);
06411 unsigned int __attribute__((overloadable)) atom_max(__local unsigned int *p, unsigned int val);
06412
06413 int __attribute__((overloadable)) atom_max(volatile __global int *p, int val);
06414 unsigned int __attribute__((overloadable)) atom_max(volatile __global unsigned int *p, unsigned int
      val);
06415 int __attribute__((overloadable)) atom_max(volatile __local int *p, int val);
06416 unsigned int __attribute__((overloadable)) atom_max(volatile __local unsigned int *p, unsigned int
      val);
06417
06418 /**
06419 * Read the 32-bit value (referred to as old)
06420 * stored at location pointed by p.  Compute
06421 * (old & val) and store result at location
06422 * pointed by p.  The function returns old.
06423 */
06424 int __attribute__((overloadable)) atomic_and(volatile __global int *p, int val);
06425 unsigned int __attribute__((overloadable)) atomic_and(volatile __global unsigned int *p, unsigned int
      val);
06426 int __attribute__((overloadable)) atomic_and(volatile __local int *p, int val);
06427 unsigned int __attribute__((overloadable)) atomic_and(volatile __local unsigned int *p, unsigned int
      val);
06428
06429 int __attribute__((overloadable)) atom_and(__global int *p, int val);
06430 unsigned int __attribute__((overloadable)) atom_and(__global unsigned int *p, unsigned int val);
06431 int __attribute__((overloadable)) atom_and(__local int *p, int val);
06432 unsigned int __attribute__((overloadable)) atom_and(__local unsigned int *p, unsigned int val);
06433
06434 int __attribute__((overloadable)) atom_and(volatile __global int *p, int val);
06435 unsigned int __attribute__((overloadable)) atom_and(volatile __global unsigned int *p, unsigned int
      val);
06436 int __attribute__((overloadable)) atom_and(volatile __local int *p, int val);
06437 unsigned int __attribute__((overloadable)) atom_and(volatile __local unsigned int *p, unsigned int
      val);
06438
06439 /**
06440 * Read the 32-bit value (referred to as old)
06441 * stored at location pointed by p.  Compute
06442 * (old | val) and store result at location
06443 * pointed by p.  The function returns old.
06444 */
06445 int __attribute__((overloadable)) atomic_or(volatile __global int *p, int val);
06446 unsigned int __attribute__((overloadable)) atomic_or(volatile __global unsigned int *p, unsigned int
      val);
06447 int __attribute__((overloadable)) atomic_or(volatile __local int *p, int val);
06448 unsigned int __attribute__((overloadable)) atomic_or(volatile __local unsigned int *p, unsigned int
      val);
06449
06450 int __attribute__((overloadable)) atom_or(__global int *p, int val);
06451 unsigned int __attribute__((overloadable)) atom_or(__global unsigned int *p, unsigned int val);
06452 int __attribute__((overloadable)) atom_or(__local int *p, int val);
06453 unsigned int __attribute__((overloadable)) atom_or(__local unsigned int *p, unsigned int val);
06454
06455 int __attribute__((overloadable)) atom_or(volatile __global int *p, int val);
```

```
06456 unsigned int __attribute__((overloadable)) atom_or(volatile __global unsigned int *p, unsigned int
      val);
06457 int __attribute__((overloadable)) atom_or(volatile __local int *p, int val);
06458 unsigned int __attribute__((overloadable)) atom_or(volatile __local unsigned int *p, unsigned int
      val);
06459
06460 /**
06461 * Read the 32-bit value (referred to as old)
06462 * stored at location pointed by p.  Compute
06463 * (old ^ val) and store result at location
06464 * pointed by p.  The function returns old.
06465 */
06466 int __attribute__((overloadable)) atomic_xor(volatile __global int *p, int val);
06467 unsigned int __attribute__((overloadable)) atomic_xor(volatile __global unsigned int *p, unsigned int
      val);
06468 int __attribute__((overloadable)) atomic_xor(volatile __local int *p, int val);
06469 unsigned int __attribute__((overloadable)) atomic_xor(volatile __local unsigned int *p, unsigned int
      val);
06470
06471 int __attribute__((overloadable)) atom_xor(__global int *p, int val);
06472 unsigned int __attribute__((overloadable)) atom_xor(__global unsigned int *p, unsigned int val);
06473 int __attribute__((overloadable)) atom_xor(__local int *p, int val);
06474 unsigned int __attribute__((overloadable)) atom_xor(__local unsigned int *p, unsigned int val);
06475
06476 int __attribute__((overloadable)) atom_xor(volatile __global int *p, int val);
06477 unsigned int __attribute__((overloadable)) atom_xor(volatile __global unsigned int *p, unsigned int
      val);
06478 int __attribute__((overloadable)) atom_xor(volatile __local int *p, int val);
06479 unsigned int __attribute__((overloadable)) atom_xor(volatile __local unsigned int *p, unsigned int
      val);
06480
06481 // Function qualifiers (section 6.7)
06482
06483 #define __kernel_exec(X, typen) __kernel \
06484 __attribute__((work_group_size_hint(X, 1, 1))) \
06485 __attribute__((vec_type_hint(typen)))
06486
06487 #define kernel_exec(X, typen) __kernel \
06488 __attribute__((work_group_size_hint(X, 1, 1))) \
06489 __attribute__((vec_type_hint(typen)))
06490
06491 // Miscellaneous vector functions
06492
06493 /**
06494 * The shuffle and shuffle2 built-in functions construct
06495 * a permutation of elements from one or two input
06496 * vectors respectively that are of the same type,
06497 * returning a vector with the same element type as the
06498 * input and length that is the same as the shuffle mask.
06499 * The size of each element in the mask must match the
06500 * size of each element in the result.  For shuffle, only
06501 * the ilogb(2m-1) least significant bits of each mask
06502 * element are considered.  For shuffle2, only the
06503 * ilogb(2m-1)+1 least significant bits of each mask
06504 * element are considered.  Other bits in the mask shall
06505 * be ignored.
06506 * The elements of the input vectors are numbered from
06507 * left to right across one or both of the vectors.  For this
06508 * purpose, the number of elements in a vector is given
06509 * by vec_step(gentypem).  The shuffle mask operand
06510 * specifies, for each element of the result vector, which
06511 * element of the one or two input vectors the result
06512 * element gets.
06513 * Examples:
06514 * uint4 mask = (uint4)(3, 2,
06515 * 1, 0);
06516 * float4 a;
06517 * float4 r = shuffle(a, mask);
06518 * // r.s0123 = a.wzyx
06519 * uint8 mask = (uint8)(0, 1, 2, 3,
06520 * 4, 5, 6, 7);
06521 * float4 a, b;
06522 * float8 r = shuffle2(a, b, mask);
06523 * // r.s0123 = a.xyzw
06524 * // r.s4567 = b.xyzw
06525 * uint4 mask;
06526 * float8 a;
06527 * float4 b;
06528 * b = shuffle(a, mask);
06529 * Examples that are not valid are:
06530 * uint8 mask;
06531 * short16 a;
06532 * short8 b;
06533 * b = shuffle(a, mask); <- not valid
06534 */
06535 char2       const_func __attribute__((overloadable)) shuffle(char2 x, uchar2 mask);
06536 char2       const_func __attribute__((overloadable)) shuffle(char4 x, uchar2 mask);
```

```
06537 char2       const_func __attribute__((overloadable)) shuffle(char8 x, uchar2 mask);
06538 char2       const_func __attribute__((overloadable)) shuffle(char16 x, uchar2 mask);
06539
06540 uchar2      const_func __attribute__((overloadable)) shuffle(uchar2 x, uchar2 mask);
06541 uchar2      const_func __attribute__((overloadable)) shuffle(uchar4 x, uchar2 mask);
06542 uchar2      const_func __attribute__((overloadable)) shuffle(uchar8 x, uchar2 mask);
06543 uchar2      const_func __attribute__((overloadable)) shuffle(uchar16 x, uchar2 mask);
06544
06545 short2 const_func __attribute__((overloadable)) shuffle(short2 x, ushort2 mask);
06546 short2 const_func __attribute__((overloadable)) shuffle(short4 x, ushort2 mask);
06547 short2 const_func __attribute__((overloadable)) shuffle(short8 x, ushort2 mask);
06548 short2 const_func __attribute__((overloadable)) shuffle(short16 x, ushort2 mask);
06549
06550 ushort2 const_func __attribute__((overloadable)) shuffle(ushort2 x, ushort2 mask);
06551 ushort2 const_func __attribute__((overloadable)) shuffle(ushort4 x, ushort2 mask);
06552 ushort2 const_func __attribute__((overloadable)) shuffle(ushort8 x, ushort2 mask);
06553 ushort2 const_func __attribute__((overloadable)) shuffle(ushort16 x, ushort2 mask);
06554
06555 int2 const_func __attribute__((overloadable)) shuffle(int2 x, uint2 mask);
06556 int2 const_func __attribute__((overloadable)) shuffle(int4 x, uint2 mask);
06557 int2 const_func __attribute__((overloadable)) shuffle(int8 x, uint2 mask);
06558 int2 const_func __attribute__((overloadable)) shuffle(int16 x, uint2 mask);
06559
06560 uint2 const_func __attribute__((overloadable)) shuffle(uint2 x, uint2 mask);
06561 uint2 const_func __attribute__((overloadable)) shuffle(uint4 x, uint2 mask);
06562 uint2 const_func __attribute__((overloadable)) shuffle(uint8 x, uint2 mask);
06563 uint2 const_func __attribute__((overloadable)) shuffle(uint16 x, uint2 mask);
06564
06565 long2 const_func __attribute__((overloadable)) shuffle(long2 x, ulong2 mask);
06566 long2 const_func __attribute__((overloadable)) shuffle(long4 x, ulong2 mask);
06567 long2 const_func __attribute__((overloadable)) shuffle(long8 x, ulong2 mask);
06568 long2 const_func __attribute__((overloadable)) shuffle(long16 x, ulong2 mask);
06569
06570 ulong2 const_func __attribute__((overloadable)) shuffle(ulong2 x, ulong2 mask);
06571 ulong2 const_func __attribute__((overloadable)) shuffle(ulong4 x, ulong2 mask);
06572 ulong2 const_func __attribute__((overloadable)) shuffle(ulong8 x, ulong2 mask);
06573 ulong2 const_func __attribute__((overloadable)) shuffle(ulong16 x, ulong2 mask);
06574
06575 //half2 const_func __attribute__((overloadable)) shuffle(half2 x, ushort2 mask);
06576 //half2 const_func __attribute__((overloadable)) shuffle(half4 x, ushort2 mask);
06577 //half2 const_func __attribute__((overloadable)) shuffle(half8 x, ushort2 mask);
06578 //half2 const_func __attribute__((overloadable)) shuffle(half16 x, ushort2 mask);
06579
06580 float2 const_func __attribute__((overloadable)) shuffle(float2 x, uint2 mask);
06581 float2 const_func __attribute__((overloadable)) shuffle(float4 x, uint2 mask);
06582 float2 const_func __attribute__((overloadable)) shuffle(float8 x, uint2 mask);
06583 float2 const_func __attribute__((overloadable)) shuffle(float16 x, uint2 mask);
06584
06585 double2 const_func __attribute__((overloadable)) shuffle(double2 x, ulong2 mask);
06586 double2 const_func __attribute__((overloadable)) shuffle(double4 x, ulong2 mask);
06587 double2 const_func __attribute__((overloadable)) shuffle(double8 x, ulong2 mask);
06588 double2 const_func __attribute__((overloadable)) shuffle(double16 x, ulong2 mask);
06589
06590 char4 const_func __attribute__((overloadable)) shuffle(char2 x, uchar4 mask);
06591 char4 const_func __attribute__((overloadable)) shuffle(char4 x, uchar4 mask);
06592 char4 const_func __attribute__((overloadable)) shuffle(char8 x, uchar4 mask);
06593 char4 const_func __attribute__((overloadable)) shuffle(char16 x, uchar4 mask);
06594
06595 uchar4 const_func __attribute__((overloadable)) shuffle(uchar2 x, uchar4 mask);
06596 uchar4 const_func __attribute__((overloadable)) shuffle(uchar4 x, uchar4 mask);
06597 uchar4 const_func __attribute__((overloadable)) shuffle(uchar8 x, uchar4 mask);
06598 uchar4 const_func __attribute__((overloadable)) shuffle(uchar16 x, uchar4 mask);
06599
06600 short4 const_func __attribute__((overloadable)) shuffle(short2 x, ushort4 mask);
06601 short4 const_func __attribute__((overloadable)) shuffle(short4 x, ushort4 mask);
06602 short4 const_func __attribute__((overloadable)) shuffle(short8 x, ushort4 mask);
06603 short4 const_func __attribute__((overloadable)) shuffle(short16 x, ushort4 mask);
06604
06605 ushort4 const_func __attribute__((overloadable)) shuffle(ushort2 x, ushort4 mask);
06606 ushort4 const_func __attribute__((overloadable)) shuffle(ushort4 x, ushort4 mask);
06607 ushort4 const_func __attribute__((overloadable)) shuffle(ushort8 x, ushort4 mask);
06608 ushort4 const_func __attribute__((overloadable)) shuffle(ushort16 x, ushort4 mask);
06609
06610 int4 const_func __attribute__((overloadable)) shuffle(int2 x, uint4 mask);
06611 int4 const_func __attribute__((overloadable)) shuffle(int4 x, uint4 mask);
06612 int4 const_func __attribute__((overloadable)) shuffle(int8 x, uint4 mask);
06613 int4 const_func __attribute__((overloadable)) shuffle(int16 x, uint4 mask);
06614
06615 uint4 const_func __attribute__((overloadable)) shuffle(uint2 x, uint4 mask);
06616 uint4 const_func __attribute__((overloadable)) shuffle(uint4 x, uint4 mask);
06617 uint4 const_func __attribute__((overloadable)) shuffle(uint8 x, uint4 mask);
06618 uint4 const_func __attribute__((overloadable)) shuffle(uint16 x, uint4 mask);
06619
06620 long4 const_func __attribute__((overloadable)) shuffle(long2 x, ulong4 mask);
06621 long4 const_func __attribute__((overloadable)) shuffle(long4 x, ulong4 mask);
06622 long4 const_func __attribute__((overloadable)) shuffle(long8 x, ulong4 mask);
06623 long4 const_func __attribute__((overloadable)) shuffle(long16 x, ulong4 mask);
```

```
06624
06625 ulong4 const_func __attribute__((overloadable)) shuffle(ulong2 x, ulong4 mask);
06626 ulong4 const_func __attribute__((overloadable)) shuffle(ulong4 x, ulong4 mask);
06627 ulong4 const_func __attribute__((overloadable)) shuffle(ulong8 x, ulong4 mask);
06628 ulong4 const_func __attribute__((overloadable)) shuffle(ulong16 x, ulong4 mask);
06629
06630 //half4 const_func __attribute__((overloadable)) shuffle(half2 x, ushort4 mask);
06631 //half4 const_func __attribute__((overloadable)) shuffle(half4 x, ushort4 mask);
06632 //half4 const_func __attribute__((overloadable)) shuffle(half8 x, ushort4 mask);
06633 //half4 const_func __attribute__((overloadable)) shuffle(half16 x, ushort4 mask);
06634
06635 float4 const_func __attribute__((overloadable)) shuffle(float2 x, uint4 mask);
06636 float4 const_func __attribute__((overloadable)) shuffle(float4 x, uint4 mask);
06637 float4 const_func __attribute__((overloadable)) shuffle(float8 x, uint4 mask);
06638 float4 const_func __attribute__((overloadable)) shuffle(float16 x, uint4 mask);
06639
06640 double4 const_func __attribute__((overloadable)) shuffle(double2 x, ulong4 mask);
06641 double4 const_func __attribute__((overloadable)) shuffle(double4 x, ulong4 mask);
06642 double4 const_func __attribute__((overloadable)) shuffle(double8 x, ulong4 mask);
06643 double4 const_func __attribute__((overloadable)) shuffle(double16 x, ulong4 mask);
06644
06645 char8 const_func __attribute__((overloadable)) shuffle(char2 x, uchar8 mask);
06646 char8 const_func __attribute__((overloadable)) shuffle(char4 x, uchar8 mask);
06647 char8 const_func __attribute__((overloadable)) shuffle(char8 x, uchar8 mask);
06648 char8 const_func __attribute__((overloadable)) shuffle(char16 x, uchar8 mask);
06649
06650 uchar8 const_func __attribute__((overloadable)) shuffle(uchar2 x, uchar8 mask);
06651 uchar8 const_func __attribute__((overloadable)) shuffle(uchar4 x, uchar8 mask);
06652 uchar8 const_func __attribute__((overloadable)) shuffle(uchar8 x, uchar8 mask);
06653 uchar8 const_func __attribute__((overloadable)) shuffle(uchar16 x, uchar8 mask);
06654
06655 short8 const_func __attribute__((overloadable)) shuffle(short2 x, ushort8 mask);
06656 short8 const_func __attribute__((overloadable)) shuffle(short4 x, ushort8 mask);
06657 short8 const_func __attribute__((overloadable)) shuffle(short8 x, ushort8 mask);
06658 short8 const_func __attribute__((overloadable)) shuffle(short16 x, ushort8 mask);
06659
06660 ushort8 const_func __attribute__((overloadable)) shuffle(ushort2 x, ushort8 mask);
06661 ushort8 const_func __attribute__((overloadable)) shuffle(ushort4 x, ushort8 mask);
06662 ushort8 const_func __attribute__((overloadable)) shuffle(ushort8 x, ushort8 mask);
06663 ushort8 const_func __attribute__((overloadable)) shuffle(ushort16 x, ushort8 mask);
06664
06665 int8 const_func __attribute__((overloadable)) shuffle(int2 x, uint8 mask);
06666 int8 const_func __attribute__((overloadable)) shuffle(int4 x, uint8 mask);
06667 int8 const_func __attribute__((overloadable)) shuffle(int8 x, uint8 mask);
06668 int8 const_func __attribute__((overloadable)) shuffle(int16 x, uint8 mask);
06669
06670 uint8 const_func __attribute__((overloadable)) shuffle(uint2 x, uint8 mask);
06671 uint8 const_func __attribute__((overloadable)) shuffle(uint4 x, uint8 mask);
06672 uint8 const_func __attribute__((overloadable)) shuffle(uint8 x, uint8 mask);
06673 uint8 const_func __attribute__((overloadable)) shuffle(uint16 x, uint8 mask);
06674
06675 long8 const_func __attribute__((overloadable)) shuffle(long2 x, ulong8 mask);
06676 long8 const_func __attribute__((overloadable)) shuffle(long4 x, ulong8 mask);
06677 long8 const_func __attribute__((overloadable)) shuffle(long8 x, ulong8 mask);
06678 long8 const_func __attribute__((overloadable)) shuffle(long16 x, ulong8 mask);
06679
06680 ulong8 const_func __attribute__((overloadable)) shuffle(ulong2 x, ulong8 mask);
06681 ulong8 const_func __attribute__((overloadable)) shuffle(ulong4 x, ulong8 mask);
06682 ulong8 const_func __attribute__((overloadable)) shuffle(ulong8 x, ulong8 mask);
06683 ulong8 const_func __attribute__((overloadable)) shuffle(ulong16 x, ulong8 mask);
06684
06685 //half8 const_func __attribute__((overloadable)) shuffle(half2 x, ushort8 mask);
06686 //half8 const_func __attribute__((overloadable)) shuffle(half4 x, ushort8 mask);
06687 //half8 const_func __attribute__((overloadable)) shuffle(half8 x, ushort8 mask);
06688 //half8 const_func __attribute__((overloadable)) shuffle(half16 x, ushort8 mask);
06689
06690 float8 const_func __attribute__((overloadable)) shuffle(float2 x, uint8 mask);
06691 float8 const_func __attribute__((overloadable)) shuffle(float4 x, uint8 mask);
06692 float8 const_func __attribute__((overloadable)) shuffle(float8 x, uint8 mask);
06693 float8 const_func __attribute__((overloadable)) shuffle(float16 x, uint8 mask);
06694
06695 double8 const_func __attribute__((overloadable)) shuffle(double2 x, ulong8 mask);
06696 double8 const_func __attribute__((overloadable)) shuffle(double4 x, ulong8 mask);
06697 double8 const_func __attribute__((overloadable)) shuffle(double8 x, ulong8 mask);
06698 double8 const_func __attribute__((overloadable)) shuffle(double16 x, ulong8 mask);
06699
06700 char16 const_func __attribute__((overloadable)) shuffle(char2 x, uchar16 mask);
06701 char16 const_func __attribute__((overloadable)) shuffle(char4 x, uchar16 mask);
06702 char16 const_func __attribute__((overloadable)) shuffle(char8 x, uchar16 mask);
06703 char16 const_func __attribute__((overloadable)) shuffle(char16 x, uchar16 mask);
06704
06705 uchar16 const_func __attribute__((overloadable)) shuffle(uchar2 x, uchar16 mask);
06706 uchar16 const_func __attribute__((overloadable)) shuffle(uchar4 x, uchar16 mask);
06707 uchar16 const_func __attribute__((overloadable)) shuffle(uchar8 x, uchar16 mask);
06708 uchar16 const_func __attribute__((overloadable)) shuffle(uchar16 x, uchar16 mask);
06709
06710 short16 const_func __attribute__((overloadable)) shuffle(short2 x, ushort16 mask);
```

```
06711 short16 const_func __attribute__((overloadable)) shuffle(short4 x, ushort16 mask);
06712 short16 const_func __attribute__((overloadable)) shuffle(short8 x, ushort16 mask);
06713 short16 const_func __attribute__((overloadable)) shuffle(short16 x, ushort16 mask);
06714
06715 ushort16 const_func __attribute__((overloadable)) shuffle(ushort2 x, ushort16 mask);
06716 ushort16 const_func __attribute__((overloadable)) shuffle(ushort4 x, ushort16 mask);
06717 ushort16 const_func __attribute__((overloadable)) shuffle(ushort8 x, ushort16 mask);
06718 ushort16 const_func __attribute__((overloadable)) shuffle(ushort16 x, ushort16 mask);
06719
06720 int16 const_func __attribute__((overloadable)) shuffle(int2 x, uint16 mask);
06721 int16 const_func __attribute__((overloadable)) shuffle(int4 x, uint16 mask);
06722 int16 const_func __attribute__((overloadable)) shuffle(int8 x, uint16 mask);
06723 int16 const_func __attribute__((overloadable)) shuffle(int16 x, uint16 mask);
06724
06725 uint16 const_func __attribute__((overloadable)) shuffle(uint2 x, uint16 mask);
06726 uint16 const_func __attribute__((overloadable)) shuffle(uint4 x, uint16 mask);
06727 uint16 const_func __attribute__((overloadable)) shuffle(uint8 x, uint16 mask);
06728 uint16 const_func __attribute__((overloadable)) shuffle(uint16 x, uint16 mask);
06729
06730 long16 const_func __attribute__((overloadable)) shuffle(long2 x, ulong16 mask);
06731 long16 const_func __attribute__((overloadable)) shuffle(long4 x, ulong16 mask);
06732 long16 const_func __attribute__((overloadable)) shuffle(long8 x, ulong16 mask);
06733 long16 const_func __attribute__((overloadable)) shuffle(long16 x, ulong16 mask);
06734
06735 ulong16 const_func __attribute__((overloadable)) shuffle(ulong2 x, ulong16 mask);
06736 ulong16 const_func __attribute__((overloadable)) shuffle(ulong4 x, ulong16 mask);
06737 ulong16 const_func __attribute__((overloadable)) shuffle(ulong8 x, ulong16 mask);
06738 ulong16 const_func __attribute__((overloadable)) shuffle(ulong16 x, ulong16 mask);
06739
06740 //half16 const_func __attribute__((overloadable)) shuffle(half2 x, ushort16 mask);
06741 //half16 const_func __attribute__((overloadable)) shuffle(half4 x, ushort16 mask);
06742 //half16 const_func __attribute__((overloadable)) shuffle(half8 x, ushort16 mask);
06743 //half16 const_func __attribute__((overloadable)) shuffle(half16 x, ushort16 mask);
06744
06745 float16 const_func __attribute__((overloadable)) shuffle(float2 x, uint16 mask);
06746 float16 const_func __attribute__((overloadable)) shuffle(float4 x, uint16 mask);
06747 float16 const_func __attribute__((overloadable)) shuffle(float8 x, uint16 mask);
06748 float16 const_func __attribute__((overloadable)) shuffle(float16 x, uint16 mask);
06749
06750 double16 const_func __attribute__((overloadable)) shuffle(double2 x, ulong16 mask);
06751 double16 const_func __attribute__((overloadable)) shuffle(double4 x, ulong16 mask);
06752 double16 const_func __attribute__((overloadable)) shuffle(double8 x, ulong16 mask);
06753 double16 const_func __attribute__((overloadable)) shuffle(double16 x, ulong16 mask);
06754
06755
06756 char2 const_func __attribute__((overloadable)) shuffle2(char2 x, char2 y, uchar2 mask);
06757 char2 const_func __attribute__((overloadable)) shuffle2(char4 x, char4 y, uchar2 mask);
06758 char2 const_func __attribute__((overloadable)) shuffle2(char8 x, char8 y, uchar2 mask);
06759 char2 const_func __attribute__((overloadable)) shuffle2(char16 x, char16 y, uchar2 mask);
06760
06761 uchar2 const_func __attribute__((overloadable)) shuffle2(uchar2 x, uchar2 y, uchar2 mask);
06762 uchar2 const_func __attribute__((overloadable)) shuffle2(uchar4 x, uchar4 y, uchar2 mask);
06763 uchar2 const_func __attribute__((overloadable)) shuffle2(uchar8 x, uchar8 y, uchar2 mask);
06764 uchar2 const_func __attribute__((overloadable)) shuffle2(uchar16 x, uchar16 y, uchar2 mask);
06765
06766 short2 const_func __attribute__((overloadable)) shuffle2(short2 x, short2 y, ushort2 mask);
06767 short2 const_func __attribute__((overloadable)) shuffle2(short4 x, short4 y, ushort2 mask);
06768 short2 const_func __attribute__((overloadable)) shuffle2(short8 x, short8 y, ushort2 mask);
06769 short2 const_func __attribute__((overloadable)) shuffle2(short16 x, short16 y, ushort2 mask);
06770
06771 ushort2 const_func __attribute__((overloadable)) shuffle2(ushort2 x, ushort2 y, ushort2 mask);
06772 ushort2 const_func __attribute__((overloadable)) shuffle2(ushort4 x, ushort4 y, ushort2 mask);
06773 ushort2 const_func __attribute__((overloadable)) shuffle2(ushort8 x, ushort8 y, ushort2 mask);
06774 ushort2 const_func __attribute__((overloadable)) shuffle2(ushort16 x, ushort16 y, ushort2 mask);
06775
06776 int2 const_func __attribute__((overloadable)) shuffle2(int2 x, int2 y, uint2 mask);
06777 int2 const_func __attribute__((overloadable)) shuffle2(int4 x, int4 y, uint2 mask);
06778 int2 const_func __attribute__((overloadable)) shuffle2(int8 x, int8 y, uint2 mask);
06779 int2 const_func __attribute__((overloadable)) shuffle2(int16 x, int16 y, uint2 mask);
06780
06781 uint2 const_func __attribute__((overloadable)) shuffle2(uint2 x, uint2 y, uint2 mask);
06782 uint2 const_func __attribute__((overloadable)) shuffle2(uint4 x, uint4 y, uint2 mask);
06783 uint2 const_func __attribute__((overloadable)) shuffle2(uint8 x, uint8 y, uint2 mask);
06784 uint2 const_func __attribute__((overloadable)) shuffle2(uint16 x, uint16 y, uint2 mask);
06785
06786 long2 const_func __attribute__((overloadable)) shuffle2(long2 x, long2 y, ulong2 mask);
06787 long2 const_func __attribute__((overloadable)) shuffle2(long4 x, long4 y, ulong2 mask);
06788 long2 const_func __attribute__((overloadable)) shuffle2(long8 x, long8 y, ulong2 mask);
06789 long2 const_func __attribute__((overloadable)) shuffle2(long16 x, long16 y, ulong2 mask);
06790
06791 ulong2 const_func __attribute__((overloadable)) shuffle2(ulong2 x, ulong2 y, ulong2 mask);
06792 ulong2 const_func __attribute__((overloadable)) shuffle2(ulong4 x, ulong4 y, ulong2 mask);
06793 ulong2 const_func __attribute__((overloadable)) shuffle2(ulong8 x, ulong8 y, ulong2 mask);
06794 ulong2 const_func __attribute__((overloadable)) shuffle2(ulong16 x, ulong16 y, ulong2 mask);
06795
06796 //half2 const_func __attribute__((overloadable)) shuffle2(half2 x, half2 y, ushort2 mask);
06797 //half2 const_func __attribute__((overloadable)) shuffle2(half4 x, half4 y, ushort2 mask);
```

```
06798 //half2 const_func __attribute__((overloadable)) shuffle2(half8 x, half8 y, ushort2 mask);
06799 //half2 const_func __attribute__((overloadable)) shuffle2(half16 x, half16 y, ushort2 mask);
06800
06801 float2 const_func __attribute__((overloadable)) shuffle2(float2 x, float2 y, uint2 mask);
06802 float2 const_func __attribute__((overloadable)) shuffle2(float4 x, float4 y, uint2 mask);
06803 float2 const_func __attribute__((overloadable)) shuffle2(float8 x, float8 y, uint2 mask);
06804 float2 const_func __attribute__((overloadable)) shuffle2(float16 x, float16 y, uint2 mask);
06805
06806 double2 const_func __attribute__((overloadable)) shuffle2(double2 x, double2 y, ulong2 mask);
06807 double2 const_func __attribute__((overloadable)) shuffle2(double4 x, double4 y, ulong2 mask);
06808 double2 const_func __attribute__((overloadable)) shuffle2(double8 x, double8 y, ulong2 mask);
06809 double2 const_func __attribute__((overloadable)) shuffle2(double16 x, double16 y, ulong2 mask);
06810
06811 char4 const_func __attribute__((overloadable)) shuffle2(char2 x, char2 y, uchar4 mask);
06812 char4 const_func __attribute__((overloadable)) shuffle2(char4 x, char4 y, uchar4 mask);
06813 char4 const_func __attribute__((overloadable)) shuffle2(char8 x, char8 y, uchar4 mask);
06814 char4 const_func __attribute__((overloadable)) shuffle2(char16 x, char16 y, uchar4 mask);
06815
06816 uchar4 const_func __attribute__((overloadable)) shuffle2(uchar2 x, uchar2 y, uchar4 mask);
06817 uchar4 const_func __attribute__((overloadable)) shuffle2(uchar4 x, uchar4 y, uchar4 mask);
06818 uchar4 const_func __attribute__((overloadable)) shuffle2(uchar8 x, uchar8 y, uchar4 mask);
06819 uchar4 const_func __attribute__((overloadable)) shuffle2(uchar16 x, uchar16 y, uchar4 mask);
06820
06821 short4 const_func __attribute__((overloadable)) shuffle2(short2 x, short2 y, ushort4 mask);
06822 short4 const_func __attribute__((overloadable)) shuffle2(short4 x, short4 y, ushort4 mask);
06823 short4 const_func __attribute__((overloadable)) shuffle2(short8 x, short8 y, ushort4 mask);
06824 short4 const_func __attribute__((overloadable)) shuffle2(short16 x, short16 y, ushort4 mask);
06825
06826 ushort4 const_func __attribute__((overloadable)) shuffle2(ushort2 x, ushort2 y, ushort4 mask);
06827 ushort4 const_func __attribute__((overloadable)) shuffle2(ushort4 x, ushort4 y, ushort4 mask);
06828 ushort4 const_func __attribute__((overloadable)) shuffle2(ushort8 x, ushort8 y, ushort4 mask);
06829 ushort4 const_func __attribute__((overloadable)) shuffle2(ushort16 x, ushort16 y, ushort4 mask);
06830
06831 int4 const_func __attribute__((overloadable)) shuffle2(int2 x, int2 y, uint4 mask);
06832 int4 const_func __attribute__((overloadable)) shuffle2(int4 x, int4 y, uint4 mask);
06833 int4 const_func __attribute__((overloadable)) shuffle2(int8 x, int8 y, uint4 mask);
06834 int4 const_func __attribute__((overloadable)) shuffle2(int16 x, int16 y, uint4 mask);
06835
06836 uint4 const_func __attribute__((overloadable)) shuffle2(uint2 x, uint2 y, uint4 mask);
06837 uint4 const_func __attribute__((overloadable)) shuffle2(uint4 x, uint4 y, uint4 mask);
06838 uint4 const_func __attribute__((overloadable)) shuffle2(uint8 x, uint8 y, uint4 mask);
06839 uint4 const_func __attribute__((overloadable)) shuffle2(uint16 x, uint16 y, uint4 mask);
06840
06841 long4 const_func __attribute__((overloadable)) shuffle2(long2 x, long2 y, ulong4 mask);
06842 long4 const_func __attribute__((overloadable)) shuffle2(long4 x, long4 y, ulong4 mask);
06843 long4 const_func __attribute__((overloadable)) shuffle2(long8 x, long8 y, ulong4 mask);
06844 long4 const_func __attribute__((overloadable)) shuffle2(long16 x, long16 y, ulong4 mask);
06845
06846 ulong4 const_func __attribute__((overloadable)) shuffle2(ulong2 x, ulong2 y, ulong4 mask);
06847 ulong4 const_func __attribute__((overloadable)) shuffle2(ulong4 x, ulong4 y, ulong4 mask);
06848 ulong4 const_func __attribute__((overloadable)) shuffle2(ulong8 x, ulong8 y, ulong4 mask);
06849 ulong4 const_func __attribute__((overloadable)) shuffle2(ulong16 x, ulong16 y, ulong4 mask);
06850
06851 //half4 const_func __attribute__((overloadable)) shuffle2(half2 x, half2 y, ushort4 mask);
06852 //half4 const_func __attribute__((overloadable)) shuffle2(half4 x, half4 y, ushort4 mask);
06853 //half4 const_func __attribute__((overloadable)) shuffle2(half8 x, half8 y, ushort4 mask);
06854 //half4 const_func __attribute__((overloadable)) shuffle2(half16 x, half16 y, ushort4 mask);
06855
06856 float4 const_func __attribute__((overloadable)) shuffle2(float2 x, float2 y, uint4 mask);
06857 float4 const_func __attribute__((overloadable)) shuffle2(float4 x, float4 y, uint4 mask);
06858 float4 const_func __attribute__((overloadable)) shuffle2(float8 x, float8 y, uint4 mask);
06859 float4 const_func __attribute__((overloadable)) shuffle2(float16 x, float16 y, uint4 mask);
06860
06861 double4 const_func __attribute__((overloadable)) shuffle2(double2 x, double2 y, ulong4 mask);
06862 double4 const_func __attribute__((overloadable)) shuffle2(double4 x, double4 y, ulong4 mask);
06863 double4 const_func __attribute__((overloadable)) shuffle2(double8 x, double8 y, ulong4 mask);
06864 double4 const_func __attribute__((overloadable)) shuffle2(double16 x, double16 y, ulong4 mask);
06865
06866 char8 const_func __attribute__((overloadable)) shuffle2(char2 x, char2 y, uchar8 mask);
06867 char8 const_func __attribute__((overloadable)) shuffle2(char4 x, char4 y, uchar8 mask);
06868 char8 const_func __attribute__((overloadable)) shuffle2(char8 x, char8 y, uchar8 mask);
06869 char8 const_func __attribute__((overloadable)) shuffle2(char16 x, char16 y, uchar8 mask);
06870
06871 uchar8 const_func __attribute__((overloadable)) shuffle2(uchar2 x, uchar2 y, uchar8 mask);
06872 uchar8 const_func __attribute__((overloadable)) shuffle2(uchar4 x, uchar4 y, uchar8 mask);
06873 uchar8 const_func __attribute__((overloadable)) shuffle2(uchar8 x, uchar8 y, uchar8 mask);
06874 uchar8 const_func __attribute__((overloadable)) shuffle2(uchar16 x, uchar16 y, uchar8 mask);
06875
06876 short8 const_func __attribute__((overloadable)) shuffle2(short2 x, short2 y, ushort8 mask);
06877 short8 const_func __attribute__((overloadable)) shuffle2(short4 x, short4 y, ushort8 mask);
06878 short8 const_func __attribute__((overloadable)) shuffle2(short8 x, short8 y, ushort8 mask);
06879 short8 const_func __attribute__((overloadable)) shuffle2(short16 x, short16 y, ushort8 mask);
06880
06881 ushort8 const_func __attribute__((overloadable)) shuffle2(ushort2 x, ushort2 y, ushort8 mask);
06882 ushort8 const_func __attribute__((overloadable)) shuffle2(ushort4 x, ushort4 y, ushort8 mask);
06883 ushort8 const_func __attribute__((overloadable)) shuffle2(ushort8 x, ushort8 y, ushort8 mask);
06884 ushort8 const_func __attribute__((overloadable)) shuffle2(ushort16 x, ushort16 y, ushort8 mask);
```

```
06885
06886 int8 const_func __attribute__((overloadable)) shuffle2(int2 x, int2 y, uint8 mask);
06887 int8 const_func __attribute__((overloadable)) shuffle2(int4 x, int4 y, uint8 mask);
06888 int8 const_func __attribute__((overloadable)) shuffle2(int8 x, int8 y, uint8 mask);
06889 int8 const_func __attribute__((overloadable)) shuffle2(int16 x, int16 y, uint8 mask);
06890
06891 uint8 const_func __attribute__((overloadable)) shuffle2(uint2 x, uint2 y, uint8 mask);
06892 uint8 const_func __attribute__((overloadable)) shuffle2(uint4 x, uint4 y, uint8 mask);
06893 uint8 const_func __attribute__((overloadable)) shuffle2(uint8 x, uint8 y, uint8 mask);
06894 uint8 const_func __attribute__((overloadable)) shuffle2(uint16 x, uint16 y, uint8 mask);
06895
06896 long8 const_func __attribute__((overloadable)) shuffle2(long2 x, long2 y, ulong8 mask);
06897 long8 const_func __attribute__((overloadable)) shuffle2(long4 x, long4 y, ulong8 mask);
06898 long8 const_func __attribute__((overloadable)) shuffle2(long8 x, long8 y, ulong8 mask);
06899 long8 const_func __attribute__((overloadable)) shuffle2(long16 x, long16 y, ulong8 mask);
06900
06901 ulong8 const_func __attribute__((overloadable)) shuffle2(ulong2 x, ulong2 y, ulong8 mask);
06902 ulong8 const_func __attribute__((overloadable)) shuffle2(ulong4 x, ulong4 y, ulong8 mask);
06903 ulong8 const_func __attribute__((overloadable)) shuffle2(ulong8 x, ulong8 y, ulong8 mask);
06904 ulong8 const_func __attribute__((overloadable)) shuffle2(ulong16 x, ulong16 y, ulong8 mask);
06905
06906 //half8 const_func __attribute__((overloadable)) shuffle2(half2 x, half2 y, ushort8 mask);
06907 //half8 const_func __attribute__((overloadable)) shuffle2(half4 x, half4 y, ushort8 mask);
06908 //half8 const_func __attribute__((overloadable)) shuffle2(half8 x, half8 y, ushort8 mask);
06909 //half8 const_func __attribute__((overloadable)) shuffle2(half16 x, half16 y, ushort8 mask);
06910
06911 float8 const_func __attribute__((overloadable)) shuffle2(float2 x, float2 y, uint8 mask);
06912 float8 const_func __attribute__((overloadable)) shuffle2(float4 x, float4 y, uint8 mask);
06913 float8 const_func __attribute__((overloadable)) shuffle2(float8 x, float8 y, uint8 mask);
06914 float8 const_func __attribute__((overloadable)) shuffle2(float16 x, float16 y, uint8 mask);
06915
06916 double8 const_func __attribute__((overloadable)) shuffle2(double2 x, double2 y, ulong8 mask);
06917 double8 const_func __attribute__((overloadable)) shuffle2(double4 x, double4 y, ulong8 mask);
06918 double8 const_func __attribute__((overloadable)) shuffle2(double8 x, double8 y, ulong8 mask);
06919 double8 const_func __attribute__((overloadable)) shuffle2(double16 x, double16 y, ulong8 mask);
06920
06921 char16 const_func __attribute__((overloadable)) shuffle2(char2 x, char2 y, uchar16 mask);
06922 char16 const_func __attribute__((overloadable)) shuffle2(char4 x, char4 y, uchar16 mask);
06923 char16 const_func __attribute__((overloadable)) shuffle2(char8 x, char8 y, uchar16 mask);
06924 char16 const_func __attribute__((overloadable)) shuffle2(char16 x, char16 y, uchar16 mask);
06925
06926 uchar16 const_func __attribute__((overloadable)) shuffle2(uchar2 x, uchar2 y, uchar16 mask);
06927 uchar16 const_func __attribute__((overloadable)) shuffle2(uchar4 x, uchar4 y, uchar16 mask);
06928 uchar16 const_func __attribute__((overloadable)) shuffle2(uchar8 x, uchar8 y, uchar16 mask);
06929 uchar16 const_func __attribute__((overloadable)) shuffle2(uchar16 x, uchar16 y, uchar16 mask);
06930
06931 short16 const_func __attribute__((overloadable)) shuffle2(short2 x, short2 y, ushort16 mask);
06932 short16 const_func __attribute__((overloadable)) shuffle2(short4 x, short4 y, ushort16 mask);
06933 short16 const_func __attribute__((overloadable)) shuffle2(short8 x, short8 y, ushort16 mask);
06934 short16 const_func __attribute__((overloadable)) shuffle2(short16 x, short16 y, ushort16 mask);
06935
06936 ushort16 const_func __attribute__((overloadable)) shuffle2(ushort2 x, ushort2 y, ushort16 mask);
06937 ushort16 const_func __attribute__((overloadable)) shuffle2(ushort4 x, ushort4 y, ushort16 mask);
06938 ushort16 const_func __attribute__((overloadable)) shuffle2(ushort8 x, ushort8 y, ushort16 mask);
06939 ushort16 const_func __attribute__((overloadable)) shuffle2(ushort16 x, ushort16 y, ushort16 mask);
06940
06941 int16 const_func __attribute__((overloadable)) shuffle2(int2 x, int2 y, uint16 mask);
06942 int16 const_func __attribute__((overloadable)) shuffle2(int4 x, int4 y, uint16 mask);
06943 int16 const_func __attribute__((overloadable)) shuffle2(int8 x, int8 y, uint16 mask);
06944 int16 const_func __attribute__((overloadable)) shuffle2(int16 x, int16 y, uint16 mask);
06945
06946 uint16 const_func __attribute__((overloadable)) shuffle2(uint2 x, uint2 y, uint16 mask);
06947 uint16 const_func __attribute__((overloadable)) shuffle2(uint4 x, uint4 y, uint16 mask);
06948 uint16 const_func __attribute__((overloadable)) shuffle2(uint8 x, uint8 y, uint16 mask);
06949 uint16 const_func __attribute__((overloadable)) shuffle2(uint16 x, uint16 y, uint16 mask);
06950
06951 long16 const_func __attribute__((overloadable)) shuffle2(long2 x, long2 y, ulong16 mask);
06952 long16 const_func __attribute__((overloadable)) shuffle2(long4 x, long4 y, ulong16 mask);
06953 long16 const_func __attribute__((overloadable)) shuffle2(long8 x, long8 y, ulong16 mask);
06954 long16 const_func __attribute__((overloadable)) shuffle2(long16 x, long16 y, ulong16 mask);
06955
06956 ulong16 const_func __attribute__((overloadable)) shuffle2(ulong2 x, ulong2 y, ulong16 mask);
06957 ulong16 const_func __attribute__((overloadable)) shuffle2(ulong4 x, ulong4 y, ulong16 mask);
06958 ulong16 const_func __attribute__((overloadable)) shuffle2(ulong8 x, ulong8 y, ulong16 mask);
06959 ulong16 const_func __attribute__((overloadable)) shuffle2(ulong16 x, ulong16 y, ulong16 mask);
06960
06961 //half16 const_func __attribute__((overloadable)) shuffle2(half2 x, half2 y, ushort16 mask);
06962 //half16 const_func __attribute__((overloadable)) shuffle2(half4 x, half4 y, ushort16 mask);
06963 //half16 const_func __attribute__((overloadable)) shuffle2(half8 x, half8 y, ushort16 mask);
06964 //half16 const_func __attribute__((overloadable)) shuffle2(half16 x, half16 y, ushort16 mask);
06965
06966 float16 const_func __attribute__((overloadable)) shuffle2(float2 x, float2 y, uint16 mask);
06967 float16 const_func __attribute__((overloadable)) shuffle2(float4 x, float4 y, uint16 mask);
06968 float16 const_func __attribute__((overloadable)) shuffle2(float8 x, float8 y, uint16 mask);
06969 float16 const_func __attribute__((overloadable)) shuffle2(float16 x, float16 y, uint16 mask);
06970
06971 double16 const_func __attribute__((overloadable)) shuffle2(double2 x, double2 y, ulong16 mask);
```

```
06972 double16 const_func __attribute__((overloadable)) shuffle2(double4 x, double4 y, ulong16 mask);
06973 double16 const_func __attribute__((overloadable)) shuffle2(double8 x, double8 y, ulong16 mask);
06974 double16 const_func __attribute__((overloadable)) shuffle2(double16 x, double16 y, ulong16 mask);
06975
06976 // Built-in image functions
06977
06978 /**
06979 * Use the coordinate (x, y) to do an element lookup in
06980 * the 2D image object specified by image.
06981 * read_imagef returns floating-point values in the
06982 * range [0.0 ...  1.0] for image objects created with
06983 * image_channel_data_type set to one of the predefined
06984 * packed formats or CL_UNORM_INT8, or
06985 * CL_UNORM_INT16.
06986 * read_imagef returns floating-point values in the
06987 * range [-1.0 ...  1.0] for image objects created with
06988 * image_channel_data_type set to CL_SNORM_INT8,
06989 * or CL_SNORM_INT16.
06990 * read_imagef returns floating-point values for image
06991 * objects created with image_channel_data_type set to
06992 * CL_HALF_FLOAT or CL_FLOAT.
06993 * The read_imagef calls that take integer coordinates
06994 * must use a sampler with filter mode set to
06995 * CLK_FILTER_NEAREST, normalized coordinates set
06996 * to CLK_NORMALIZED_COORDS_FALSE and
06997 * addressing mode set to
06998 * CLK_ADDRESS_CLAMP_TO_EDGE,
06999 * CLK_ADDRESS_CLAMP or CLK_ADDRESS_NONE;
07000 * otherwise the values returned are undefined.
07001 * Values returned by read_imagef for image objects
07002 * with image_channel_data_type values not specified
07003 * in the description above are undefined.
07004 */
07005
07006 //
07007 // Addressing Mode.
07008 //
07009 #define CLK_ADDRESS_NONE                0
07010 #define CLK_ADDRESS_CLAMP_TO_EDGE       2
07011 #define CLK_ADDRESS_CLAMP               4
07012 #define CLK_ADDRESS_REPEAT              6
07013 #define CLK_ADDRESS_MIRRORED_REPEAT     8
07014
07015 //
07016 // Coordination Normalization
07017 //
07018 #define CLK_NORMALIZED_COORDS_FALSE     0
07019 #define CLK_NORMALIZED_COORDS_TRUE      1
07020
07021 //
07022 // Filtering Mode.
07023 //
07024 #define CLK_FILTER_NEAREST              0x10
07025 #define CLK_FILTER_LINEAR               0x20
07026
07027 float4 __attribute__((overloadable)) const_func read_imagef(__read_only image2d_t image, sampler_t
      sampler, int2 coord);
07028 float4 __attribute__((overloadable)) const_func read_imagef(__read_only image2d_t image, sampler_t
      sampler, float2 coord);
07029 //half4 __attribute__((overloadable)) const_func read_imageh(__read_only image2d_t image, sampler_t
      sampler, int2 coord);
07030 //half4 __attribute__((overloadable)) const_func read_imageh(__read_only image2d_t image, sampler_t
      sampler, float2 coord);
07031
07032 //float __attribute__((overloadable)) const_func read_imagef(__read_only image2d_depth_t image,
      sampler_t sampler, int2 coord);
07033 //float __attribute__((overloadable)) const_func read_imagef(__read_only image2d_depth_t image,
      sampler_t sampler, float2 coord);
07034 //float __attribute__((overloadable)) const_func read_imagef(__read_only image2d_depth_t image, int2
      coord);
07035 //float __attribute__((overloadable)) const_func read_imagef(__read_only image2d_array_depth_t image,
      sampler_t sampler, int4 coord);
07036 //float __attribute__((overloadable)) const_func read_imagef(__read_only image2d_array_depth_t image,
      sampler_t sampler, float4 coord);
07037 //float __attribute__((overloadable)) const_func read_imagef(__read_only image2d_array_depth_t image,
      int4 coord);
07038
07039 //void __attribute__((overloadable)) write_imagef(__write_only image2d_depth_t image, int2 coord,
      float depth);
07040 //void __attribute__((overloadable)) write_imagef (__write_only image2d_array_depth_t image, int4
      coord, float depth);
07041
07042 //int const_func __attribute__((overloadable)) get_image_width(image2d_depth_t image);
07043 //int const_func __attribute__((overloadable)) get_image_width(image2d_array_depth_t image);
07044
07045 //int const_func __attribute__((overloadable)) get_image_height(image2d_depth_t image);
07046 //int const_func __attribute__((overloadable)) get_image_height(image2d_array_depth_t image);
```

```
07047
07048 //int const_func __attribute__((overloadable)) get_image_channel_data_type(image2d_depth_t image);
07049 //int const_func __attribute__((overloadable)) get_image_channel_data_type(image2d_array_depth_t
      image);
07050
07051 //int const_func __attribute__((overloadable)) get_image_channel_order(image2d_depth_t image);
07052 //int const_func __attribute__((overloadable)) get_image_channel_order(image2d_array_depth_t image);
07053
07054 //int2 const_func __attribute__((overloadable)) get_image_dim(image2d_depth_t image);
07055 //int2 const_func __attribute__((overloadable)) get_image_dim(image2d_array_depth_t image);
07056
07057 //size_t const_func __attribute__((overloadable)) get_image_array_size(image2d_array_depth_t
      image_array);
07058
07059 /**
07060 * Use the coordinate (x, y) to do an element lookup in
07061 * the 2D image object specified by image.
07062 * read_imagei and read_imageui return
07063 * unnormalized signed integer and unsigned integer
07064 * values respectively.  Each channel will be stored in a
07065 * 32-bit integer.
07066 * read_imagei can only be used with image objects
07067 * created with image_channel_data_type set to one of
07068 * the following values:
07069 * CL_SIGNED_INT8,
07070 * CL_SIGNED_INT16 and
07071 * CL_SIGNED_INT32.
07072 * If the image_channel_data_type is not one of the
07073 * above values, the values returned by read_imagei
07074 * are undefined.
07075 * read_imageui can only be used with image objects
07076 * created with image_channel_data_type set to one of
07077 * the following values:
07078 * CL_UNSIGNED_INT8,
07079 * CL_UNSIGNED_INT16 and
07080 * CL_UNSIGNED_INT32.
07081 * If the image_channel_data_type is not one of the
07082 * above values, the values returned by read_imageui
07083 * are undefined.
07084 * The read_image{i|ui} calls support a nearest filter
07085 * only.  The filter_mode specified in sampler
07086 * must be set to CLK_FILTER_NEAREST; otherwise
07087 * the values returned are undefined.
07088 * Furthermore, the read_image{i|ui} calls that take
07089 * integer coordinates must use a sampler with
07090 * normalized coordinates set to
07091 * CLK_NORMALIZED_COORDS_FALSE and
07092 * addressing mode set to
07093 * CLK_ADDRESS_CLAMP_TO_EDGE,
07094 * CLK_ADDRESS_CLAMP or CLK_ADDRESS_NONE;
07095 * otherwise the values returned are undefined.
07096 */
07097 int4 __attribute__((overloadable)) const_func read_imagei(__read_only image2d_t image, sampler_t
      sampler, int2 coord);
07098 int4 __attribute__((overloadable)) const_func read_imagei(__read_only image2d_t image, sampler_t
      sampler, float2 coord);
07099 uint4 __attribute__((overloadable)) const_func read_imageui(__read_only image2d_t image, sampler_t
      sampler, int2 coord);
07100 uint4 __attribute__((overloadable)) const_func read_imageui(__read_only image2d_t image, sampler_t
      sampler, float2 coord);
07101
07102 /**
07103 * Write color value to location specified by coordinate
07104 * (x, y) in the 2D image object specified by image.
07105 * Appropriate data format conversion to the specified
07106 * image format is done before writing the color value.
07107 * x & y are considered to be unnormalized coordinates
07108 * and must be in the range 0 ...  image width - 1, and 0
07109 * ...  image height - 1.
07110 * write_imagef can only be used with image objects
07111 * created with image_channel_data_type set to one of
07112 * the pre-defined packed formats or set to
07113 * CL_SNORM_INT8, CL_UNORM_INT8,
07114 * CL_SNORM_INT16, CL_UNORM_INT16,
07115 * CL_HALF_FLOAT or CL_FLOAT. Appropriate data
07116 * format conversion will be done to convert channel
07117 * data from a floating-point value to actual data format
07118 * in which the channels are stored.
07119 * write_imagei can only be used with image objects
07120 * created with image_channel_data_type set to one of
07121 * the following values:
07122 * CL_SIGNED_INT8,
07123 * CL_SIGNED_INT16 and
07124 * CL_SIGNED_INT32.
07125 * write_imageui can only be used with image objects
07126 * created with image_channel_data_type set to one of
07127 * the following values:
```

```
07128 * CL_UNSIGNED_INT8,
07129 * CL_UNSIGNED_INT16 and
07130 * CL_UNSIGNED_INT32.
07131 * The behavior of write_imagef, write_imagei and
07132 * write_imageui for image objects created with
07133 * image_channel_data_type values not specified in
07134 * the description above or with (x, y) coordinate
07135 * values that are not in the range (0 ...  image width -
07136 * 1, 0 ...  image height - 1), respectively, is undefined.
07137 */
07138 void __attribute__((overloadable)) write_imagef(__write_only image2d_t image, int2 coord, float4
      color);
07139 void __attribute__((overloadable)) write_imagei(__write_only image2d_t image, int2 coord, int4 color);
07140 void __attribute__((overloadable)) write_imageui(__write_only image2d_t image, int2 coord, uint4
      color);
07141 void __attribute__((overloadable)) write_imageh(__write_only image2d_t image, int2 coord, half4
      color);
07142
07143 /**
07144 * Use the coordinate (coord.x, coord.y, coord.z) to do
07145 * an element lookup in the 3D image object specified
07146 * by image.  coord.w is ignored.
07147 * read_imagef returns floating-point values in the
07148 * range [0.0 ...  1.0] for image objects created with
07149 * image_channel_data_type set to one of the predefined
07150 * packed formats or CL_UNORM_INT8, or
07151 * CL_UNORM_INT16.
07152 * read_imagef returns floating-point values in the
07153 * range [-1.0 ...  1.0] for image objects created with
07154 * image_channel_data_type set to CL_SNORM_INT8,
07155 * or CL_SNORM_INT16.
07156 * read_imagef returns floating-point values for image
07157 * objects created with image_channel_data_type set to
07158 * CL_HALF_FLOAT or CL_FLOAT.
07159 * The read_imagef calls that take integer coordinates
07160 * must use a sampler with filter mode set to
07161 * CLK_FILTER_NEAREST, normalized coordinates set
07162 * to CLK_NORMALIZED_COORDS_FALSE and
07163 * addressing mode set to
07164 * CLK_ADDRESS_CLAMP_TO_EDGE,
07165 * CLK_ADDRESS_CLAMP or CLK_ADDRESS_NONE;
07166 * otherwise the values returned are undefined.
07167 * Values returned by read_imagef for image objects
07168 * with image_channel_data_type values not specified
07169 * in the description are undefined.
07170 */
07171 float4 __attribute__((overloadable)) const_func read_imagef(__read_only image3d_t image, sampler_t
      sampler, int4 coord);
07172 float4 __attribute__((overloadable)) const_func read_imagef(__read_only image3d_t image, sampler_t
      sampler, float4 coord);
07173
07174 /**
07175 * Use the coordinate (coord.x, coord.y, coord.z) to do
07176 * an element lookup in the 3D image object specified
07177 * by image.  coord.w is ignored.
07178 * read_imagei and read_imageui return
07179 * unnormalized signed integer and unsigned integer
07180 * values respectively.  Each channel will be stored in a
07181 * 32-bit integer.
07182 * read_imagei can only be used with image objects
07183 * created with image_channel_data_type set to one of
07184 * the following values:
07185 * CL_SIGNED_INT8,
07186 * CL_SIGNED_INT16 and
07187 * CL_SIGNED_INT32.
07188 * If the image_channel_data_type is not one of the
07189 * above values, the values returned by read_imagei
07190 * are undefined.
07191 * read_imageui can only be used with image objects
07192 * created with image_channel_data_type set to one of
07193 * the following values:
07194 * CL_UNSIGNED_INT8,
07195 * CL_UNSIGNED_INT16 and
07196 * CL_UNSIGNED_INT32.
07197 * If the image_channel_data_type is not one of the
07198 * above values, the values returned by read_imageui
07199 * are undefined.
07200 * The read_image{i|ui} calls support a nearest filter
07201 * only.  The filter_mode specified in sampler
07202 * must be set to CLK_FILTER_NEAREST; otherwise
07203 * the values returned are undefined.
07204 * Furthermore, the read_image{i|ui} calls that take
07205 * integer coordinates must use a sampler with
07206 * normalized coordinates set to
07207 * CLK_NORMALIZED_COORDS_FALSE and
07208 * addressing mode set to
07209 * CLK_ADDRESS_CLAMP_TO_EDGE,
```

```
07210 * CLK_ADDRESS_CLAMP or CLK_ADDRESS_NONE;
07211 * otherwise the values returned are undefined.
07212 */
07213 int4 __attribute__((overloadable)) const_func read_imagei(__read_only image3d_t image, sampler_t
      sampler, int4 coord);
07214 int4 __attribute__((overloadable)) const_func read_imagei(__read_only image3d_t image, sampler_t
      sampler, float4 coord);
07215 uint4 __attribute__((overloadable)) const_func read_imageui(__read_only image3d_t image, sampler_t
      sampler, int4 coord);
07216 uint4 __attribute__((overloadable)) const_func read_imageui(__read_only image3d_t image, sampler_t
      sampler, float4 coord);
07217
07218 // IMAGE 1.2 built-ins
07219 // with samplers and samplerless
07220 float4  __attribute__((overloadable)) const_func read_imagef(__read_only image2d_array_t image,
      sampler_t sampler, int4 coord);
07221 float4  __attribute__((overloadable)) const_func read_imagef(__read_only image2d_array_t image,
      sampler_t sampler, float4 coord);
07222 int4  __attribute__((overloadable)) const_func read_imagei(__read_only image2d_array_t image,
      sampler_t sampler, int4 coord);
07223 int4  __attribute__((overloadable)) const_func read_imagei(__read_only image2d_array_t image,
      sampler_t sampler, float4 coord);
07224 uint4  __attribute__((overloadable)) const_func read_imageui(__read_only image2d_array_t image,
      sampler_t sampler, int4 coord);
07225 uint4  __attribute__((overloadable)) const_func read_imageui(__read_only image2d_array_t image,
      sampler_t sampler, float4 coord);
07226 float4 __attribute__((overloadable)) const_func read_imagef(__read_only image1d_t image, sampler_t
      sampler, int coord);
07227 float4 __attribute__((overloadable)) const_func read_imagef(__read_only image1d_t image, sampler_t
      sampler, float coord);
07228 int4 __attribute__((overloadable)) const_func read_imagei(__read_only image1d_t image, sampler_t
      sampler, int coord);
07229 int4 __attribute__((overloadable)) const_func read_imagei(__read_only image1d_t image, sampler_t
      sampler, float coord);
07230 uint4 __attribute__((overloadable)) const_func read_imageui(__read_only image1d_t image, sampler_t
      sampler, int coord);
07231 uint4 __attribute__((overloadable)) const_func read_imageui(__read_only image1d_t image, sampler_t
      sampler, float coord);
07232 float4 __attribute__((overloadable)) const_func read_imagef(__read_only image1d_array_t image,
      sampler_t sampler, int2 coord);
07233 float4 __attribute__((overloadable)) const_func read_imagef(__read_only image1d_array_t image,
      sampler_t sampler, float2 coord);
07234 int4 __attribute__((overloadable)) const_func read_imagei(__read_only image1d_array_t image, sampler_t
      sampler, int2 coord);
07235 int4 __attribute__((overloadable)) const_func read_imagei(__read_only image1d_array_t image, sampler_t
      sampler, float2 coord);
07236 uint4 __attribute__((overloadable)) const_func read_imageui(__read_only image1d_array_t image,
      sampler_t sampler, int2 coord);
07237 uint4 __attribute__((overloadable)) const_func read_imageui(__read_only image1d_array_t image,
      sampler_t sampler, float2 coord);
07238 float4 __attribute__((overloadable)) const_func read_imagef (__read_only image2d_t image, int2 coord);
07239 int4 __attribute__((overloadable)) const_func read_imagei (__read_only image2d_t image, int2 coord);
07240 uint4 __attribute__((overloadable)) const_func read_imageui (__read_only image2d_t image, int2 coord);
07241 float4 __attribute__((overloadable)) const_func read_imagef (__read_only image3d_t image, int4 coord);
07242 int4 __attribute__((overloadable)) const_func read_imagei (__read_only image3d_t image, int4 coord);
07243 uint4 __attribute__((overloadable)) const_func read_imageui (__read_only image3d_t image, int4 coord);
07244 float4 __attribute__((overloadable)) const_func read_imagef (__read_only image2d_array_t image, int4
      coord);
07245 int4 __attribute__((overloadable)) const_func read_imagei (__read_only image2d_array_t image, int4
      coord);
07246 uint4 __attribute__((overloadable)) const_func read_imageui (__read_only image2d_array_t image, int4
      coord);
07247 float4 __attribute__((overloadable)) const_func read_imagef (__read_only image1d_t image, int coord);
07248 float4 __attribute__((overloadable)) const_func read_imagef (__read_only image1d_buffer_t image, int
      coord);
07249 int4 __attribute__((overloadable)) const_func read_imagei(__read_only image1d_t image, int coord);
07250 uint4 __attribute__((overloadable)) const_func read_imageui(__read_only image1d_t image, int coord);
07251 int4 __attribute__((overloadable)) const_func read_imagei(__read_only image1d_buffer_t image, int
      coord);
07252 uint4 __attribute__((overloadable)) const_func read_imageui(__read_only image1d_buffer_t image, int
      coord);
07253 float4 __attribute__((overloadable)) const_func read_imagef(__read_only image1d_array_t image, int2
      coord);
07254 int4 __attribute__((overloadable)) const_func read_imagei(__read_only image1d_array_t image, int2
      coord);
07255 uint4 __attribute__((overloadable)) const_func read_imageui(__read_only image1d_array_t image, int2
      coord);
07256 void __attribute__((overloadable)) write_imagef (__write_only image2d_array_t image, int4 coord,
      float4 color);
07257 void __attribute__((overloadable)) write_imagei (__write_only image2d_array_t image, int4 coord, int4
      color);
07258 void __attribute__((overloadable)) write_imageui (__write_only image2d_array_t image, int4 coord,
      uint4 color);
07259 void __attribute__((overloadable)) write_imagef (__write_only image1d_t image, int coord, float4
      color);
07260 void __attribute__((overloadable)) write_imagei (__write_only image1d_t image, int coord, int4 color);
07261 void __attribute__((overloadable)) write_imageui (__write_only image1d_t image, int coord, uint4
```

```
      color);
07262 void __attribute__((overloadable)) write_imagef (__write_only image1d_buffer_t image, int coord,
      float4 color);
07263 void __attribute__((overloadable)) write_imagei (__write_only image1d_buffer_t image, int coord, int4
      color);
07264 void __attribute__((overloadable)) write_imageui (__write_only image1d_buffer_t image, int coord,
      uint4 color);
07265 void __attribute__((overloadable)) write_imagef (__write_only image1d_array_t image, int2 coord,
      float4 color);
07266 void __attribute__((overloadable)) write_imagei (__write_only image1d_array_t image, int2 coord, int4
      color);
07267 void __attribute__((overloadable)) write_imageui (__write_only image1d_array_t image, int2 coord,
      uint4 color);
07268
07269 /**
07270 * Write color value to location specified by coordinate
07271 * (x, y, z) in the 3D image object specified by image.
07272 * Appropriate data format conversion to the specified
07273 * image format is done before writing the color value.
07274 * x & y are considered to be unnormalized coordinates
07275 * and must be in the range 0 ...  image width - 1, and 0
07276 * ...  image height - 1.
07277 * write_imagef can only be used with image objects
07278 * created with image_channel_data_type set to one of
07279 * the pre-defined packed formats or set to
07280 * CL_SNORM_INT8, CL_UNORM_INT8,
07281 * CL_SNORM_INT16, CL_UNORM_INT16,
07282 * CL_HALF_FLOAT or CL_FLOAT. Appropriate data
07283 * format conversion will be done to convert channel
07284 * data from a floating-point value to actual data format
07285 * in which the channels are stored.
07286 * write_imagei can only be used with image objects
07287 * created with image_channel_data_type set to one of
07288 * the following values:
07289 * CL_SIGNED_INT8,
07290 * CL_SIGNED_INT16 and
07291 * CL_SIGNED_INT32.
07292 * write_imageui can only be used with image objects
07293 * created with image_channel_data_type set to one of
07294 * the following values:
07295 * CL_UNSIGNED_INT8,
07296 * CL_UNSIGNED_INT16 and
07297 * CL_UNSIGNED_INT32.
07298 * The behavior of write_imagef, write_imagei and
07299 * write_imageui for image objects created with
07300 * image_channel_data_type values not specified in
07301 * the description above or with (x, y) coordinate
07302 * values that are not in the range (0 ...  image width -
07303 * 1, 0 ...  image height - 1), respectively, is undefined.
07304 */
07305 void __attribute__((overloadable)) write_imagef(__write_only image3d_t image, int4 coord, float4
      color);
07306 void __attribute__((overloadable)) write_imagei(__write_only image3d_t image, int4 coord, int4 color);
07307 void __attribute__((overloadable)) write_imageui(__write_only image3d_t image, int4 coord, uint4
      color);
07308 void __attribute__((overloadable)) write_imageh(__write_only image3d_t image, int4 coord, half4
      color);
07309
07310 /**
07311 * Return the image width in pixels.
07312 */
07313 int const_func __attribute__((overloadable)) get_image_width(image1d_t image);
07314 int const_func __attribute__((overloadable)) get_image_width(image1d_buffer_t image);
07315 int const_func __attribute__((overloadable)) get_image_width(image2d_t image);
07316 int const_func __attribute__((overloadable)) get_image_width(image3d_t image);
07317 int const_func __attribute__((overloadable)) get_image_width(image1d_array_t image);
07318 int const_func __attribute__((overloadable)) get_image_width(image2d_array_t image);
07319
07320 /**
07321 * Return the image height in pixels.
07322 */
07323 int const_func __attribute__((overloadable)) get_image_height(image2d_t image);
07324 int const_func __attribute__((overloadable)) get_image_height(image3d_t image);
07325 int const_func __attribute__((overloadable)) get_image_height(image2d_array_t image);
07326
07327 /**
07328 * Return the image depth in pixels.
07329 */
07330 int const_func __attribute__((overloadable)) get_image_depth(image3d_t image);
07331
07332 /**
07333 * Return the channel data type.  Valid values are:
07334 * CLK_SNORM_INT8
07335 * CLK_SNORM_INT16
07336 * CLK_UNORM_INT8
07337 * CLK_UNORM_INT16
07338 * CLK_UNORM_SHORT_565
```

```
07339 * CLK_UNORM_SHORT_555
07340 * CLK_UNORM_SHORT_101010
07341 * CLK_SIGNED_INT8
07342 * CLK_SIGNED_INT16
07343 * CLK_SIGNED_INT32
07344 * CLK_UNSIGNED_INT8
07345 * CLK_UNSIGNED_INT16
07346 * CLK_UNSIGNED_INT32
07347 * CLK_HALF_FLOAT
07348 * CLK_FLOAT
07349 */
07350
07351 // Channel order, numbering must be aligned with cl_channel_order in cl.h
07352 //
07353 // Channel order.
07354 //
07355 #define CLK_R                                           0x10B0
07356 #define CLK_A                                           0x10B1
07357 #define CLK_RG                                          0x10B2
07358 #define CLK_RA                                          0x10B3
07359 #define CLK_RGB                                         0x10B4
07360 #define CLK_RGBA                                        0x10B5
07361 #define CLK_BGRA                                        0x10B6
07362 #define CLK_ARGB                                        0x10B7
07363 #define CLK_INTENSITY                                   0x10B8
07364 #define CLK_LUMINANCE                                   0x10B9
07365 #define CLK_Rx                                          0x10BA
07366 #define CLK_RGx                                         0x10BB
07367 #define CLK_RGBx                                        0x10BC
07368 #define CLK_DEPTH                                       0x10BD
07369 #define CLK_DEPTH_STENCIL                               0x10BE
07370
07371 //
07372 // Channel Datatype.
07373 //
07374 #define CLK_SNORM_INT8          0x10D0
07375 #define CLK_SNORM_INT16         0x10D1
07376 #define CLK_UNORM_INT8          0x10D2
07377 #define CLK_UNORM_INT16         0x10D3
07378 #define CLK_UNORM_SHORT_565     0x10D4
07379 #define CLK_UNORM_SHORT_555     0x10D5
07380 #define CLK_UNORM_INT_101010    0x10D6
07381 #define CLK_SIGNED_INT8         0x10D7
07382 #define CLK_SIGNED_INT16        0x10D8
07383 #define CLK_SIGNED_INT32        0x10D9
07384 #define CLK_UNSIGNED_INT8       0x10DA
07385 #define CLK_UNSIGNED_INT16      0x10DB
07386 #define CLK_UNSIGNED_INT32      0x10DC
07387 #define CLK_HALF_FLOAT          0x10DD
07388 #define CLK_FLOAT               0x10DE
07389 #define CLK_UNORM_INT24         0x10DF
07390
07391 int const_func __attribute__((overloadable)) get_image_channel_data_type(image1d_t image);
07392 int const_func __attribute__((overloadable)) get_image_channel_data_type(image1d_buffer_t image);
07393 int const_func __attribute__((overloadable)) get_image_channel_data_type(image2d_t image);
07394 int const_func __attribute__((overloadable)) get_image_channel_data_type(image3d_t image);
07395 int const_func __attribute__((overloadable)) get_image_channel_data_type(image1d_array_t image);
07396 int const_func __attribute__((overloadable)) get_image_channel_data_type(image2d_array_t image);
07397
07398 /**
07399 * Return the image channel order.  Valid values are:
07400 * CLK_A
07401 * CLK_R
07402 * CLK_Rx
07403 * CLK_RG
07404 * CLK_RGx
07405 * CLK_RA
07406 * CLK_RGB
07407 * CLK_RGBx
07408 * CLK_RGBA
07409 * CLK_ARGB
07410 * CLK_BGRA
07411 * CLK_INTENSITY
07412 * CLK_LUMINANCE
07413 */
07414 int const_func __attribute__((overloadable)) get_image_channel_order(image1d_t image);
07415 int const_func __attribute__((overloadable)) get_image_channel_order(image1d_buffer_t image);
07416 int const_func __attribute__((overloadable)) get_image_channel_order(image2d_t image);
07417 int const_func __attribute__((overloadable)) get_image_channel_order(image3d_t image);
07418 int const_func __attribute__((overloadable)) get_image_channel_order(image1d_array_t image);
07419 int const_func __attribute__((overloadable)) get_image_channel_order(image2d_array_t image);
07420
07421 /**
07422 * Return the 2D image width and height as an int2
07423 * type.  The width is returned in the x component, and
07424 * the height in the y component.
07425 */
```

```
07426 int2 const_func __attribute__((overloadable)) get_image_dim(image2d_t image);
07427 int2 const_func __attribute__((overloadable)) get_image_dim(image2d_array_t image);
07428
07429 /**
07430 * Return the 3D image width, height, and depth as an
07431 * int4 type.  The width is returned in the x
07432 * component, height in the y component, depth in the z
07433 * component and the w component is 0.
07434 */
07435 int4 const_func __attribute__((overloadable)) get_image_dim(image3d_t image);
07436
07437 /**
07438 * Return the number of images in the 2D image array.
07439 */
07440 size_t const_func __attribute__((overloadable)) get_image_array_size(image2d_array_t image_array);
07441
07442 /**
07443 * Return the number of images in the 1D image array.
07444 */
07445 size_t const_func __attribute__((overloadable)) get_image_array_size(image1d_array_t image_array);
07446
07447 /**
07448 *  Use coord.xy to do an element lookup in the 2D image layer identified by index coord.z in the 2D
      image array.
07449 */
07450 float4 const_func __attribute__((overloadable)) read_imagef(__read_only image2d_array_t image_array,
      sampler_t sampler, int4 coord);
07451
07452 /**
07453 * Use coord.xy to do an element lookup in the 2D image layer identified by index coord.z in the 2D
      image array.
07454 */
07455 float4 const_func __attribute__((overloadable)) read_imagef(__read_only image2d_array_t image_array,
      sampler_t sampler, float4 coord);
07456
07457 /**
07458 * Use coord.xy to do an element lookup in the 2D image layer identified by index coord.z in the 2D
      image array.
07459 */
07460 int4 const_func __attribute__((overloadable)) read_imagei(__read_only image2d_array_t image_array,
      sampler_t sampler, int4 coord);
07461
07462 /**
07463 * Use coord.xy to do an element lookup in the 2D image layer identified by index coord.z in the 2D
      image array.
07464 */
07465 int4 const_func __attribute__((overloadable)) read_imagei(__read_only image2d_array_t image_array,
      sampler_t sampler, float4 coord);
07466
07467 /**
07468 * Use coord.xy to do an element lookup in the 2D image layer identified by index coord.z in the 2D
      image array.
07469 */
07470 uint4 const_func __attribute__((overloadable)) read_imageui(__read_only image2d_array_t image_array,
      sampler_t sampler, int4 coord);
07471
07472 /**
07473 * Use coord.xy to do an element lookup in the 2D image layer identified by index coord.z in the 2D
      image array.
07474 */
07475 uint4 const_func __attribute__((overloadable)) read_imageui(__read_only image2d_array_t image_array,
      sampler_t sampler, float4 coord);
07476
07477 /**
07478 * Write color value to location specified by coord.xy in the 2D image layer identified by index
      coord.z in the 2D image array.
07479 */
07480 void __attribute__((overloadable)) write_imagef(__write_only image2d_array_t image_array, int4 coord,
      float4 color);
07481
07482 /**
07483 * Write color value to location specified by coord.xy in the 2D image layer identified by index
      coord.z in the 2D image array.
07484 */
07485 void __attribute__((overloadable)) write_imagei(__write_only image2d_array_t image_array, int4 coord,
      int4 color);
07486
07487 /**
07488 * Write color value to location specified by coord.xy in the 2D image layer identified by index
      coord.z in the 2D image array.
07489 */
07490 void __attribute__((overloadable)) write_imageui(__write_only image2d_array_t image_array, int4 coord,
      uint4 color);
07491
07492 /**
07493 * OpenCL as_typen operators
07494 * Reinterprets a data type as another data type of the same size
```

```
07495 */
07496
07497 #define as_char(x) __builtin_astype((x), char)
07498 #define as_char2(x) __builtin_astype((x), char2)
07499 #define as_char3(x) __builtin_astype((x), char3)
07500 #define as_char4(x) __builtin_astype((x), char4)
07501 #define as_char8(x) __builtin_astype((x), char8)
07502 #define as_char16(x) __builtin_astype((x), char16)
07503
07504 #define as_uchar(x) __builtin_astype((x), uchar)
07505 #define as_uchar2(x) __builtin_astype((x), uchar2)
07506 #define as_uchar3(x) __builtin_astype((x), uchar3)
07507 #define as_uchar4(x) __builtin_astype((x), uchar4)
07508 #define as_uchar8(x) __builtin_astype((x), uchar8)
07509 #define as_uchar16(x) __builtin_astype((x), uchar16)
07510
07511 #define as_short(x) __builtin_astype((x), short)
07512 #define as_short2(x) __builtin_astype((x), short2)
07513 #define as_short3(x) __builtin_astype((x), short3)
07514 #define as_short4(x) __builtin_astype((x), short4)
07515 #define as_short8(x) __builtin_astype((x), short8)
07516 #define as_short16(x) __builtin_astype((x), short16)
07517
07518 #define as_ushort(x) __builtin_astype((x), ushort)
07519 #define as_ushort2(x) __builtin_astype((x), ushort2)
07520 #define as_ushort3(x) __builtin_astype((x), ushort3)
07521 #define as_ushort4(x) __builtin_astype((x), ushort4)
07522 #define as_ushort8(x) __builtin_astype((x), ushort8)
07523 #define as_ushort16(x) __builtin_astype((x), ushort16)
07524
07525 #define as_int(x) __builtin_astype((x), int)
07526 #define as_int2(x) __builtin_astype((x), int2)
07527 #define as_int3(x) __builtin_astype((x), int3)
07528 #define as_int4(x) __builtin_astype((x), int4)
07529 #define as_int8(x) __builtin_astype((x), int8)
07530 #define as_int16(x) __builtin_astype((x), int16)
07531
07532 #define as_uint(x) __builtin_astype((x), uint)
07533 #define as_uint2(x) __builtin_astype((x), uint2)
07534 #define as_uint3(x) __builtin_astype((x), uint3)
07535 #define as_uint4(x) __builtin_astype((x), uint4)
07536 #define as_uint8(x) __builtin_astype((x), uint8)
07537 #define as_uint16(x) __builtin_astype((x), uint16)
07538
07539 #define as_long(x) __builtin_astype((x), long)
07540 #define as_long2(x) __builtin_astype((x), long2)
07541 #define as_long3(x) __builtin_astype((x), long3)
07542 #define as_long4(x) __builtin_astype((x), long4)
07543 #define as_long8(x) __builtin_astype((x), long8)
07544 #define as_long16(x) __builtin_astype((x), long16)
07545
07546 #define as_ulong(x) __builtin_astype((x), ulong)
07547 #define as_ulong2(x) __builtin_astype((x), ulong2)
07548 #define as_ulong3(x) __builtin_astype((x), ulong3)
07549 #define as_ulong4(x) __builtin_astype((x), ulong4)
07550 #define as_ulong8(x) __builtin_astype((x), ulong8)
07551 #define as_ulong16(x) __builtin_astype((x), ulong16)
07552
07553 #define as_half(x) __builtin_astype((x), half)
07554 #define as_half2(x) __builtin_astype((x), half2)
07555 #define as_half3(x) __builtin_astype((x), half3)
07556 #define as_half4(x) __builtin_astype((x), half4)
07557 #define as_half8(x) __builtin_astype((x), half8)
07558 #define as_half16(x) __builtin_astype((x), half16)
07559
07560 #define as_float(x) __builtin_astype((x), float)
07561 #define as_float2(x) __builtin_astype((x), float2)
07562 #define as_float3(x) __builtin_astype((x), float3)
07563 #define as_float4(x) __builtin_astype((x), float4)
07564 #define as_float8(x) __builtin_astype((x), float8)
07565 #define as_float16(x) __builtin_astype((x), float16)
07566
07567 #define as_double(x) __builtin_astype((x), double)
07568 #define as_double2(x) __builtin_astype((x), double2)
07569 #define as_double3(x) __builtin_astype((x), double3)
07570 #define as_double4(x) __builtin_astype((x), double4)
07571 #define as_double8(x) __builtin_astype((x), double8)
07572 #define as_double16(x) __builtin_astype((x), double16)
07573
07574 // Explicit conversions
07575
07576 char const_func __attribute__((overloadable)) convert_char_rte(char);
07577 char const_func __attribute__((overloadable)) convert_char_sat_rte(char);
07578 char const_func __attribute__((overloadable)) convert_char_rtz(char);
07579 char const_func __attribute__((overloadable)) convert_char_sat_rtz(char);
07580 char const_func __attribute__((overloadable)) convert_char_rtp(char);
07581 char const_func __attribute__((overloadable)) convert_char_sat_rtp(char);
```

```
07582 char const_func __attribute__((overloadable)) convert_char_rtn(char);
07583 char const_func __attribute__((overloadable)) convert_char_sat_rtn(char);
07584 char const_func __attribute__((overloadable)) convert_char(char);
07585 char const_func __attribute__((overloadable)) convert_char_sat(char);
07586 char const_func __attribute__((overloadable)) convert_char_rte(uchar);
07587 char const_func __attribute__((overloadable)) convert_char_sat_rte(uchar);
07588 char const_func __attribute__((overloadable)) convert_char_rtz(uchar);
07589 char const_func __attribute__((overloadable)) convert_char_sat_rtz(uchar);
07590 char const_func __attribute__((overloadable)) convert_char_rtp(uchar);
07591 char const_func __attribute__((overloadable)) convert_char_sat_rtp(uchar);
07592 char const_func __attribute__((overloadable)) convert_char_rtn(uchar);
07593 char const_func __attribute__((overloadable)) convert_char_sat_rtn(uchar);
07594 char const_func __attribute__((overloadable)) convert_char(uchar);
07595 char const_func __attribute__((overloadable)) convert_char_sat(uchar);
07596 char const_func __attribute__((overloadable)) convert_char_rte(short);
07597 char const_func __attribute__((overloadable)) convert_char_sat_rte(short);
07598 char const_func __attribute__((overloadable)) convert_char_rtz(short);
07599 char const_func __attribute__((overloadable)) convert_char_sat_rtz(short);
07600 char const_func __attribute__((overloadable)) convert_char_rtp(short);
07601 char const_func __attribute__((overloadable)) convert_char_sat_rtp(short);
07602 char const_func __attribute__((overloadable)) convert_char_rtn(short);
07603 char const_func __attribute__((overloadable)) convert_char_sat_rtn(short);
07604 char const_func __attribute__((overloadable)) convert_char(short);
07605 char const_func __attribute__((overloadable)) convert_char_sat(short);
07606 char const_func __attribute__((overloadable)) convert_char_rte(ushort);
07607 char const_func __attribute__((overloadable)) convert_char_sat_rte(ushort);
07608 char const_func __attribute__((overloadable)) convert_char_rtz(ushort);
07609 char const_func __attribute__((overloadable)) convert_char_sat_rtz(ushort);
07610 char const_func __attribute__((overloadable)) convert_char_rtp(ushort);
07611 char const_func __attribute__((overloadable)) convert_char_sat_rtp(ushort);
07612 char const_func __attribute__((overloadable)) convert_char_rtn(ushort);
07613 char const_func __attribute__((overloadable)) convert_char_sat_rtn(ushort);
07614 char const_func __attribute__((overloadable)) convert_char(ushort);
07615 char const_func __attribute__((overloadable)) convert_char_sat(ushort);
07616 char const_func __attribute__((overloadable)) convert_char_rte(int);
07617 char const_func __attribute__((overloadable)) convert_char_sat_rte(int);
07618 char const_func __attribute__((overloadable)) convert_char_rtz(int);
07619 char const_func __attribute__((overloadable)) convert_char_sat_rtz(int);
07620 char const_func __attribute__((overloadable)) convert_char_rtp(int);
07621 char const_func __attribute__((overloadable)) convert_char_sat_rtp(int);
07622 char const_func __attribute__((overloadable)) convert_char_rtn(int);
07623 char const_func __attribute__((overloadable)) convert_char_sat_rtn(int);
07624 char const_func __attribute__((overloadable)) convert_char(int);
07625 char const_func __attribute__((overloadable)) convert_char_sat(int);
07626 char const_func __attribute__((overloadable)) convert_char_rte(uint);
07627 char const_func __attribute__((overloadable)) convert_char_sat_rte(uint);
07628 char const_func __attribute__((overloadable)) convert_char_rtz(uint);
07629 char const_func __attribute__((overloadable)) convert_char_sat_rtz(uint);
07630 char const_func __attribute__((overloadable)) convert_char_rtp(uint);
07631 char const_func __attribute__((overloadable)) convert_char_sat_rtp(uint);
07632 char const_func __attribute__((overloadable)) convert_char_rtn(uint);
07633 char const_func __attribute__((overloadable)) convert_char_sat_rtn(uint);
07634 char const_func __attribute__((overloadable)) convert_char(uint);
07635 char const_func __attribute__((overloadable)) convert_char_sat(uint);
07636 char const_func __attribute__((overloadable)) convert_char_rte(long);
07637 char const_func __attribute__((overloadable)) convert_char_sat_rte(long);
07638 char const_func __attribute__((overloadable)) convert_char_rtz(long);
07639 char const_func __attribute__((overloadable)) convert_char_sat_rtz(long);
07640 char const_func __attribute__((overloadable)) convert_char_rtp(long);
07641 char const_func __attribute__((overloadable)) convert_char_sat_rtp(long);
07642 char const_func __attribute__((overloadable)) convert_char_rtn(long);
07643 char const_func __attribute__((overloadable)) convert_char_sat_rtn(long);
07644 char const_func __attribute__((overloadable)) convert_char(long);
07645 char const_func __attribute__((overloadable)) convert_char_sat(long);
07646 char const_func __attribute__((overloadable)) convert_char_rte(ulong);
07647 char const_func __attribute__((overloadable)) convert_char_sat_rte(ulong);
07648 char const_func __attribute__((overloadable)) convert_char_rtz(ulong);
07649 char const_func __attribute__((overloadable)) convert_char_sat_rtz(ulong);
07650 char const_func __attribute__((overloadable)) convert_char_rtp(ulong);
07651 char const_func __attribute__((overloadable)) convert_char_sat_rtp(ulong);
07652 char const_func __attribute__((overloadable)) convert_char_rtn(ulong);
07653 char const_func __attribute__((overloadable)) convert_char_sat_rtn(ulong);
07654 char const_func __attribute__((overloadable)) convert_char(ulong);
07655 char const_func __attribute__((overloadable)) convert_char_sat(ulong);
07656 char const_func __attribute__((overloadable)) convert_char_rte(float);
07657 char const_func __attribute__((overloadable)) convert_char_sat_rte(float);
07658 char const_func __attribute__((overloadable)) convert_char_rtz(float);
07659 char const_func __attribute__((overloadable)) convert_char_sat_rtz(float);
07660 char const_func __attribute__((overloadable)) convert_char_rtp(float);
07661 char const_func __attribute__((overloadable)) convert_char_sat_rtp(float);
07662 char const_func __attribute__((overloadable)) convert_char_rtn(float);
07663 char const_func __attribute__((overloadable)) convert_char_sat_rtn(float);
07664 char const_func __attribute__((overloadable)) convert_char(float);
07665 char const_func __attribute__((overloadable)) convert_char_sat(float);
07666 char const_func __attribute__((overloadable)) convert_char_rte(double);
07667 char const_func __attribute__((overloadable)) convert_char_sat_rte(double);
07668 char const_func __attribute__((overloadable)) convert_char_rtz(double);
```

```
07669 char const_func __attribute__((overloadable)) convert_char_sat_rtz(double);
07670 char const_func __attribute__((overloadable)) convert_char_rtp(double);
07671 char const_func __attribute__((overloadable)) convert_char_sat_rtp(double);
07672 char const_func __attribute__((overloadable)) convert_char_rtn(double);
07673 char const_func __attribute__((overloadable)) convert_char_sat_rtn(double);
07674 char const_func __attribute__((overloadable)) convert_char(double);
07675 char const_func __attribute__((overloadable)) convert_char_sat(double);
07676 uchar const_func __attribute__((overloadable)) convert_uchar_rte(char);
07677 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rte(char);
07678 uchar const_func __attribute__((overloadable)) convert_uchar_rtz(char);
07679 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtz(char);
07680 uchar const_func __attribute__((overloadable)) convert_uchar_rtp(char);
07681 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtp(char);
07682 uchar const_func __attribute__((overloadable)) convert_uchar_rtn(char);
07683 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtn(char);
07684 uchar const_func __attribute__((overloadable)) convert_uchar(char);
07685 uchar const_func __attribute__((overloadable)) convert_uchar_sat(char);
07686 uchar const_func __attribute__((overloadable)) convert_uchar_rte(uchar);
07687 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rte(uchar);
07688 uchar const_func __attribute__((overloadable)) convert_uchar_rtz(uchar);
07689 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtz(uchar);
07690 uchar const_func __attribute__((overloadable)) convert_uchar_rtp(uchar);
07691 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtp(uchar);
07692 uchar const_func __attribute__((overloadable)) convert_uchar_rtn(uchar);
07693 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtn(uchar);
07694 uchar const_func __attribute__((overloadable)) convert_uchar(uchar);
07695 uchar const_func __attribute__((overloadable)) convert_uchar_sat(uchar);
07696 uchar const_func __attribute__((overloadable)) convert_uchar_rte(short);
07697 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rte(short);
07698 uchar const_func __attribute__((overloadable)) convert_uchar_rtz(short);
07699 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtz(short);
07700 uchar const_func __attribute__((overloadable)) convert_uchar_rtp(short);
07701 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtp(short);
07702 uchar const_func __attribute__((overloadable)) convert_uchar_rtn(short);
07703 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtn(short);
07704 uchar const_func __attribute__((overloadable)) convert_uchar(short);
07705 uchar const_func __attribute__((overloadable)) convert_uchar_sat(short);
07706 uchar const_func __attribute__((overloadable)) convert_uchar_rte(ushort);
07707 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rte(ushort);
07708 uchar const_func __attribute__((overloadable)) convert_uchar_rtz(ushort);
07709 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtz(ushort);
07710 uchar const_func __attribute__((overloadable)) convert_uchar_rtp(ushort);
07711 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtp(ushort);
07712 uchar const_func __attribute__((overloadable)) convert_uchar_rtn(ushort);
07713 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtn(ushort);
07714 uchar const_func __attribute__((overloadable)) convert_uchar(ushort);
07715 uchar const_func __attribute__((overloadable)) convert_uchar_sat(ushort);
07716 uchar const_func __attribute__((overloadable)) convert_uchar_rte(int);
07717 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rte(int);
07718 uchar const_func __attribute__((overloadable)) convert_uchar_rtz(int);
07719 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtz(int);
07720 uchar const_func __attribute__((overloadable)) convert_uchar_rtp(int);
07721 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtp(int);
07722 uchar const_func __attribute__((overloadable)) convert_uchar_rtn(int);
07723 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtn(int);
07724 uchar const_func __attribute__((overloadable)) convert_uchar(int);
07725 uchar const_func __attribute__((overloadable)) convert_uchar_sat(int);
07726 uchar const_func __attribute__((overloadable)) convert_uchar_rte(uint);
07727 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rte(uint);
07728 uchar const_func __attribute__((overloadable)) convert_uchar_rtz(uint);
07729 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtz(uint);
07730 uchar const_func __attribute__((overloadable)) convert_uchar_rtp(uint);
07731 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtp(uint);
07732 uchar const_func __attribute__((overloadable)) convert_uchar_rtn(uint);
07733 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtn(uint);
07734 uchar const_func __attribute__((overloadable)) convert_uchar(uint);
07735 uchar const_func __attribute__((overloadable)) convert_uchar_sat(uint);
07736 uchar const_func __attribute__((overloadable)) convert_uchar_rte(long);
07737 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rte(long);
07738 uchar const_func __attribute__((overloadable)) convert_uchar_rtz(long);
07739 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtz(long);
07740 uchar const_func __attribute__((overloadable)) convert_uchar_rtp(long);
07741 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtp(long);
07742 uchar const_func __attribute__((overloadable)) convert_uchar_rtn(long);
07743 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtn(long);
07744 uchar const_func __attribute__((overloadable)) convert_uchar(long);
07745 uchar const_func __attribute__((overloadable)) convert_uchar_sat(long);
07746 uchar const_func __attribute__((overloadable)) convert_uchar_rte(ulong);
07747 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rte(ulong);
07748 uchar const_func __attribute__((overloadable)) convert_uchar_rtz(ulong);
07749 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtz(ulong);
07750 uchar const_func __attribute__((overloadable)) convert_uchar_rtp(ulong);
07751 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtp(ulong);
07752 uchar const_func __attribute__((overloadable)) convert_uchar_rtn(ulong);
07753 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtn(ulong);
07754 uchar const_func __attribute__((overloadable)) convert_uchar(ulong);
07755 uchar const_func __attribute__((overloadable)) convert_uchar_sat(ulong);
```

```
07756 uchar const_func __attribute__((overloadable)) convert_uchar_rte(float);
07757 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rte(float);
07758 uchar const_func __attribute__((overloadable)) convert_uchar_rtz(float);
07759 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtz(float);
07760 uchar const_func __attribute__((overloadable)) convert_uchar_rtp(float);
07761 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtp(float);
07762 uchar const_func __attribute__((overloadable)) convert_uchar_rtn(float);
07763 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtn(float);
07764 uchar const_func __attribute__((overloadable)) convert_uchar(float);
07765 uchar const_func __attribute__((overloadable)) convert_uchar_sat(float);
07766 uchar const_func __attribute__((overloadable)) convert_uchar_rte(double);
07767 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rte(double);
07768 uchar const_func __attribute__((overloadable)) convert_uchar_rtz(double);
07769 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtz(double);
07770 uchar const_func __attribute__((overloadable)) convert_uchar_rtp(double);
07771 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtp(double);
07772 uchar const_func __attribute__((overloadable)) convert_uchar_rtn(double);
07773 uchar const_func __attribute__((overloadable)) convert_uchar_sat_rtn(double);
07774 uchar const_func __attribute__((overloadable)) convert_uchar(double);
07775 uchar const_func __attribute__((overloadable)) convert_uchar_sat(double);
07776
07777 short const_func __attribute__((overloadable)) convert_short_rte(char);
07778 short const_func __attribute__((overloadable)) convert_short_sat_rte(char);
07779 short const_func __attribute__((overloadable)) convert_short_rtz(char);
07780 short const_func __attribute__((overloadable)) convert_short_sat_rtz(char);
07781 short const_func __attribute__((overloadable)) convert_short_rtp(char);
07782 short const_func __attribute__((overloadable)) convert_short_sat_rtp(char);
07783 short const_func __attribute__((overloadable)) convert_short_rtn(char);
07784 short const_func __attribute__((overloadable)) convert_short_sat_rtn(char);
07785 short const_func __attribute__((overloadable)) convert_short(char);
07786 short const_func __attribute__((overloadable)) convert_short_sat(char);
07787 short const_func __attribute__((overloadable)) convert_short_rte(uchar);
07788 short const_func __attribute__((overloadable)) convert_short_sat_rte(uchar);
07789 short const_func __attribute__((overloadable)) convert_short_rtz(uchar);
07790 short const_func __attribute__((overloadable)) convert_short_sat_rtz(uchar);
07791 short const_func __attribute__((overloadable)) convert_short_rtp(uchar);
07792 short const_func __attribute__((overloadable)) convert_short_sat_rtp(uchar);
07793 short const_func __attribute__((overloadable)) convert_short_rtn(uchar);
07794 short const_func __attribute__((overloadable)) convert_short_sat_rtn(uchar);
07795 short const_func __attribute__((overloadable)) convert_short(uchar);
07796 short const_func __attribute__((overloadable)) convert_short_sat(uchar);
07797 short const_func __attribute__((overloadable)) convert_short_rte(short);
07798 short const_func __attribute__((overloadable)) convert_short_sat_rte(short);
07799 short const_func __attribute__((overloadable)) convert_short_rtz(short);
07800 short const_func __attribute__((overloadable)) convert_short_sat_rtz(short);
07801 short const_func __attribute__((overloadable)) convert_short_rtp(short);
07802 short const_func __attribute__((overloadable)) convert_short_sat_rtp(short);
07803 short const_func __attribute__((overloadable)) convert_short_rtn(short);
07804 short const_func __attribute__((overloadable)) convert_short_sat_rtn(short);
07805 short const_func __attribute__((overloadable)) convert_short(short);
07806 short const_func __attribute__((overloadable)) convert_short_sat(short);
07807 short const_func __attribute__((overloadable)) convert_short_rte(ushort);
07808 short const_func __attribute__((overloadable)) convert_short_sat_rte(ushort);
07809 short const_func __attribute__((overloadable)) convert_short_rtz(ushort);
07810 short const_func __attribute__((overloadable)) convert_short_sat_rtz(ushort);
07811 short const_func __attribute__((overloadable)) convert_short_rtp(ushort);
07812 short const_func __attribute__((overloadable)) convert_short_sat_rtp(ushort);
07813 short const_func __attribute__((overloadable)) convert_short_rtn(ushort);
07814 short const_func __attribute__((overloadable)) convert_short_sat_rtn(ushort);
07815 short const_func __attribute__((overloadable)) convert_short(ushort);
07816 short const_func __attribute__((overloadable)) convert_short_sat(ushort);
07817 short const_func __attribute__((overloadable)) convert_short_rte(int);
07818 short const_func __attribute__((overloadable)) convert_short_sat_rte(int);
07819 short const_func __attribute__((overloadable)) convert_short_rtz(int);
07820 short const_func __attribute__((overloadable)) convert_short_sat_rtz(int);
07821 short const_func __attribute__((overloadable)) convert_short_rtp(int);
07822 short const_func __attribute__((overloadable)) convert_short_sat_rtp(int);
07823 short const_func __attribute__((overloadable)) convert_short_rtn(int);
07824 short const_func __attribute__((overloadable)) convert_short_sat_rtn(int);
07825 short const_func __attribute__((overloadable)) convert_short(int);
07826 short const_func __attribute__((overloadable)) convert_short_sat(int);
07827 short const_func __attribute__((overloadable)) convert_short_rte(uint);
07828 short const_func __attribute__((overloadable)) convert_short_sat_rte(uint);
07829 short const_func __attribute__((overloadable)) convert_short_rtz(uint);
07830 short const_func __attribute__((overloadable)) convert_short_sat_rtz(uint);
07831 short const_func __attribute__((overloadable)) convert_short_rtp(uint);
07832 short const_func __attribute__((overloadable)) convert_short_sat_rtp(uint);
07833 short const_func __attribute__((overloadable)) convert_short_rtn(uint);
07834 short const_func __attribute__((overloadable)) convert_short_sat_rtn(uint);
07835 short const_func __attribute__((overloadable)) convert_short(uint);
07836 short const_func __attribute__((overloadable)) convert_short_sat(uint);
07837 short const_func __attribute__((overloadable)) convert_short_rte(long);
07838 short const_func __attribute__((overloadable)) convert_short_sat_rte(long);
07839 short const_func __attribute__((overloadable)) convert_short_rtz(long);
07840 short const_func __attribute__((overloadable)) convert_short_sat_rtz(long);
07841 short const_func __attribute__((overloadable)) convert_short_rtp(long);
07842 short const_func __attribute__((overloadable)) convert_short_sat_rtp(long);
```

```
07843 short const_func __attribute__((overloadable)) convert_short_rtn(long);
07844 short const_func __attribute__((overloadable)) convert_short_sat_rtn(long);
07845 short const_func __attribute__((overloadable)) convert_short(long);
07846 short const_func __attribute__((overloadable)) convert_short_sat(long);
07847 short const_func __attribute__((overloadable)) convert_short_rte(ulong);
07848 short const_func __attribute__((overloadable)) convert_short_sat_rte(ulong);
07849 short const_func __attribute__((overloadable)) convert_short_rtz(ulong);
07850 short const_func __attribute__((overloadable)) convert_short_sat_rtz(ulong);
07851 short const_func __attribute__((overloadable)) convert_short_rtp(ulong);
07852 short const_func __attribute__((overloadable)) convert_short_sat_rtp(ulong);
07853 short const_func __attribute__((overloadable)) convert_short_rtn(ulong);
07854 short const_func __attribute__((overloadable)) convert_short_sat_rtn(ulong);
07855 short const_func __attribute__((overloadable)) convert_short(ulong);
07856 short const_func __attribute__((overloadable)) convert_short_sat(ulong);
07857 short const_func __attribute__((overloadable)) convert_short_rte(float);
07858 short const_func __attribute__((overloadable)) convert_short_sat_rte(float);
07859 short const_func __attribute__((overloadable)) convert_short_rtz(float);
07860 short const_func __attribute__((overloadable)) convert_short_sat_rtz(float);
07861 short const_func __attribute__((overloadable)) convert_short_rtp(float);
07862 short const_func __attribute__((overloadable)) convert_short_sat_rtp(float);
07863 short const_func __attribute__((overloadable)) convert_short_rtn(float);
07864 short const_func __attribute__((overloadable)) convert_short_sat_rtn(float);
07865 short const_func __attribute__((overloadable)) convert_short(float);
07866 short const_func __attribute__((overloadable)) convert_short_sat(float);
07867 short const_func __attribute__((overloadable)) convert_short_rte(double);
07868 short const_func __attribute__((overloadable)) convert_short_sat_rte(double);
07869 short const_func __attribute__((overloadable)) convert_short_rtz(double);
07870 short const_func __attribute__((overloadable)) convert_short_sat_rtz(double);
07871 short const_func __attribute__((overloadable)) convert_short_rtp(double);
07872 short const_func __attribute__((overloadable)) convert_short_sat_rtp(double);
07873 short const_func __attribute__((overloadable)) convert_short_rtn(double);
07874 short const_func __attribute__((overloadable)) convert_short_sat_rtn(double);
07875 short const_func __attribute__((overloadable)) convert_short(double);
07876 short const_func __attribute__((overloadable)) convert_short_sat(double);
07877
07878 ushort const_func __attribute__((overloadable)) convert_ushort_rte(char);
07879 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rte(char);
07880 ushort const_func __attribute__((overloadable)) convert_ushort_rtz(char);
07881 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtz(char);
07882 ushort const_func __attribute__((overloadable)) convert_ushort_rtp(char);
07883 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtp(char);
07884 ushort const_func __attribute__((overloadable)) convert_ushort_rtn(char);
07885 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtn(char);
07886 ushort const_func __attribute__((overloadable)) convert_ushort(char);
07887 ushort const_func __attribute__((overloadable)) convert_ushort_sat(char);
07888 ushort const_func __attribute__((overloadable)) convert_ushort_rte(uchar);
07889 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rte(uchar);
07890 ushort const_func __attribute__((overloadable)) convert_ushort_rtz(uchar);
07891 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtz(uchar);
07892 ushort const_func __attribute__((overloadable)) convert_ushort_rtp(uchar);
07893 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtp(uchar);
07894 ushort const_func __attribute__((overloadable)) convert_ushort_rtn(uchar);
07895 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtn(uchar);
07896 ushort const_func __attribute__((overloadable)) convert_ushort(uchar);
07897 ushort const_func __attribute__((overloadable)) convert_ushort_sat(uchar);
07898 ushort const_func __attribute__((overloadable)) convert_ushort_rte(short);
07899 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rte(short);
07900 ushort const_func __attribute__((overloadable)) convert_ushort_rtz(short);
07901 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtz(short);
07902 ushort const_func __attribute__((overloadable)) convert_ushort_rtp(short);
07903 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtp(short);
07904 ushort const_func __attribute__((overloadable)) convert_ushort_rtn(short);
07905 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtn(short);
07906 ushort const_func __attribute__((overloadable)) convert_ushort(short);
07907 ushort const_func __attribute__((overloadable)) convert_ushort_sat(short);
07908 ushort const_func __attribute__((overloadable)) convert_ushort_rte(ushort);
07909 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rte(ushort);
07910 ushort const_func __attribute__((overloadable)) convert_ushort_rtz(ushort);
07911 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtz(ushort);
07912 ushort const_func __attribute__((overloadable)) convert_ushort_rtp(ushort);
07913 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtp(ushort);
07914 ushort const_func __attribute__((overloadable)) convert_ushort_rtn(ushort);
07915 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtn(ushort);
07916 ushort const_func __attribute__((overloadable)) convert_ushort(ushort);
07917 ushort const_func __attribute__((overloadable)) convert_ushort_sat(ushort);
07918 ushort const_func __attribute__((overloadable)) convert_ushort_rte(int);
07919 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rte(int);
07920 ushort const_func __attribute__((overloadable)) convert_ushort_rtz(int);
07921 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtz(int);
07922 ushort const_func __attribute__((overloadable)) convert_ushort_rtp(int);
07923 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtp(int);
07924 ushort const_func __attribute__((overloadable)) convert_ushort_rtn(int);
07925 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtn(int);
07926 ushort const_func __attribute__((overloadable)) convert_ushort(int);
07927 ushort const_func __attribute__((overloadable)) convert_ushort_sat(int);
07928 ushort const_func __attribute__((overloadable)) convert_ushort_rte(uint);
07929 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rte(uint);
```

```
07930 ushort const_func __attribute__((overloadable)) convert_ushort_rtz(uint);
07931 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtz(uint);
07932 ushort const_func __attribute__((overloadable)) convert_ushort_rtp(uint);
07933 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtp(uint);
07934 ushort const_func __attribute__((overloadable)) convert_ushort_rtn(uint);
07935 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtn(uint);
07936 ushort const_func __attribute__((overloadable)) convert_ushort(uint);
07937 ushort const_func __attribute__((overloadable)) convert_ushort_sat(uint);
07938 ushort const_func __attribute__((overloadable)) convert_ushort_rte(long);
07939 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rte(long);
07940 ushort const_func __attribute__((overloadable)) convert_ushort_rtz(long);
07941 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtz(long);
07942 ushort const_func __attribute__((overloadable)) convert_ushort_rtp(long);
07943 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtp(long);
07944 ushort const_func __attribute__((overloadable)) convert_ushort_rtn(long);
07945 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtn(long);
07946 ushort const_func __attribute__((overloadable)) convert_ushort(long);
07947 ushort const_func __attribute__((overloadable)) convert_ushort_sat(long);
07948 ushort const_func __attribute__((overloadable)) convert_ushort_rte(ulong);
07949 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rte(ulong);
07950 ushort const_func __attribute__((overloadable)) convert_ushort_rtz(ulong);
07951 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtz(ulong);
07952 ushort const_func __attribute__((overloadable)) convert_ushort_rtp(ulong);
07953 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtp(ulong);
07954 ushort const_func __attribute__((overloadable)) convert_ushort_rtn(ulong);
07955 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtn(ulong);
07956 ushort const_func __attribute__((overloadable)) convert_ushort(ulong);
07957 ushort const_func __attribute__((overloadable)) convert_ushort_sat(ulong);
07958 ushort const_func __attribute__((overloadable)) convert_ushort_rte(float);
07959 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rte(float);
07960 ushort const_func __attribute__((overloadable)) convert_ushort_rtz(float);
07961 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtz(float);
07962 ushort const_func __attribute__((overloadable)) convert_ushort_rtp(float);
07963 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtp(float);
07964 ushort const_func __attribute__((overloadable)) convert_ushort_rtn(float);
07965 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtn(float);
07966 ushort const_func __attribute__((overloadable)) convert_ushort(float);
07967 ushort const_func __attribute__((overloadable)) convert_ushort_sat(float);
07968 ushort const_func __attribute__((overloadable)) convert_ushort_rte(double);
07969 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rte(double);
07970 ushort const_func __attribute__((overloadable)) convert_ushort_rtz(double);
07971 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtz(double);
07972 ushort const_func __attribute__((overloadable)) convert_ushort_rtp(double);
07973 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtp(double);
07974 ushort const_func __attribute__((overloadable)) convert_ushort_rtn(double);
07975 ushort const_func __attribute__((overloadable)) convert_ushort_sat_rtn(double);
07976 ushort const_func __attribute__((overloadable)) convert_ushort(double);
07977 ushort const_func __attribute__((overloadable)) convert_ushort_sat(double);
07978
07979 int const_func __attribute__((overloadable)) convert_int_rte(char);
07980 int const_func __attribute__((overloadable)) convert_int_sat_rte(char);
07981 int const_func __attribute__((overloadable)) convert_int_rtz(char);
07982 int const_func __attribute__((overloadable)) convert_int_sat_rtz(char);
07983 int const_func __attribute__((overloadable)) convert_int_rtp(char);
07984 int const_func __attribute__((overloadable)) convert_int_sat_rtp(char);
07985 int const_func __attribute__((overloadable)) convert_int_rtn(char);
07986 int const_func __attribute__((overloadable)) convert_int_sat_rtn(char);
07987 int const_func __attribute__((overloadable)) convert_int(char);
07988 int const_func __attribute__((overloadable)) convert_int_sat(char);
07989 int const_func __attribute__((overloadable)) convert_int_rte(uchar);
07990 int const_func __attribute__((overloadable)) convert_int_sat_rte(uchar);
07991 int const_func __attribute__((overloadable)) convert_int_rtz(uchar);
07992 int const_func __attribute__((overloadable)) convert_int_sat_rtz(uchar);
07993 int const_func __attribute__((overloadable)) convert_int_rtp(uchar);
07994 int const_func __attribute__((overloadable)) convert_int_sat_rtp(uchar);
07995 int const_func __attribute__((overloadable)) convert_int_rtn(uchar);
07996 int const_func __attribute__((overloadable)) convert_int_sat_rtn(uchar);
07997 int const_func __attribute__((overloadable)) convert_int(uchar);
07998 int const_func __attribute__((overloadable)) convert_int_sat(uchar);
07999 int const_func __attribute__((overloadable)) convert_int_rte(short);
08000 int const_func __attribute__((overloadable)) convert_int_sat_rte(short);
08001 int const_func __attribute__((overloadable)) convert_int_rtz(short);
08002 int const_func __attribute__((overloadable)) convert_int_sat_rtz(short);
08003 int const_func __attribute__((overloadable)) convert_int_rtp(short);
08004 int const_func __attribute__((overloadable)) convert_int_sat_rtp(short);
08005 int const_func __attribute__((overloadable)) convert_int_rtn(short);
08006 int const_func __attribute__((overloadable)) convert_int_sat_rtn(short);
08007 int const_func __attribute__((overloadable)) convert_int(short);
08008 int const_func __attribute__((overloadable)) convert_int_sat(short);
08009 int const_func __attribute__((overloadable)) convert_int_rte(ushort);
08010 int const_func __attribute__((overloadable)) convert_int_sat_rte(ushort);
08011 int const_func __attribute__((overloadable)) convert_int_rtz(ushort);
08012 int const_func __attribute__((overloadable)) convert_int_sat_rtz(ushort);
08013 int const_func __attribute__((overloadable)) convert_int_rtp(ushort);
08014 int const_func __attribute__((overloadable)) convert_int_sat_rtp(ushort);
08015 int const_func __attribute__((overloadable)) convert_int_rtn(ushort);
08016 int const_func __attribute__((overloadable)) convert_int_sat_rtn(ushort);
```

```
08017 int const_func __attribute__((overloadable)) convert_int(ushort);
08018 int const_func __attribute__((overloadable)) convert_int_sat(ushort);
08019 int const_func __attribute__((overloadable)) convert_int_rte(int);
08020 int const_func __attribute__((overloadable)) convert_int_sat_rte(int);
08021 int const_func __attribute__((overloadable)) convert_int_rtz(int);
08022 int const_func __attribute__((overloadable)) convert_int_sat_rtz(int);
08023 int const_func __attribute__((overloadable)) convert_int_rtp(int);
08024 int const_func __attribute__((overloadable)) convert_int_sat_rtp(int);
08025 int const_func __attribute__((overloadable)) convert_int_rtn(int);
08026 int const_func __attribute__((overloadable)) convert_int_sat_rtn(int);
08027 int const_func __attribute__((overloadable)) convert_int(int);
08028 int const_func __attribute__((overloadable)) convert_int_sat(int);
08029 int const_func __attribute__((overloadable)) convert_int_rte(uint);
08030 int const_func __attribute__((overloadable)) convert_int_sat_rte(uint);
08031 int const_func __attribute__((overloadable)) convert_int_rtz(uint);
08032 int const_func __attribute__((overloadable)) convert_int_sat_rtz(uint);
08033 int const_func __attribute__((overloadable)) convert_int_rtp(uint);
08034 int const_func __attribute__((overloadable)) convert_int_sat_rtp(uint);
08035 int const_func __attribute__((overloadable)) convert_int_rtn(uint);
08036 int const_func __attribute__((overloadable)) convert_int_sat_rtn(uint);
08037 int const_func __attribute__((overloadable)) convert_int(uint);
08038 int const_func __attribute__((overloadable)) convert_int_sat(uint);
08039 int const_func __attribute__((overloadable)) convert_int_rte(long);
08040 int const_func __attribute__((overloadable)) convert_int_sat_rte(long);
08041 int const_func __attribute__((overloadable)) convert_int_rtz(long);
08042 int const_func __attribute__((overloadable)) convert_int_sat_rtz(long);
08043 int const_func __attribute__((overloadable)) convert_int_rtp(long);
08044 int const_func __attribute__((overloadable)) convert_int_sat_rtp(long);
08045 int const_func __attribute__((overloadable)) convert_int_rtn(long);
08046 int const_func __attribute__((overloadable)) convert_int_sat_rtn(long);
08047 int const_func __attribute__((overloadable)) convert_int(long);
08048 int const_func __attribute__((overloadable)) convert_int_sat(long);
08049 int const_func __attribute__((overloadable)) convert_int_rte(ulong);
08050 int const_func __attribute__((overloadable)) convert_int_sat_rte(ulong);
08051 int const_func __attribute__((overloadable)) convert_int_rtz(ulong);
08052 int const_func __attribute__((overloadable)) convert_int_sat_rtz(ulong);
08053 int const_func __attribute__((overloadable)) convert_int_rtp(ulong);
08054 int const_func __attribute__((overloadable)) convert_int_sat_rtp(ulong);
08055 int const_func __attribute__((overloadable)) convert_int_rtn(ulong);
08056 int const_func __attribute__((overloadable)) convert_int_sat_rtn(ulong);
08057 int const_func __attribute__((overloadable)) convert_int(ulong);
08058 int const_func __attribute__((overloadable)) convert_int_sat(ulong);
08059 int const_func __attribute__((overloadable)) convert_int_rte(float);
08060 int const_func __attribute__((overloadable)) convert_int_sat_rte(float);
08061 int const_func __attribute__((overloadable)) convert_int_rtz(float);
08062 int const_func __attribute__((overloadable)) convert_int_sat_rtz(float);
08063 int const_func __attribute__((overloadable)) convert_int_rtp(float);
08064 int const_func __attribute__((overloadable)) convert_int_sat_rtp(float);
08065 int const_func __attribute__((overloadable)) convert_int_rtn(float);
08066 int const_func __attribute__((overloadable)) convert_int_sat_rtn(float);
08067 int const_func __attribute__((overloadable)) convert_int(float);
08068 int const_func __attribute__((overloadable)) convert_int_sat(float);
08069 int const_func __attribute__((overloadable)) convert_int_rte(double);
08070 int const_func __attribute__((overloadable)) convert_int_sat_rte(double);
08071 int const_func __attribute__((overloadable)) convert_int_rtz(double);
08072 int const_func __attribute__((overloadable)) convert_int_sat_rtz(double);
08073 int const_func __attribute__((overloadable)) convert_int_rtp(double);
08074 int const_func __attribute__((overloadable)) convert_int_sat_rtp(double);
08075 int const_func __attribute__((overloadable)) convert_int_rtn(double);
08076 int const_func __attribute__((overloadable)) convert_int_sat_rtn(double);
08077 int const_func __attribute__((overloadable)) convert_int(double);
08078 int const_func __attribute__((overloadable)) convert_int_sat(double);
08079
08080 uint const_func __attribute__((overloadable)) convert_uint_rte(char);
08081 uint const_func __attribute__((overloadable)) convert_uint_sat_rte(char);
08082 uint const_func __attribute__((overloadable)) convert_uint_rtz(char);
08083 uint const_func __attribute__((overloadable)) convert_uint_sat_rtz(char);
08084 uint const_func __attribute__((overloadable)) convert_uint_rtp(char);
08085 uint const_func __attribute__((overloadable)) convert_uint_sat_rtp(char);
08086 uint const_func __attribute__((overloadable)) convert_uint_rtn(char);
08087 uint const_func __attribute__((overloadable)) convert_uint_sat_rtn(char);
08088 uint const_func __attribute__((overloadable)) convert_uint(char);
08089 uint const_func __attribute__((overloadable)) convert_uint_sat(char);
08090 uint const_func __attribute__((overloadable)) convert_uint_rte(uchar);
08091 uint const_func __attribute__((overloadable)) convert_uint_sat_rte(uchar);
08092 uint const_func __attribute__((overloadable)) convert_uint_rtz(uchar);
08093 uint const_func __attribute__((overloadable)) convert_uint_sat_rtz(uchar);
08094 uint const_func __attribute__((overloadable)) convert_uint_rtp(uchar);
08095 uint const_func __attribute__((overloadable)) convert_uint_sat_rtp(uchar);
08096 uint const_func __attribute__((overloadable)) convert_uint_rtn(uchar);
08097 uint const_func __attribute__((overloadable)) convert_uint_sat_rtn(uchar);
08098 uint const_func __attribute__((overloadable)) convert_uint(uchar);
08099 uint const_func __attribute__((overloadable)) convert_uint_sat(uchar);
08100 uint const_func __attribute__((overloadable)) convert_uint_rte(short);
08101 uint const_func __attribute__((overloadable)) convert_uint_sat_rte(short);
08102 uint const_func __attribute__((overloadable)) convert_uint_rtz(short);
08103 uint const_func __attribute__((overloadable)) convert_uint_sat_rtz(short);
```

```
08104 uint const_func __attribute__((overloadable)) convert_uint_rtp(short);
08105 uint const_func __attribute__((overloadable)) convert_uint_sat_rtp(short);
08106 uint const_func __attribute__((overloadable)) convert_uint_rtn(short);
08107 uint const_func __attribute__((overloadable)) convert_uint_sat_rtn(short);
08108 uint const_func __attribute__((overloadable)) convert_uint(short);
08109 uint const_func __attribute__((overloadable)) convert_uint_sat(short);
08110 uint const_func __attribute__((overloadable)) convert_uint_rte(ushort);
08111 uint const_func __attribute__((overloadable)) convert_uint_sat_rte(ushort);
08112 uint const_func __attribute__((overloadable)) convert_uint_rtz(ushort);
08113 uint const_func __attribute__((overloadable)) convert_uint_sat_rtz(ushort);
08114 uint const_func __attribute__((overloadable)) convert_uint_rtp(ushort);
08115 uint const_func __attribute__((overloadable)) convert_uint_sat_rtp(ushort);
08116 uint const_func __attribute__((overloadable)) convert_uint_rtn(ushort);
08117 uint const_func __attribute__((overloadable)) convert_uint_sat_rtn(ushort);
08118 uint const_func __attribute__((overloadable)) convert_uint(ushort);
08119 uint const_func __attribute__((overloadable)) convert_uint_sat(ushort);
08120 uint const_func __attribute__((overloadable)) convert_uint_rte(int);
08121 uint const_func __attribute__((overloadable)) convert_uint_sat_rte(int);
08122 uint const_func __attribute__((overloadable)) convert_uint_rtz(int);
08123 uint const_func __attribute__((overloadable)) convert_uint_sat_rtz(int);
08124 uint const_func __attribute__((overloadable)) convert_uint_rtp(int);
08125 uint const_func __attribute__((overloadable)) convert_uint_sat_rtp(int);
08126 uint const_func __attribute__((overloadable)) convert_uint_rtn(int);
08127 uint const_func __attribute__((overloadable)) convert_uint_sat_rtn(int);
08128 uint const_func __attribute__((overloadable)) convert_uint(int);
08129 uint const_func __attribute__((overloadable)) convert_uint_sat(int);
08130 uint const_func __attribute__((overloadable)) convert_uint_rte(uint);
08131 uint const_func __attribute__((overloadable)) convert_uint_sat_rte(uint);
08132 uint const_func __attribute__((overloadable)) convert_uint_rtz(uint);
08133 uint const_func __attribute__((overloadable)) convert_uint_sat_rtz(uint);
08134 uint const_func __attribute__((overloadable)) convert_uint_rtp(uint);
08135 uint const_func __attribute__((overloadable)) convert_uint_sat_rtp(uint);
08136 uint const_func __attribute__((overloadable)) convert_uint_rtn(uint);
08137 uint const_func __attribute__((overloadable)) convert_uint_sat_rtn(uint);
08138 uint const_func __attribute__((overloadable)) convert_uint(uint);
08139 uint const_func __attribute__((overloadable)) convert_uint_sat(uint);
08140 uint const_func __attribute__((overloadable)) convert_uint_rte(long);
08141 uint const_func __attribute__((overloadable)) convert_uint_sat_rte(long);
08142 uint const_func __attribute__((overloadable)) convert_uint_rtz(long);
08143 uint const_func __attribute__((overloadable)) convert_uint_sat_rtz(long);
08144 uint const_func __attribute__((overloadable)) convert_uint_rtp(long);
08145 uint const_func __attribute__((overloadable)) convert_uint_sat_rtp(long);
08146 uint const_func __attribute__((overloadable)) convert_uint_rtn(long);
08147 uint const_func __attribute__((overloadable)) convert_uint_sat_rtn(long);
08148 uint const_func __attribute__((overloadable)) convert_uint(long);
08149 uint const_func __attribute__((overloadable)) convert_uint_sat(long);
08150 uint const_func __attribute__((overloadable)) convert_uint_rte(ulong);
08151 uint const_func __attribute__((overloadable)) convert_uint_sat_rte(ulong);
08152 uint const_func __attribute__((overloadable)) convert_uint_rtz(ulong);
08153 uint const_func __attribute__((overloadable)) convert_uint_sat_rtz(ulong);
08154 uint const_func __attribute__((overloadable)) convert_uint_rtp(ulong);
08155 uint const_func __attribute__((overloadable)) convert_uint_sat_rtp(ulong);
08156 uint const_func __attribute__((overloadable)) convert_uint_rtn(ulong);
08157 uint const_func __attribute__((overloadable)) convert_uint_sat_rtn(ulong);
08158 uint const_func __attribute__((overloadable)) convert_uint(ulong);
08159 uint const_func __attribute__((overloadable)) convert_uint_sat(ulong);
08160 uint const_func __attribute__((overloadable)) convert_uint_rte(float);
08161 uint const_func __attribute__((overloadable)) convert_uint_sat_rte(float);
08162 uint const_func __attribute__((overloadable)) convert_uint_rtz(float);
08163 uint const_func __attribute__((overloadable)) convert_uint_sat_rtz(float);
08164 uint const_func __attribute__((overloadable)) convert_uint_rtp(float);
08165 uint const_func __attribute__((overloadable)) convert_uint_sat_rtp(float);
08166 uint const_func __attribute__((overloadable)) convert_uint_rtn(float);
08167 uint const_func __attribute__((overloadable)) convert_uint_sat_rtn(float);
08168 uint const_func __attribute__((overloadable)) convert_uint(float);
08169 uint const_func __attribute__((overloadable)) convert_uint_sat(float);
08170 uint const_func __attribute__((overloadable)) convert_uint_rte(double);
08171 uint const_func __attribute__((overloadable)) convert_uint_sat_rte(double);
08172 uint const_func __attribute__((overloadable)) convert_uint_rtz(double);
08173 uint const_func __attribute__((overloadable)) convert_uint_sat_rtz(double);
08174 uint const_func __attribute__((overloadable)) convert_uint_rtp(double);
08175 uint const_func __attribute__((overloadable)) convert_uint_sat_rtp(double);
08176 uint const_func __attribute__((overloadable)) convert_uint_rtn(double);
08177 uint const_func __attribute__((overloadable)) convert_uint_sat_rtn(double);
08178 uint const_func __attribute__((overloadable)) convert_uint(double);
08179 uint const_func __attribute__((overloadable)) convert_uint_sat(double);
08180 long const_func __attribute__((overloadable)) convert_long_rte(char);
08181 long const_func __attribute__((overloadable)) convert_long_sat_rte(char);
08182 long const_func __attribute__((overloadable)) convert_long_rtz(char);
08183 long const_func __attribute__((overloadable)) convert_long_sat_rtz(char);
08184 long const_func __attribute__((overloadable)) convert_long_rtp(char);
08185 long const_func __attribute__((overloadable)) convert_long_sat_rtp(char);
08186 long const_func __attribute__((overloadable)) convert_long_rtn(char);
08187 long const_func __attribute__((overloadable)) convert_long_sat_rtn(char);
08188 long const_func __attribute__((overloadable)) convert_long(char);
08189 long const_func __attribute__((overloadable)) convert_long_sat(char);
08190 long const_func __attribute__((overloadable)) convert_long_rte(uchar);
```

```
08191 long const_func __attribute__((overloadable)) convert_long_sat_rte(uchar);
08192 long const_func __attribute__((overloadable)) convert_long_rtz(uchar);
08193 long const_func __attribute__((overloadable)) convert_long_sat_rtz(uchar);
08194 long const_func __attribute__((overloadable)) convert_long_rtp(uchar);
08195 long const_func __attribute__((overloadable)) convert_long_sat_rtp(uchar);
08196 long const_func __attribute__((overloadable)) convert_long_rtn(uchar);
08197 long const_func __attribute__((overloadable)) convert_long_sat_rtn(uchar);
08198 long const_func __attribute__((overloadable)) convert_long(uchar);
08199 long const_func __attribute__((overloadable)) convert_long_sat(uchar);
08200 long const_func __attribute__((overloadable)) convert_long_rte(short);
08201 long const_func __attribute__((overloadable)) convert_long_sat_rte(short);
08202 long const_func __attribute__((overloadable)) convert_long_rtz(short);
08203 long const_func __attribute__((overloadable)) convert_long_sat_rtz(short);
08204 long const_func __attribute__((overloadable)) convert_long_rtp(short);
08205 long const_func __attribute__((overloadable)) convert_long_sat_rtp(short);
08206 long const_func __attribute__((overloadable)) convert_long_rtn(short);
08207 long const_func __attribute__((overloadable)) convert_long_sat_rtn(short);
08208 long const_func __attribute__((overloadable)) convert_long(short);
08209 long const_func __attribute__((overloadable)) convert_long_sat(short);
08210 long const_func __attribute__((overloadable)) convert_long_rte(ushort);
08211 long const_func __attribute__((overloadable)) convert_long_sat_rte(ushort);
08212 long const_func __attribute__((overloadable)) convert_long_rtz(ushort);
08213 long const_func __attribute__((overloadable)) convert_long_sat_rtz(ushort);
08214 long const_func __attribute__((overloadable)) convert_long_rtp(ushort);
08215 long const_func __attribute__((overloadable)) convert_long_sat_rtp(ushort);
08216 long const_func __attribute__((overloadable)) convert_long_rtn(ushort);
08217 long const_func __attribute__((overloadable)) convert_long_sat_rtn(ushort);
08218 long const_func __attribute__((overloadable)) convert_long(ushort);
08219 long const_func __attribute__((overloadable)) convert_long_sat(ushort);
08220 long const_func __attribute__((overloadable)) convert_long_rte(int);
08221 long const_func __attribute__((overloadable)) convert_long_sat_rte(int);
08222 long const_func __attribute__((overloadable)) convert_long_rtz(int);
08223 long const_func __attribute__((overloadable)) convert_long_sat_rtz(int);
08224 long const_func __attribute__((overloadable)) convert_long_rtp(int);
08225 long const_func __attribute__((overloadable)) convert_long_sat_rtp(int);
08226 long const_func __attribute__((overloadable)) convert_long_rtn(int);
08227 long const_func __attribute__((overloadable)) convert_long_sat_rtn(int);
08228 long const_func __attribute__((overloadable)) convert_long(int);
08229 long const_func __attribute__((overloadable)) convert_long_sat(int);
08230 long const_func __attribute__((overloadable)) convert_long_rte(uint);
08231 long const_func __attribute__((overloadable)) convert_long_sat_rte(uint);
08232 long const_func __attribute__((overloadable)) convert_long_rtz(uint);
08233 long const_func __attribute__((overloadable)) convert_long_sat_rtz(uint);
08234 long const_func __attribute__((overloadable)) convert_long_rtp(uint);
08235 long const_func __attribute__((overloadable)) convert_long_sat_rtp(uint);
08236 long const_func __attribute__((overloadable)) convert_long_rtn(uint);
08237 long const_func __attribute__((overloadable)) convert_long_sat_rtn(uint);
08238 long const_func __attribute__((overloadable)) convert_long(uint);
08239 long const_func __attribute__((overloadable)) convert_long_sat(uint);
08240 long const_func __attribute__((overloadable)) convert_long_rte(long);
08241 long const_func __attribute__((overloadable)) convert_long_sat_rte(long);
08242 long const_func __attribute__((overloadable)) convert_long_rtz(long);
08243 long const_func __attribute__((overloadable)) convert_long_sat_rtz(long);
08244 long const_func __attribute__((overloadable)) convert_long_rtp(long);
08245 long const_func __attribute__((overloadable)) convert_long_sat_rtp(long);
08246 long const_func __attribute__((overloadable)) convert_long_rtn(long);
08247 long const_func __attribute__((overloadable)) convert_long_sat_rtn(long);
08248 long const_func __attribute__((overloadable)) convert_long(long);
08249 long const_func __attribute__((overloadable)) convert_long_sat(long);
08250 long const_func __attribute__((overloadable)) convert_long_rte(ulong);
08251 long const_func __attribute__((overloadable)) convert_long_sat_rte(ulong);
08252 long const_func __attribute__((overloadable)) convert_long_rtz(ulong);
08253 long const_func __attribute__((overloadable)) convert_long_sat_rtz(ulong);
08254 long const_func __attribute__((overloadable)) convert_long_rtp(ulong);
08255 long const_func __attribute__((overloadable)) convert_long_sat_rtp(ulong);
08256 long const_func __attribute__((overloadable)) convert_long_rtn(ulong);
08257 long const_func __attribute__((overloadable)) convert_long_sat_rtn(ulong);
08258 long const_func __attribute__((overloadable)) convert_long(ulong);
08259 long const_func __attribute__((overloadable)) convert_long_sat(ulong);
08260 long const_func __attribute__((overloadable)) convert_long_rte(float);
08261 long const_func __attribute__((overloadable)) convert_long_sat_rte(float);
08262 long const_func __attribute__((overloadable)) convert_long_rtz(float);
08263 long const_func __attribute__((overloadable)) convert_long_sat_rtz(float);
08264 long const_func __attribute__((overloadable)) convert_long_rtp(float);
08265 long const_func __attribute__((overloadable)) convert_long_sat_rtp(float);
08266 long const_func __attribute__((overloadable)) convert_long_rtn(float);
08267 long const_func __attribute__((overloadable)) convert_long_sat_rtn(float);
08268 long const_func __attribute__((overloadable)) convert_long(float);
08269 long const_func __attribute__((overloadable)) convert_long_sat(float);
08270 long const_func __attribute__((overloadable)) convert_long_rte(double);
08271 long const_func __attribute__((overloadable)) convert_long_sat_rte(double);
08272 long const_func __attribute__((overloadable)) convert_long_rtz(double);
08273 long const_func __attribute__((overloadable)) convert_long_sat_rtz(double);
08274 long const_func __attribute__((overloadable)) convert_long_rtp(double);
08275 long const_func __attribute__((overloadable)) convert_long_sat_rtp(double);
08276 long const_func __attribute__((overloadable)) convert_long_rtn(double);
08277 long const_func __attribute__((overloadable)) convert_long_sat_rtn(double);
```

```
08278 long const_func __attribute__((overloadable)) convert_long(double);
08279 long const_func __attribute__((overloadable)) convert_long_sat(double);
08280
08281 ulong const_func __attribute__((overloadable)) convert_ulong_rte(char);
08282 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rte(char);
08283 ulong const_func __attribute__((overloadable)) convert_ulong_rtz(char);
08284 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtz(char);
08285 ulong const_func __attribute__((overloadable)) convert_ulong_rtp(char);
08286 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtp(char);
08287 ulong const_func __attribute__((overloadable)) convert_ulong_rtn(char);
08288 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtn(char);
08289 ulong const_func __attribute__((overloadable)) convert_ulong(char);
08290 ulong const_func __attribute__((overloadable)) convert_ulong_sat(char);
08291 ulong const_func __attribute__((overloadable)) convert_ulong_rte(uchar);
08292 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rte(uchar);
08293 ulong const_func __attribute__((overloadable)) convert_ulong_rtz(uchar);
08294 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtz(uchar);
08295 ulong const_func __attribute__((overloadable)) convert_ulong_rtp(uchar);
08296 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtp(uchar);
08297 ulong const_func __attribute__((overloadable)) convert_ulong_rtn(uchar);
08298 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtn(uchar);
08299 ulong const_func __attribute__((overloadable)) convert_ulong(uchar);
08300 ulong const_func __attribute__((overloadable)) convert_ulong_sat(uchar);
08301 ulong const_func __attribute__((overloadable)) convert_ulong_rte(short);
08302 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rte(short);
08303 ulong const_func __attribute__((overloadable)) convert_ulong_rtz(short);
08304 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtz(short);
08305 ulong const_func __attribute__((overloadable)) convert_ulong_rtp(short);
08306 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtp(short);
08307 ulong const_func __attribute__((overloadable)) convert_ulong_rtn(short);
08308 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtn(short);
08309 ulong const_func __attribute__((overloadable)) convert_ulong(short);
08310 ulong const_func __attribute__((overloadable)) convert_ulong_sat(short);
08311 ulong const_func __attribute__((overloadable)) convert_ulong_rte(ushort);
08312 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rte(ushort);
08313 ulong const_func __attribute__((overloadable)) convert_ulong_rtz(ushort);
08314 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtz(ushort);
08315 ulong const_func __attribute__((overloadable)) convert_ulong_rtp(ushort);
08316 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtp(ushort);
08317 ulong const_func __attribute__((overloadable)) convert_ulong_rtn(ushort);
08318 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtn(ushort);
08319 ulong const_func __attribute__((overloadable)) convert_ulong(ushort);
08320 ulong const_func __attribute__((overloadable)) convert_ulong_sat(ushort);
08321 ulong const_func __attribute__((overloadable)) convert_ulong_rte(int);
08322 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rte(int);
08323 ulong const_func __attribute__((overloadable)) convert_ulong_rtz(int);
08324 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtz(int);
08325 ulong const_func __attribute__((overloadable)) convert_ulong_rtp(int);
08326 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtp(int);
08327 ulong const_func __attribute__((overloadable)) convert_ulong_rtn(int);
08328 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtn(int);
08329 ulong const_func __attribute__((overloadable)) convert_ulong(int);
08330 ulong const_func __attribute__((overloadable)) convert_ulong_sat(int);
08331 ulong const_func __attribute__((overloadable)) convert_ulong_rte(uint);
08332 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rte(uint);
08333 ulong const_func __attribute__((overloadable)) convert_ulong_rtz(uint);
08334 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtz(uint);
08335 ulong const_func __attribute__((overloadable)) convert_ulong_rtp(uint);
08336 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtp(uint);
08337 ulong const_func __attribute__((overloadable)) convert_ulong_rtn(uint);
08338 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtn(uint);
08339 ulong const_func __attribute__((overloadable)) convert_ulong(uint);
08340 ulong const_func __attribute__((overloadable)) convert_ulong_sat(uint);
08341 ulong const_func __attribute__((overloadable)) convert_ulong_rte(long);
08342 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rte(long);
08343 ulong const_func __attribute__((overloadable)) convert_ulong_rtz(long);
08344 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtz(long);
08345 ulong const_func __attribute__((overloadable)) convert_ulong_rtp(long);
08346 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtp(long);
08347 ulong const_func __attribute__((overloadable)) convert_ulong_rtn(long);
08348 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtn(long);
08349 ulong const_func __attribute__((overloadable)) convert_ulong(long);
08350 ulong const_func __attribute__((overloadable)) convert_ulong_sat(long);
08351 ulong const_func __attribute__((overloadable)) convert_ulong_rte(ulong);
08352 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rte(ulong);
08353 ulong const_func __attribute__((overloadable)) convert_ulong_rtz(ulong);
08354 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtz(ulong);
08355 ulong const_func __attribute__((overloadable)) convert_ulong_rtp(ulong);
08356 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtp(ulong);
08357 ulong const_func __attribute__((overloadable)) convert_ulong_rtn(ulong);
08358 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtn(ulong);
08359 ulong const_func __attribute__((overloadable)) convert_ulong(ulong);
08360 ulong const_func __attribute__((overloadable)) convert_ulong_sat(ulong);
08361 ulong const_func __attribute__((overloadable)) convert_ulong_rte(float);
08362 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rte(float);
08363 ulong const_func __attribute__((overloadable)) convert_ulong_rtz(float);
08364 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtz(float);
```

```
08365 ulong const_func __attribute__((overloadable)) convert_ulong_rtp(float);
08366 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtp(float);
08367 ulong const_func __attribute__((overloadable)) convert_ulong_rtn(float);
08368 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtn(float);
08369 ulong const_func __attribute__((overloadable)) convert_ulong(float);
08370 ulong const_func __attribute__((overloadable)) convert_ulong_sat(float);
08371 ulong const_func __attribute__((overloadable)) convert_ulong_rte(double);
08372 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rte(double);
08373 ulong const_func __attribute__((overloadable)) convert_ulong_rtz(double);
08374 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtz(double);
08375 ulong const_func __attribute__((overloadable)) convert_ulong_rtp(double);
08376 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtp(double);
08377 ulong const_func __attribute__((overloadable)) convert_ulong_rtn(double);
08378 ulong const_func __attribute__((overloadable)) convert_ulong_sat_rtn(double);
08379 ulong const_func __attribute__((overloadable)) convert_ulong(double);
08380 ulong const_func __attribute__((overloadable)) convert_ulong_sat(double);
08381
08382 float const_func __attribute__((overloadable)) convert_float_rte(char);
08383 float const_func __attribute__((overloadable)) convert_float_rtz(char);
08384 float const_func __attribute__((overloadable)) convert_float_rtp(char);
08385 float const_func __attribute__((overloadable)) convert_float_rtn(char);
08386 float const_func __attribute__((overloadable)) convert_float(char);
08387 float const_func __attribute__((overloadable)) convert_float_rte(uchar);
08388 float const_func __attribute__((overloadable)) convert_float_rtz(uchar);
08389 float const_func __attribute__((overloadable)) convert_float_rtp(uchar);
08390 float const_func __attribute__((overloadable)) convert_float_rtn(uchar);
08391 float const_func __attribute__((overloadable)) convert_float(uchar);
08392 float const_func __attribute__((overloadable)) convert_float_rte(short);
08393 float const_func __attribute__((overloadable)) convert_float_rtz(short);
08394 float const_func __attribute__((overloadable)) convert_float_rtp(short);
08395 float const_func __attribute__((overloadable)) convert_float_rtn(short);
08396 float const_func __attribute__((overloadable)) convert_float(short);
08397 float const_func __attribute__((overloadable)) convert_float_rte(ushort);
08398 float const_func __attribute__((overloadable)) convert_float_rtz(ushort);
08399 float const_func __attribute__((overloadable)) convert_float_rtp(ushort);
08400 float const_func __attribute__((overloadable)) convert_float_rtn(ushort);
08401 float const_func __attribute__((overloadable)) convert_float(ushort);
08402 float const_func __attribute__((overloadable)) convert_float_rte(int);
08403 float const_func __attribute__((overloadable)) convert_float_rtz(int);
08404 float const_func __attribute__((overloadable)) convert_float_rtp(int);
08405 float const_func __attribute__((overloadable)) convert_float_rtn(int);
08406 float const_func __attribute__((overloadable)) convert_float(int);
08407 float const_func __attribute__((overloadable)) convert_float_rte(uint);
08408 float const_func __attribute__((overloadable)) convert_float_rtz(uint);
08409 float const_func __attribute__((overloadable)) convert_float_rtp(uint);
08410 float const_func __attribute__((overloadable)) convert_float_rtn(uint);
08411 float const_func __attribute__((overloadable)) convert_float(uint);
08412 float const_func __attribute__((overloadable)) convert_float_rte(long);
08413 float const_func __attribute__((overloadable)) convert_float_rtz(long);
08414 float const_func __attribute__((overloadable)) convert_float_rtp(long);
08415 float const_func __attribute__((overloadable)) convert_float_rtn(long);
08416 float const_func __attribute__((overloadable)) convert_float(long);
08417 float const_func __attribute__((overloadable)) convert_float_rte(ulong);
08418 float const_func __attribute__((overloadable)) convert_float_rtz(ulong);
08419 float const_func __attribute__((overloadable)) convert_float_rtp(ulong);
08420 float const_func __attribute__((overloadable)) convert_float_rtn(ulong);
08421 float const_func __attribute__((overloadable)) convert_float(ulong);
08422 float const_func __attribute__((overloadable)) convert_float_rte(float);
08423 float const_func __attribute__((overloadable)) convert_float_rtz(float);
08424 float const_func __attribute__((overloadable)) convert_float_rtp(float);
08425 float const_func __attribute__((overloadable)) convert_float_rtn(float);
08426 float const_func __attribute__((overloadable)) convert_float(float);
08427 float const_func __attribute__((overloadable)) convert_float_rte(double);
08428 float const_func __attribute__((overloadable)) convert_float_rtz(double);
08429 float const_func __attribute__((overloadable)) convert_float_rtp(double);
08430 float const_func __attribute__((overloadable)) convert_float_rtn(double);
08431 float const_func __attribute__((overloadable)) convert_float(double);
08432
08433 double const_func __attribute__((overloadable)) convert_double_rte(char);
08434 double const_func __attribute__((overloadable)) convert_double_rtz(char);
08435 double const_func __attribute__((overloadable)) convert_double_rtp(char);
08436 double const_func __attribute__((overloadable)) convert_double_rtn(char);
08437 double const_func __attribute__((overloadable)) convert_double(char);
08438 double const_func __attribute__((overloadable)) convert_double_rte(uchar);
08439 double const_func __attribute__((overloadable)) convert_double_rtz(uchar);
08440 double const_func __attribute__((overloadable)) convert_double_rtp(uchar);
08441 double const_func __attribute__((overloadable)) convert_double_rtn(uchar);
08442 double const_func __attribute__((overloadable)) convert_double(uchar);
08443 double const_func __attribute__((overloadable)) convert_double_rte(short);
08444 double const_func __attribute__((overloadable)) convert_double_rtz(short);
08445 double const_func __attribute__((overloadable)) convert_double_rtp(short);
08446 double const_func __attribute__((overloadable)) convert_double_rtn(short);
08447 double const_func __attribute__((overloadable)) convert_double(short);
08448 double const_func __attribute__((overloadable)) convert_double_rte(ushort);
08449 double const_func __attribute__((overloadable)) convert_double_rtz(ushort);
08450 double const_func __attribute__((overloadable)) convert_double_rtp(ushort);
08451 double const_func __attribute__((overloadable)) convert_double_rtn(ushort);
```

```
08452 double const_func __attribute__((overloadable)) convert_double(ushort);
08453 double const_func __attribute__((overloadable)) convert_double_rte(int);
08454 double const_func __attribute__((overloadable)) convert_double_rtz(int);
08455 double const_func __attribute__((overloadable)) convert_double_rtp(int);
08456 double const_func __attribute__((overloadable)) convert_double_rtn(int);
08457 double const_func __attribute__((overloadable)) convert_double(int);
08458 double const_func __attribute__((overloadable)) convert_double_rte(uint);
08459 double const_func __attribute__((overloadable)) convert_double_rtz(uint);
08460 double const_func __attribute__((overloadable)) convert_double_rtp(uint);
08461 double const_func __attribute__((overloadable)) convert_double_rtn(uint);
08462 double const_func __attribute__((overloadable)) convert_double(uint);
08463 double const_func __attribute__((overloadable)) convert_double_rte(long);
08464 double const_func __attribute__((overloadable)) convert_double_rtz(long);
08465 double const_func __attribute__((overloadable)) convert_double_rtp(long);
08466 double const_func __attribute__((overloadable)) convert_double_rtn(long);
08467 double const_func __attribute__((overloadable)) convert_double(long);
08468 double const_func __attribute__((overloadable)) convert_double_rte(ulong);
08469 double const_func __attribute__((overloadable)) convert_double_rtz(ulong);
08470 double const_func __attribute__((overloadable)) convert_double_rtp(ulong);
08471 double const_func __attribute__((overloadable)) convert_double_rtn(ulong);
08472 double const_func __attribute__((overloadable)) convert_double(ulong);
08473 double const_func __attribute__((overloadable)) convert_double_rte(float);
08474 double const_func __attribute__((overloadable)) convert_double_rtz(float);
08475 double const_func __attribute__((overloadable)) convert_double_rtp(float);
08476 double const_func __attribute__((overloadable)) convert_double_rtn(float);
08477 double const_func __attribute__((overloadable)) convert_double(float);
08478 double const_func __attribute__((overloadable)) convert_double_rte(double);
08479 double const_func __attribute__((overloadable)) convert_double_rtz(double);
08480 double const_func __attribute__((overloadable)) convert_double_rtp(double);
08481 double const_func __attribute__((overloadable)) convert_double_rtn(double);
08482 double const_func __attribute__((overloadable)) convert_double(double);
08483
08484 char2 const_func __attribute__((overloadable)) convert_char2_rte(char2);
08485 char2 const_func __attribute__((overloadable)) convert_char2_sat_rte(char2);
08486 char2 const_func __attribute__((overloadable)) convert_char2_rtz(char2);
08487 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtz(char2);
08488 char2 const_func __attribute__((overloadable)) convert_char2_rtp(char2);
08489 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtp(char2);
08490 char2 const_func __attribute__((overloadable)) convert_char2_rtn(char2);
08491 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtn(char2);
08492 char2 const_func __attribute__((overloadable)) convert_char2(char2);
08493 char2 const_func __attribute__((overloadable)) convert_char2_sat(char2);
08494 char2 const_func __attribute__((overloadable)) convert_char2_rte(uchar2);
08495 char2 const_func __attribute__((overloadable)) convert_char2_sat_rte(uchar2);
08496 char2 const_func __attribute__((overloadable)) convert_char2_rtz(uchar2);
08497 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtz(uchar2);
08498 char2 const_func __attribute__((overloadable)) convert_char2_rtp(uchar2);
08499 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtp(uchar2);
08500 char2 const_func __attribute__((overloadable)) convert_char2_rtn(uchar2);
08501 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtn(uchar2);
08502 char2 const_func __attribute__((overloadable)) convert_char2(uchar2);
08503 char2 const_func __attribute__((overloadable)) convert_char2_sat(uchar2);
08504 char2 const_func __attribute__((overloadable)) convert_char2_rte(short2);
08505 char2 const_func __attribute__((overloadable)) convert_char2_sat_rte(short2);
08506 char2 const_func __attribute__((overloadable)) convert_char2_rtz(short2);
08507 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtz(short2);
08508 char2 const_func __attribute__((overloadable)) convert_char2_rtp(short2);
08509 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtp(short2);
08510 char2 const_func __attribute__((overloadable)) convert_char2_rtn(short2);
08511 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtn(short2);
08512 char2 const_func __attribute__((overloadable)) convert_char2(short2);
08513 char2 const_func __attribute__((overloadable)) convert_char2_sat(short2);
08514 char2 const_func __attribute__((overloadable)) convert_char2_rte(ushort2);
08515 char2 const_func __attribute__((overloadable)) convert_char2_sat_rte(ushort2);
08516 char2 const_func __attribute__((overloadable)) convert_char2_rtz(ushort2);
08517 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtz(ushort2);
08518 char2 const_func __attribute__((overloadable)) convert_char2_rtp(ushort2);
08519 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtp(ushort2);
08520 char2 const_func __attribute__((overloadable)) convert_char2_rtn(ushort2);
08521 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtn(ushort2);
08522 char2 const_func __attribute__((overloadable)) convert_char2(ushort2);
08523 char2 const_func __attribute__((overloadable)) convert_char2_sat(ushort2);
08524 char2 const_func __attribute__((overloadable)) convert_char2_rte(int2);
08525 char2 const_func __attribute__((overloadable)) convert_char2_sat_rte(int2);
08526 char2 const_func __attribute__((overloadable)) convert_char2_rtz(int2);
08527 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtz(int2);
08528 char2 const_func __attribute__((overloadable)) convert_char2_rtp(int2);
08529 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtp(int2);
08530 char2 const_func __attribute__((overloadable)) convert_char2_rtn(int2);
08531 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtn(int2);
08532 char2 const_func __attribute__((overloadable)) convert_char2(int2);
08533 char2 const_func __attribute__((overloadable)) convert_char2_sat(int2);
08534 char2 const_func __attribute__((overloadable)) convert_char2_rte(uint2);
08535 char2 const_func __attribute__((overloadable)) convert_char2_sat_rte(uint2);
08536 char2 const_func __attribute__((overloadable)) convert_char2_rtz(uint2);
08537 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtz(uint2);
08538 char2 const_func __attribute__((overloadable)) convert_char2_rtp(uint2);
```

```
08539 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtp(uint2);
08540 char2 const_func __attribute__((overloadable)) convert_char2_rtn(uint2);
08541 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtn(uint2);
08542 char2 const_func __attribute__((overloadable)) convert_char2(uint2);
08543 char2 const_func __attribute__((overloadable)) convert_char2_sat(uint2);
08544 char2 const_func __attribute__((overloadable)) convert_char2_rte(long2);
08545 char2 const_func __attribute__((overloadable)) convert_char2_sat_rte(long2);
08546 char2 const_func __attribute__((overloadable)) convert_char2_rtz(long2);
08547 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtz(long2);
08548 char2 const_func __attribute__((overloadable)) convert_char2_rtp(long2);
08549 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtp(long2);
08550 char2 const_func __attribute__((overloadable)) convert_char2_rtn(long2);
08551 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtn(long2);
08552 char2 const_func __attribute__((overloadable)) convert_char2(long2);
08553 char2 const_func __attribute__((overloadable)) convert_char2_sat(long2);
08554 char2 const_func __attribute__((overloadable)) convert_char2_rte(ulong2);
08555 char2 const_func __attribute__((overloadable)) convert_char2_sat_rte(ulong2);
08556 char2 const_func __attribute__((overloadable)) convert_char2_rtz(ulong2);
08557 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtz(ulong2);
08558 char2 const_func __attribute__((overloadable)) convert_char2_rtp(ulong2);
08559 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtp(ulong2);
08560 char2 const_func __attribute__((overloadable)) convert_char2_rtn(ulong2);
08561 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtn(ulong2);
08562 char2 const_func __attribute__((overloadable)) convert_char2(ulong2);
08563 char2 const_func __attribute__((overloadable)) convert_char2_sat(ulong2);
08564 char2 const_func __attribute__((overloadable)) convert_char2_rte(float2);
08565 char2 const_func __attribute__((overloadable)) convert_char2_sat_rte(float2);
08566 char2 const_func __attribute__((overloadable)) convert_char2_rtz(float2);
08567 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtz(float2);
08568 char2 const_func __attribute__((overloadable)) convert_char2_rtp(float2);
08569 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtp(float2);
08570 char2 const_func __attribute__((overloadable)) convert_char2_rtn(float2);
08571 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtn(float2);
08572 char2 const_func __attribute__((overloadable)) convert_char2(float2);
08573 char2 const_func __attribute__((overloadable)) convert_char2_sat(float2);
08574 char2 const_func __attribute__((overloadable)) convert_char2_rte(double2);
08575 char2 const_func __attribute__((overloadable)) convert_char2_sat_rte(double2);
08576 char2 const_func __attribute__((overloadable)) convert_char2_rtz(double2);
08577 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtz(double2);
08578 char2 const_func __attribute__((overloadable)) convert_char2_rtp(double2);
08579 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtp(double2);
08580 char2 const_func __attribute__((overloadable)) convert_char2_rtn(double2);
08581 char2 const_func __attribute__((overloadable)) convert_char2_sat_rtn(double2);
08582 char2 const_func __attribute__((overloadable)) convert_char2(double2);
08583 char2 const_func __attribute__((overloadable)) convert_char2_sat(double2);
08584
08585 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rte(char2);
08586 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rte(char2);
08587 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtz(char2);
08588 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtz(char2);
08589 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtp(char2);
08590 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtp(char2);
08591 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtn(char2);
08592 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtn(char2);
08593 uchar2 const_func __attribute__((overloadable)) convert_uchar2(char2);
08594 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat(char2);
08595 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rte(uchar2);
08596 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rte(uchar2);
08597 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtz(uchar2);
08598 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtz(uchar2);
08599 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtp(uchar2);
08600 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtp(uchar2);
08601 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtn(uchar2);
08602 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtn(uchar2);
08603 uchar2 const_func __attribute__((overloadable)) convert_uchar2(uchar2);
08604 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat(uchar2);
08605 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rte(short2);
08606 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rte(short2);
08607 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtz(short2);
08608 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtz(short2);
08609 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtp(short2);
08610 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtp(short2);
08611 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtn(short2);
08612 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtn(short2);
08613 uchar2 const_func __attribute__((overloadable)) convert_uchar2(short2);
08614 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat(short2);
08615 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rte(ushort2);
08616 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rte(ushort2);
08617 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtz(ushort2);
08618 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtz(ushort2);
08619 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtp(ushort2);
08620 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtp(ushort2);
08621 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtn(ushort2);
08622 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtn(ushort2);
08623 uchar2 const_func __attribute__((overloadable)) convert_uchar2(ushort2);
08624 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat(ushort2);
08625 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rte(int2);
```

```
08626 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rte(int2);
08627 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtz(int2);
08628 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtz(int2);
08629 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtp(int2);
08630 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtp(int2);
08631 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtn(int2);
08632 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtn(int2);
08633 uchar2 const_func __attribute__((overloadable)) convert_uchar2(int2);
08634 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat(int2);
08635 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rte(uint2);
08636 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rte(uint2);
08637 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtz(uint2);
08638 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtz(uint2);
08639 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtp(uint2);
08640 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtp(uint2);
08641 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtn(uint2);
08642 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtn(uint2);
08643 uchar2 const_func __attribute__((overloadable)) convert_uchar2(uint2);
08644 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat(uint2);
08645 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rte(long2);
08646 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rte(long2);
08647 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtz(long2);
08648 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtz(long2);
08649 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtp(long2);
08650 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtp(long2);
08651 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtn(long2);
08652 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtn(long2);
08653 uchar2 const_func __attribute__((overloadable)) convert_uchar2(long2);
08654 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat(long2);
08655 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rte(ulong2);
08656 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rte(ulong2);
08657 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtz(ulong2);
08658 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtz(ulong2);
08659 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtp(ulong2);
08660 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtp(ulong2);
08661 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtn(ulong2);
08662 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtn(ulong2);
08663 uchar2 const_func __attribute__((overloadable)) convert_uchar2(ulong2);
08664 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat(ulong2);
08665 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rte(float2);
08666 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rte(float2);
08667 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtz(float2);
08668 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtz(float2);
08669 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtp(float2);
08670 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtp(float2);
08671 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtn(float2);
08672 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtn(float2);
08673 uchar2 const_func __attribute__((overloadable)) convert_uchar2(float2);
08674 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat(float2);
08675 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rte(double2);
08676 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rte(double2);
08677 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtz(double2);
08678 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtz(double2);
08679 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtp(double2);
08680 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtp(double2);
08681 uchar2 const_func __attribute__((overloadable)) convert_uchar2_rtn(double2);
08682 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat_rtn(double2);
08683 uchar2 const_func __attribute__((overloadable)) convert_uchar2(double2);
08684 uchar2 const_func __attribute__((overloadable)) convert_uchar2_sat(double2);
08685
08686 short2 const_func __attribute__((overloadable)) convert_short2_rte(char2);
08687 short2 const_func __attribute__((overloadable)) convert_short2_sat_rte(char2);
08688 short2 const_func __attribute__((overloadable)) convert_short2_rtz(char2);
08689 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtz(char2);
08690 short2 const_func __attribute__((overloadable)) convert_short2_rtp(char2);
08691 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtp(char2);
08692 short2 const_func __attribute__((overloadable)) convert_short2_rtn(char2);
08693 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtn(char2);
08694 short2 const_func __attribute__((overloadable)) convert_short2(char2);
08695 short2 const_func __attribute__((overloadable)) convert_short2_sat(char2);
08696 short2 const_func __attribute__((overloadable)) convert_short2_rte(uchar2);
08697 short2 const_func __attribute__((overloadable)) convert_short2_sat_rte(uchar2);
08698 short2 const_func __attribute__((overloadable)) convert_short2_rtz(uchar2);
08699 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtz(uchar2);
08700 short2 const_func __attribute__((overloadable)) convert_short2_rtp(uchar2);
08701 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtp(uchar2);
08702 short2 const_func __attribute__((overloadable)) convert_short2_rtn(uchar2);
08703 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtn(uchar2);
08704 short2 const_func __attribute__((overloadable)) convert_short2(uchar2);
08705 short2 const_func __attribute__((overloadable)) convert_short2_sat(uchar2);
08706 short2 const_func __attribute__((overloadable)) convert_short2_rte(short2);
08707 short2 const_func __attribute__((overloadable)) convert_short2_sat_rte(short2);
08708 short2 const_func __attribute__((overloadable)) convert_short2_rtz(short2);
08709 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtz(short2);
08710 short2 const_func __attribute__((overloadable)) convert_short2_rtp(short2);
08711 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtp(short2);
08712 short2 const_func __attribute__((overloadable)) convert_short2_rtn(short2);
```

```
08713 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtn(short2);
08714 short2 const_func __attribute__((overloadable)) convert_short2(short2);
08715 short2 const_func __attribute__((overloadable)) convert_short2_sat(short2);
08716 short2 const_func __attribute__((overloadable)) convert_short2_rte(ushort2);
08717 short2 const_func __attribute__((overloadable)) convert_short2_sat_rte(ushort2);
08718 short2 const_func __attribute__((overloadable)) convert_short2_rtz(ushort2);
08719 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtz(ushort2);
08720 short2 const_func __attribute__((overloadable)) convert_short2_rtp(ushort2);
08721 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtp(ushort2);
08722 short2 const_func __attribute__((overloadable)) convert_short2_rtn(ushort2);
08723 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtn(ushort2);
08724 short2 const_func __attribute__((overloadable)) convert_short2(ushort2);
08725 short2 const_func __attribute__((overloadable)) convert_short2_sat(ushort2);
08726 short2 const_func __attribute__((overloadable)) convert_short2_rte(int2);
08727 short2 const_func __attribute__((overloadable)) convert_short2_sat_rte(int2);
08728 short2 const_func __attribute__((overloadable)) convert_short2_rtz(int2);
08729 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtz(int2);
08730 short2 const_func __attribute__((overloadable)) convert_short2_rtp(int2);
08731 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtp(int2);
08732 short2 const_func __attribute__((overloadable)) convert_short2_rtn(int2);
08733 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtn(int2);
08734 short2 const_func __attribute__((overloadable)) convert_short2(int2);
08735 short2 const_func __attribute__((overloadable)) convert_short2_sat(int2);
08736 short2 const_func __attribute__((overloadable)) convert_short2_rte(uint2);
08737 short2 const_func __attribute__((overloadable)) convert_short2_sat_rte(uint2);
08738 short2 const_func __attribute__((overloadable)) convert_short2_rtz(uint2);
08739 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtz(uint2);
08740 short2 const_func __attribute__((overloadable)) convert_short2_rtp(uint2);
08741 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtp(uint2);
08742 short2 const_func __attribute__((overloadable)) convert_short2_rtn(uint2);
08743 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtn(uint2);
08744 short2 const_func __attribute__((overloadable)) convert_short2(uint2);
08745 short2 const_func __attribute__((overloadable)) convert_short2_sat(uint2);
08746 short2 const_func __attribute__((overloadable)) convert_short2_rte(long2);
08747 short2 const_func __attribute__((overloadable)) convert_short2_sat_rte(long2);
08748 short2 const_func __attribute__((overloadable)) convert_short2_rtz(long2);
08749 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtz(long2);
08750 short2 const_func __attribute__((overloadable)) convert_short2_rtp(long2);
08751 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtp(long2);
08752 short2 const_func __attribute__((overloadable)) convert_short2_rtn(long2);
08753 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtn(long2);
08754 short2 const_func __attribute__((overloadable)) convert_short2(long2);
08755 short2 const_func __attribute__((overloadable)) convert_short2_sat(long2);
08756 short2 const_func __attribute__((overloadable)) convert_short2_rte(ulong2);
08757 short2 const_func __attribute__((overloadable)) convert_short2_sat_rte(ulong2);
08758 short2 const_func __attribute__((overloadable)) convert_short2_rtz(ulong2);
08759 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtz(ulong2);
08760 short2 const_func __attribute__((overloadable)) convert_short2_rtp(ulong2);
08761 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtp(ulong2);
08762 short2 const_func __attribute__((overloadable)) convert_short2_rtn(ulong2);
08763 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtn(ulong2);
08764 short2 const_func __attribute__((overloadable)) convert_short2(ulong2);
08765 short2 const_func __attribute__((overloadable)) convert_short2_sat(ulong2);
08766 short2 const_func __attribute__((overloadable)) convert_short2_rte(float2);
08767 short2 const_func __attribute__((overloadable)) convert_short2_sat_rte(float2);
08768 short2 const_func __attribute__((overloadable)) convert_short2_rtz(float2);
08769 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtz(float2);
08770 short2 const_func __attribute__((overloadable)) convert_short2_rtp(float2);
08771 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtp(float2);
08772 short2 const_func __attribute__((overloadable)) convert_short2_rtn(float2);
08773 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtn(float2);
08774 short2 const_func __attribute__((overloadable)) convert_short2(float2);
08775 short2 const_func __attribute__((overloadable)) convert_short2_sat(float2);
08776 short2 const_func __attribute__((overloadable)) convert_short2_rte(double2);
08777 short2 const_func __attribute__((overloadable)) convert_short2_sat_rte(double2);
08778 short2 const_func __attribute__((overloadable)) convert_short2_rtz(double2);
08779 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtz(double2);
08780 short2 const_func __attribute__((overloadable)) convert_short2_rtp(double2);
08781 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtp(double2);
08782 short2 const_func __attribute__((overloadable)) convert_short2_rtn(double2);
08783 short2 const_func __attribute__((overloadable)) convert_short2_sat_rtn(double2);
08784 short2 const_func __attribute__((overloadable)) convert_short2(double2);
08785 short2 const_func __attribute__((overloadable)) convert_short2_sat(double2);
08786
08787 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rte(char2);
08788 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rte(char2);
08789 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtz(char2);
08790 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtz(char2);
08791 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtp(char2);
08792 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtp(char2);
08793 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtn(char2);
08794 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtn(char2);
08795 ushort2 const_func __attribute__((overloadable)) convert_ushort2(char2);
08796 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat(char2);
08797 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rte(uchar2);
08798 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rte(uchar2);
08799 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtz(uchar2);
```

```
08800 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtz(uchar2);
08801 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtp(uchar2);
08802 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtp(uchar2);
08803 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtn(uchar2);
08804 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtn(uchar2);
08805 ushort2 const_func __attribute__((overloadable)) convert_ushort2(uchar2);
08806 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat(uchar2);
08807 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rte(short2);
08808 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rte(short2);
08809 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtz(short2);
08810 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtz(short2);
08811 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtp(short2);
08812 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtp(short2);
08813 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtn(short2);
08814 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtn(short2);
08815 ushort2 const_func __attribute__((overloadable)) convert_ushort2(short2);
08816 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat(short2);
08817 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rte(ushort2);
08818 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rte(ushort2);
08819 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtz(ushort2);
08820 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtz(ushort2);
08821 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtp(ushort2);
08822 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtp(ushort2);
08823 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtn(ushort2);
08824 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtn(ushort2);
08825 ushort2 const_func __attribute__((overloadable)) convert_ushort2(ushort2);
08826 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat(ushort2);
08827 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rte(int2);
08828 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rte(int2);
08829 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtz(int2);
08830 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtz(int2);
08831 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtp(int2);
08832 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtp(int2);
08833 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtn(int2);
08834 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtn(int2);
08835 ushort2 const_func __attribute__((overloadable)) convert_ushort2(int2);
08836 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat(int2);
08837 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rte(uint2);
08838 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rte(uint2);
08839 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtz(uint2);
08840 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtz(uint2);
08841 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtp(uint2);
08842 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtp(uint2);
08843 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtn(uint2);
08844 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtn(uint2);
08845 ushort2 const_func __attribute__((overloadable)) convert_ushort2(uint2);
08846 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat(uint2);
08847 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rte(long2);
08848 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rte(long2);
08849 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtz(long2);
08850 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtz(long2);
08851 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtp(long2);
08852 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtp(long2);
08853 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtn(long2);
08854 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtn(long2);
08855 ushort2 const_func __attribute__((overloadable)) convert_ushort2(long2);
08856 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat(long2);
08857 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rte(ulong2);
08858 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rte(ulong2);
08859 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtz(ulong2);
08860 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtz(ulong2);
08861 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtp(ulong2);
08862 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtp(ulong2);
08863 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtn(ulong2);
08864 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtn(ulong2);
08865 ushort2 const_func __attribute__((overloadable)) convert_ushort2(ulong2);
08866 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat(ulong2);
08867 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rte(float2);
08868 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rte(float2);
08869 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtz(float2);
08870 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtz(float2);
08871 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtp(float2);
08872 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtp(float2);
08873 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtn(float2);
08874 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtn(float2);
08875 ushort2 const_func __attribute__((overloadable)) convert_ushort2(float2);
08876 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat(float2);
08877 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rte(double2);
08878 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rte(double2);
08879 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtz(double2);
08880 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtz(double2);
08881 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtp(double2);
08882 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtp(double2);
08883 ushort2 const_func __attribute__((overloadable)) convert_ushort2_rtn(double2);
08884 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat_rtn(double2);
08885 ushort2 const_func __attribute__((overloadable)) convert_ushort2(double2);
08886 ushort2 const_func __attribute__((overloadable)) convert_ushort2_sat(double2);
```

```
08887
08888 int2 const_func __attribute__((overloadable)) convert_int2_rte(char2);
08889 int2 const_func __attribute__((overloadable)) convert_int2_sat_rte(char2);
08890 int2 const_func __attribute__((overloadable)) convert_int2_rtz(char2);
08891 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtz(char2);
08892 int2 const_func __attribute__((overloadable)) convert_int2_rtp(char2);
08893 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtp(char2);
08894 int2 const_func __attribute__((overloadable)) convert_int2_rtn(char2);
08895 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtn(char2);
08896 int2 const_func __attribute__((overloadable)) convert_int2(char2);
08897 int2 const_func __attribute__((overloadable)) convert_int2_sat(char2);
08898 int2 const_func __attribute__((overloadable)) convert_int2_rte(uchar2);
08899 int2 const_func __attribute__((overloadable)) convert_int2_sat_rte(uchar2);
08900 int2 const_func __attribute__((overloadable)) convert_int2_rtz(uchar2);
08901 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtz(uchar2);
08902 int2 const_func __attribute__((overloadable)) convert_int2_rtp(uchar2);
08903 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtp(uchar2);
08904 int2 const_func __attribute__((overloadable)) convert_int2_rtn(uchar2);
08905 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtn(uchar2);
08906 int2 const_func __attribute__((overloadable)) convert_int2(uchar2);
08907 int2 const_func __attribute__((overloadable)) convert_int2_sat(uchar2);
08908 int2 const_func __attribute__((overloadable)) convert_int2_rte(short2);
08909 int2 const_func __attribute__((overloadable)) convert_int2_sat_rte(short2);
08910 int2 const_func __attribute__((overloadable)) convert_int2_rtz(short2);
08911 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtz(short2);
08912 int2 const_func __attribute__((overloadable)) convert_int2_rtp(short2);
08913 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtp(short2);
08914 int2 const_func __attribute__((overloadable)) convert_int2_rtn(short2);
08915 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtn(short2);
08916 int2 const_func __attribute__((overloadable)) convert_int2(short2);
08917 int2 const_func __attribute__((overloadable)) convert_int2_sat(short2);
08918 int2 const_func __attribute__((overloadable)) convert_int2_rte(ushort2);
08919 int2 const_func __attribute__((overloadable)) convert_int2_sat_rte(ushort2);
08920 int2 const_func __attribute__((overloadable)) convert_int2_rtz(ushort2);
08921 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtz(ushort2);
08922 int2 const_func __attribute__((overloadable)) convert_int2_rtp(ushort2);
08923 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtp(ushort2);
08924 int2 const_func __attribute__((overloadable)) convert_int2_rtn(ushort2);
08925 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtn(ushort2);
08926 int2 const_func __attribute__((overloadable)) convert_int2(ushort2);
08927 int2 const_func __attribute__((overloadable)) convert_int2_sat(ushort2);
08928 int2 const_func __attribute__((overloadable)) convert_int2_rte(int2);
08929 int2 const_func __attribute__((overloadable)) convert_int2_sat_rte(int2);
08930 int2 const_func __attribute__((overloadable)) convert_int2_rtz(int2);
08931 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtz(int2);
08932 int2 const_func __attribute__((overloadable)) convert_int2_rtp(int2);
08933 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtp(int2);
08934 int2 const_func __attribute__((overloadable)) convert_int2_rtn(int2);
08935 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtn(int2);
08936 int2 const_func __attribute__((overloadable)) convert_int2(int2);
08937 int2 const_func __attribute__((overloadable)) convert_int2_sat(int2);
08938 int2 const_func __attribute__((overloadable)) convert_int2_rte(uint2);
08939 int2 const_func __attribute__((overloadable)) convert_int2_sat_rte(uint2);
08940 int2 const_func __attribute__((overloadable)) convert_int2_rtz(uint2);
08941 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtz(uint2);
08942 int2 const_func __attribute__((overloadable)) convert_int2_rtp(uint2);
08943 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtp(uint2);
08944 int2 const_func __attribute__((overloadable)) convert_int2_rtn(uint2);
08945 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtn(uint2);
08946 int2 const_func __attribute__((overloadable)) convert_int2(uint2);
08947 int2 const_func __attribute__((overloadable)) convert_int2_sat(uint2);
08948 int2 const_func __attribute__((overloadable)) convert_int2_rte(long2);
08949 int2 const_func __attribute__((overloadable)) convert_int2_sat_rte(long2);
08950 int2 const_func __attribute__((overloadable)) convert_int2_rtz(long2);
08951 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtz(long2);
08952 int2 const_func __attribute__((overloadable)) convert_int2_rtp(long2);
08953 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtp(long2);
08954 int2 const_func __attribute__((overloadable)) convert_int2_rtn(long2);
08955 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtn(long2);
08956 int2 const_func __attribute__((overloadable)) convert_int2(long2);
08957 int2 const_func __attribute__((overloadable)) convert_int2_sat(long2);
08958 int2 const_func __attribute__((overloadable)) convert_int2_rte(ulong2);
08959 int2 const_func __attribute__((overloadable)) convert_int2_sat_rte(ulong2);
08960 int2 const_func __attribute__((overloadable)) convert_int2_rtz(ulong2);
08961 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtz(ulong2);
08962 int2 const_func __attribute__((overloadable)) convert_int2_rtp(ulong2);
08963 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtp(ulong2);
08964 int2 const_func __attribute__((overloadable)) convert_int2_rtn(ulong2);
08965 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtn(ulong2);
08966 int2 const_func __attribute__((overloadable)) convert_int2(ulong2);
08967 int2 const_func __attribute__((overloadable)) convert_int2_sat(ulong2);
08968 int2 const_func __attribute__((overloadable)) convert_int2_rte(float2);
08969 int2 const_func __attribute__((overloadable)) convert_int2_sat_rte(float2);
08970 int2 const_func __attribute__((overloadable)) convert_int2_rtz(float2);
08971 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtz(float2);
08972 int2 const_func __attribute__((overloadable)) convert_int2_rtp(float2);
08973 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtp(float2);
```

```
08974 int2 const_func __attribute__((overloadable)) convert_int2_rtn(float2);
08975 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtn(float2);
08976 int2 const_func __attribute__((overloadable)) convert_int2(float2);
08977 int2 const_func __attribute__((overloadable)) convert_int2_sat(float2);
08978 int2 const_func __attribute__((overloadable)) convert_int2_rte(double2);
08979 int2 const_func __attribute__((overloadable)) convert_int2_sat_rte(double2);
08980 int2 const_func __attribute__((overloadable)) convert_int2_rtz(double2);
08981 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtz(double2);
08982 int2 const_func __attribute__((overloadable)) convert_int2_rtp(double2);
08983 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtp(double2);
08984 int2 const_func __attribute__((overloadable)) convert_int2_rtn(double2);
08985 int2 const_func __attribute__((overloadable)) convert_int2_sat_rtn(double2);
08986 int2 const_func __attribute__((overloadable)) convert_int2(double2);
08987 int2 const_func __attribute__((overloadable)) convert_int2_sat(double2);
08988
08989 uint2 const_func __attribute__((overloadable)) convert_uint2_rte(char2);
08990 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rte(char2);
08991 uint2 const_func __attribute__((overloadable)) convert_uint2_rtz(char2);
08992 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtz(char2);
08993 uint2 const_func __attribute__((overloadable)) convert_uint2_rtp(char2);
08994 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtp(char2);
08995 uint2 const_func __attribute__((overloadable)) convert_uint2_rtn(char2);
08996 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtn(char2);
08997 uint2 const_func __attribute__((overloadable)) convert_uint2(char2);
08998 uint2 const_func __attribute__((overloadable)) convert_uint2_sat(char2);
08999 uint2 const_func __attribute__((overloadable)) convert_uint2_rte(uchar2);
09000 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rte(uchar2);
09001 uint2 const_func __attribute__((overloadable)) convert_uint2_rtz(uchar2);
09002 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtz(uchar2);
09003 uint2 const_func __attribute__((overloadable)) convert_uint2_rtp(uchar2);
09004 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtp(uchar2);
09005 uint2 const_func __attribute__((overloadable)) convert_uint2_rtn(uchar2);
09006 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtn(uchar2);
09007 uint2 const_func __attribute__((overloadable)) convert_uint2(uchar2);
09008 uint2 const_func __attribute__((overloadable)) convert_uint2_sat(uchar2);
09009 uint2 const_func __attribute__((overloadable)) convert_uint2_rte(short2);
09010 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rte(short2);
09011 uint2 const_func __attribute__((overloadable)) convert_uint2_rtz(short2);
09012 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtz(short2);
09013 uint2 const_func __attribute__((overloadable)) convert_uint2_rtp(short2);
09014 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtp(short2);
09015 uint2 const_func __attribute__((overloadable)) convert_uint2_rtn(short2);
09016 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtn(short2);
09017 uint2 const_func __attribute__((overloadable)) convert_uint2(short2);
09018 uint2 const_func __attribute__((overloadable)) convert_uint2_sat(short2);
09019 uint2 const_func __attribute__((overloadable)) convert_uint2_rte(ushort2);
09020 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rte(ushort2);
09021 uint2 const_func __attribute__((overloadable)) convert_uint2_rtz(ushort2);
09022 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtz(ushort2);
09023 uint2 const_func __attribute__((overloadable)) convert_uint2_rtp(ushort2);
09024 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtp(ushort2);
09025 uint2 const_func __attribute__((overloadable)) convert_uint2_rtn(ushort2);
09026 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtn(ushort2);
09027 uint2 const_func __attribute__((overloadable)) convert_uint2(ushort2);
09028 uint2 const_func __attribute__((overloadable)) convert_uint2_sat(ushort2);
09029 uint2 const_func __attribute__((overloadable)) convert_uint2_rte(int2);
09030 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rte(int2);
09031 uint2 const_func __attribute__((overloadable)) convert_uint2_rtz(int2);
09032 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtz(int2);
09033 uint2 const_func __attribute__((overloadable)) convert_uint2_rtp(int2);
09034 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtp(int2);
09035 uint2 const_func __attribute__((overloadable)) convert_uint2_rtn(int2);
09036 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtn(int2);
09037 uint2 const_func __attribute__((overloadable)) convert_uint2(int2);
09038 uint2 const_func __attribute__((overloadable)) convert_uint2_sat(int2);
09039 uint2 const_func __attribute__((overloadable)) convert_uint2_rte(uint2);
09040 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rte(uint2);
09041 uint2 const_func __attribute__((overloadable)) convert_uint2_rtz(uint2);
09042 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtz(uint2);
09043 uint2 const_func __attribute__((overloadable)) convert_uint2_rtp(uint2);
09044 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtp(uint2);
09045 uint2 const_func __attribute__((overloadable)) convert_uint2_rtn(uint2);
09046 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtn(uint2);
09047 uint2 const_func __attribute__((overloadable)) convert_uint2(uint2);
09048 uint2 const_func __attribute__((overloadable)) convert_uint2_sat(uint2);
09049 uint2 const_func __attribute__((overloadable)) convert_uint2_rte(long2);
09050 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rte(long2);
09051 uint2 const_func __attribute__((overloadable)) convert_uint2_rtz(long2);
09052 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtz(long2);
09053 uint2 const_func __attribute__((overloadable)) convert_uint2_rtp(long2);
09054 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtp(long2);
09055 uint2 const_func __attribute__((overloadable)) convert_uint2_rtn(long2);
09056 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtn(long2);
09057 uint2 const_func __attribute__((overloadable)) convert_uint2(long2);
09058 uint2 const_func __attribute__((overloadable)) convert_uint2_sat(long2);
09059 uint2 const_func __attribute__((overloadable)) convert_uint2_rte(ulong2);
09060 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rte(ulong2);
```

```
09061 uint2 const_func __attribute__((overloadable)) convert_uint2_rtz(ulong2);
09062 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtz(ulong2);
09063 uint2 const_func __attribute__((overloadable)) convert_uint2_rtp(ulong2);
09064 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtp(ulong2);
09065 uint2 const_func __attribute__((overloadable)) convert_uint2_rtn(ulong2);
09066 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtn(ulong2);
09067 uint2 const_func __attribute__((overloadable)) convert_uint2(ulong2);
09068 uint2 const_func __attribute__((overloadable)) convert_uint2_sat(ulong2);
09069 uint2 const_func __attribute__((overloadable)) convert_uint2_rte(float2);
09070 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rte(float2);
09071 uint2 const_func __attribute__((overloadable)) convert_uint2_rtz(float2);
09072 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtz(float2);
09073 uint2 const_func __attribute__((overloadable)) convert_uint2_rtp(float2);
09074 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtp(float2);
09075 uint2 const_func __attribute__((overloadable)) convert_uint2_rtn(float2);
09076 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtn(float2);
09077 uint2 const_func __attribute__((overloadable)) convert_uint2(float2);
09078 uint2 const_func __attribute__((overloadable)) convert_uint2_sat(float2);
09079 uint2 const_func __attribute__((overloadable)) convert_uint2_rte(double2);
09080 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rte(double2);
09081 uint2 const_func __attribute__((overloadable)) convert_uint2_rtz(double2);
09082 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtz(double2);
09083 uint2 const_func __attribute__((overloadable)) convert_uint2_rtp(double2);
09084 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtp(double2);
09085 uint2 const_func __attribute__((overloadable)) convert_uint2_rtn(double2);
09086 uint2 const_func __attribute__((overloadable)) convert_uint2_sat_rtn(double2);
09087 uint2 const_func __attribute__((overloadable)) convert_uint2(double2);
09088 uint2 const_func __attribute__((overloadable)) convert_uint2_sat(double2);
09089 long2 const_func __attribute__((overloadable)) convert_long2_rte(char2);
09090 long2 const_func __attribute__((overloadable)) convert_long2_sat_rte(char2);
09091 long2 const_func __attribute__((overloadable)) convert_long2_rtz(char2);
09092 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtz(char2);
09093 long2 const_func __attribute__((overloadable)) convert_long2_rtp(char2);
09094 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtp(char2);
09095 long2 const_func __attribute__((overloadable)) convert_long2_rtn(char2);
09096 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtn(char2);
09097 long2 const_func __attribute__((overloadable)) convert_long2(char2);
09098 long2 const_func __attribute__((overloadable)) convert_long2_sat(char2);
09099 long2 const_func __attribute__((overloadable)) convert_long2_rte(uchar2);
09100 long2 const_func __attribute__((overloadable)) convert_long2_sat_rte(uchar2);
09101 long2 const_func __attribute__((overloadable)) convert_long2_rtz(uchar2);
09102 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtz(uchar2);
09103 long2 const_func __attribute__((overloadable)) convert_long2_rtp(uchar2);
09104 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtp(uchar2);
09105 long2 const_func __attribute__((overloadable)) convert_long2_rtn(uchar2);
09106 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtn(uchar2);
09107 long2 const_func __attribute__((overloadable)) convert_long2(uchar2);
09108 long2 const_func __attribute__((overloadable)) convert_long2_sat(uchar2);
09109 long2 const_func __attribute__((overloadable)) convert_long2_rte(short2);
09110 long2 const_func __attribute__((overloadable)) convert_long2_sat_rte(short2);
09111 long2 const_func __attribute__((overloadable)) convert_long2_rtz(short2);
09112 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtz(short2);
09113 long2 const_func __attribute__((overloadable)) convert_long2_rtp(short2);
09114 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtp(short2);
09115 long2 const_func __attribute__((overloadable)) convert_long2_rtn(short2);
09116 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtn(short2);
09117 long2 const_func __attribute__((overloadable)) convert_long2(short2);
09118 long2 const_func __attribute__((overloadable)) convert_long2_sat(short2);
09119 long2 const_func __attribute__((overloadable)) convert_long2_rte(ushort2);
09120 long2 const_func __attribute__((overloadable)) convert_long2_sat_rte(ushort2);
09121 long2 const_func __attribute__((overloadable)) convert_long2_rtz(ushort2);
09122 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtz(ushort2);
09123 long2 const_func __attribute__((overloadable)) convert_long2_rtp(ushort2);
09124 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtp(ushort2);
09125 long2 const_func __attribute__((overloadable)) convert_long2_rtn(ushort2);
09126 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtn(ushort2);
09127 long2 const_func __attribute__((overloadable)) convert_long2(ushort2);
09128 long2 const_func __attribute__((overloadable)) convert_long2_sat(ushort2);
09129 long2 const_func __attribute__((overloadable)) convert_long2_rte(int2);
09130 long2 const_func __attribute__((overloadable)) convert_long2_sat_rte(int2);
09131 long2 const_func __attribute__((overloadable)) convert_long2_rtz(int2);
09132 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtz(int2);
09133 long2 const_func __attribute__((overloadable)) convert_long2_rtp(int2);
09134 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtp(int2);
09135 long2 const_func __attribute__((overloadable)) convert_long2_rtn(int2);
09136 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtn(int2);
09137 long2 const_func __attribute__((overloadable)) convert_long2(int2);
09138 long2 const_func __attribute__((overloadable)) convert_long2_sat(int2);
09139 long2 const_func __attribute__((overloadable)) convert_long2_rte(uint2);
09140 long2 const_func __attribute__((overloadable)) convert_long2_sat_rte(uint2);
09141 long2 const_func __attribute__((overloadable)) convert_long2_rtz(uint2);
09142 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtz(uint2);
09143 long2 const_func __attribute__((overloadable)) convert_long2_rtp(uint2);
09144 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtp(uint2);
09145 long2 const_func __attribute__((overloadable)) convert_long2_rtn(uint2);
09146 long2 const_func __attribute__((overloadable)) convert_long2_sat_rtn(uint2);
09147 long2 const_func __attribute__((overloadable)) convert_long2(uint2);
```

```
long2 const_func __attribute__((overloadable)) convert_long2_sat(uint2);
long2 const_func __attribute__((overloadable)) convert_long2_rte(long2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rte(long2);
long2 const_func __attribute__((overloadable)) convert_long2_rtz(long2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rtz(long2);
long2 const_func __attribute__((overloadable)) convert_long2_rtp(long2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rtp(long2);
long2 const_func __attribute__((overloadable)) convert_long2_rtn(long2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rtn(long2);
long2 const_func __attribute__((overloadable)) convert_long2(long2);
long2 const_func __attribute__((overloadable)) convert_long2_sat(long2);
long2 const_func __attribute__((overloadable)) convert_long2_rte(ulong2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rte(ulong2);
long2 const_func __attribute__((overloadable)) convert_long2_rtz(ulong2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rtz(ulong2);
long2 const_func __attribute__((overloadable)) convert_long2_rtp(ulong2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rtp(ulong2);
long2 const_func __attribute__((overloadable)) convert_long2_rtn(ulong2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rtn(ulong2);
long2 const_func __attribute__((overloadable)) convert_long2(ulong2);
long2 const_func __attribute__((overloadable)) convert_long2_sat(ulong2);
long2 const_func __attribute__((overloadable)) convert_long2_rte(float2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rte(float2);
long2 const_func __attribute__((overloadable)) convert_long2_rtz(float2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rtz(float2);
long2 const_func __attribute__((overloadable)) convert_long2_rtp(float2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rtp(float2);
long2 const_func __attribute__((overloadable)) convert_long2_rtn(float2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rtn(float2);
long2 const_func __attribute__((overloadable)) convert_long2(float2);
long2 const_func __attribute__((overloadable)) convert_long2_sat(float2);
long2 const_func __attribute__((overloadable)) convert_long2_rte(double2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rte(double2);
long2 const_func __attribute__((overloadable)) convert_long2_rtz(double2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rtz(double2);
long2 const_func __attribute__((overloadable)) convert_long2_rtp(double2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rtp(double2);
long2 const_func __attribute__((overloadable)) convert_long2_rtn(double2);
long2 const_func __attribute__((overloadable)) convert_long2_sat_rtn(double2);
long2 const_func __attribute__((overloadable)) convert_long2(double2);
long2 const_func __attribute__((overloadable)) convert_long2_sat(double2);

ulong2 const_func __attribute__((overloadable)) convert_ulong2_rte(char2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rte(char2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtz(char2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtz(char2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtp(char2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtp(char2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtn(char2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtn(char2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2(char2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat(char2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rte(uchar2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rte(uchar2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtz(uchar2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtz(uchar2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtp(uchar2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtp(uchar2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtn(uchar2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtn(uchar2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2(uchar2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat(uchar2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rte(short2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rte(short2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtz(short2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtz(short2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtp(short2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtp(short2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtn(short2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtn(short2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2(short2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat(short2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rte(ushort2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rte(ushort2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtz(ushort2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtz(ushort2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtp(ushort2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtp(ushort2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtn(ushort2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtn(ushort2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2(ushort2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat(ushort2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rte(int2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rte(int2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtz(int2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtz(int2);
ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtp(int2);
```

```
09235 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtp(int2);
09236 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtn(int2);
09237 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtn(int2);
09238 ulong2 const_func __attribute__((overloadable)) convert_ulong2(int2);
09239 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat(int2);
09240 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rte(uint2);
09241 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rte(uint2);
09242 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtz(uint2);
09243 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtz(uint2);
09244 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtp(uint2);
09245 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtp(uint2);
09246 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtn(uint2);
09247 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtn(uint2);
09248 ulong2 const_func __attribute__((overloadable)) convert_ulong2(uint2);
09249 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat(uint2);
09250 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rte(long2);
09251 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rte(long2);
09252 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtz(long2);
09253 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtz(long2);
09254 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtp(long2);
09255 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtp(long2);
09256 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtn(long2);
09257 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtn(long2);
09258 ulong2 const_func __attribute__((overloadable)) convert_ulong2(long2);
09259 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat(long2);
09260 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rte(ulong2);
09261 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rte(ulong2);
09262 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtz(ulong2);
09263 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtz(ulong2);
09264 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtp(ulong2);
09265 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtp(ulong2);
09266 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtn(ulong2);
09267 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtn(ulong2);
09268 ulong2 const_func __attribute__((overloadable)) convert_ulong2(ulong2);
09269 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat(ulong2);
09270 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rte(float2);
09271 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rte(float2);
09272 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtz(float2);
09273 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtz(float2);
09274 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtp(float2);
09275 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtp(float2);
09276 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtn(float2);
09277 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtn(float2);
09278 ulong2 const_func __attribute__((overloadable)) convert_ulong2(float2);
09279 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat(float2);
09280 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rte(double2);
09281 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rte(double2);
09282 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtz(double2);
09283 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtz(double2);
09284 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtp(double2);
09285 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtp(double2);
09286 ulong2 const_func __attribute__((overloadable)) convert_ulong2_rtn(double2);
09287 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat_rtn(double2);
09288 ulong2 const_func __attribute__((overloadable)) convert_ulong2(double2);
09289 ulong2 const_func __attribute__((overloadable)) convert_ulong2_sat(double2);
09290
09291 float2 const_func __attribute__((overloadable)) convert_float2_rte(char2);
09292 float2 const_func __attribute__((overloadable)) convert_float2_rtz(char2);
09293 float2 const_func __attribute__((overloadable)) convert_float2_rtp(char2);
09294 float2 const_func __attribute__((overloadable)) convert_float2_rtn(char2);
09295 float2 const_func __attribute__((overloadable)) convert_float2(char2);
09296 float2 const_func __attribute__((overloadable)) convert_float2_rte(uchar2);
09297 float2 const_func __attribute__((overloadable)) convert_float2_rtz(uchar2);
09298 float2 const_func __attribute__((overloadable)) convert_float2_rtp(uchar2);
09299 float2 const_func __attribute__((overloadable)) convert_float2_rtn(uchar2);
09300 float2 const_func __attribute__((overloadable)) convert_float2(uchar2);
09301 float2 const_func __attribute__((overloadable)) convert_float2_rte(short2);
09302 float2 const_func __attribute__((overloadable)) convert_float2_rtz(short2);
09303 float2 const_func __attribute__((overloadable)) convert_float2_rtp(short2);
09304 float2 const_func __attribute__((overloadable)) convert_float2_rtn(short2);
09305 float2 const_func __attribute__((overloadable)) convert_float2(short2);
09306 float2 const_func __attribute__((overloadable)) convert_float2_rte(ushort2);
09307 float2 const_func __attribute__((overloadable)) convert_float2_rtz(ushort2);
09308 float2 const_func __attribute__((overloadable)) convert_float2_rtp(ushort2);
09309 float2 const_func __attribute__((overloadable)) convert_float2_rtn(ushort2);
09310 float2 const_func __attribute__((overloadable)) convert_float2(ushort2);
09311 float2 const_func __attribute__((overloadable)) convert_float2_rte(int2);
09312 float2 const_func __attribute__((overloadable)) convert_float2_rtz(int2);
09313 float2 const_func __attribute__((overloadable)) convert_float2_rtp(int2);
09314 float2 const_func __attribute__((overloadable)) convert_float2_rtn(int2);
09315 float2 const_func __attribute__((overloadable)) convert_float2(int2);
09316 float2 const_func __attribute__((overloadable)) convert_float2_rte(uint2);
09317 float2 const_func __attribute__((overloadable)) convert_float2_rtz(uint2);
09318 float2 const_func __attribute__((overloadable)) convert_float2_rtp(uint2);
09319 float2 const_func __attribute__((overloadable)) convert_float2_rtn(uint2);
09320 float2 const_func __attribute__((overloadable)) convert_float2(uint2);
09321 float2 const_func __attribute__((overloadable)) convert_float2_rte(long2);
```

```
09322 float2 const_func __attribute__((overloadable)) convert_float2_rtz(long2);
09323 float2 const_func __attribute__((overloadable)) convert_float2_rtp(long2);
09324 float2 const_func __attribute__((overloadable)) convert_float2_rtn(long2);
09325 float2 const_func __attribute__((overloadable)) convert_float2(long2);
09326 float2 const_func __attribute__((overloadable)) convert_float2_rte(ulong2);
09327 float2 const_func __attribute__((overloadable)) convert_float2_rtz(ulong2);
09328 float2 const_func __attribute__((overloadable)) convert_float2_rtp(ulong2);
09329 float2 const_func __attribute__((overloadable)) convert_float2_rtn(ulong2);
09330 float2 const_func __attribute__((overloadable)) convert_float2(ulong2);
09331 float2 const_func __attribute__((overloadable)) convert_float2_rte(float2);
09332 float2 const_func __attribute__((overloadable)) convert_float2_rtz(float2);
09333 float2 const_func __attribute__((overloadable)) convert_float2_rtp(float2);
09334 float2 const_func __attribute__((overloadable)) convert_float2_rtn(float2);
09335 float2 const_func __attribute__((overloadable)) convert_float2(float2);
09336 float2 const_func __attribute__((overloadable)) convert_float2_rte(double2);
09337 float2 const_func __attribute__((overloadable)) convert_float2_rtz(double2);
09338 float2 const_func __attribute__((overloadable)) convert_float2_rtp(double2);
09339 float2 const_func __attribute__((overloadable)) convert_float2_rtn(double2);
09340 float2 const_func __attribute__((overloadable)) convert_float2(double2);
09341
09342 double2 const_func __attribute__((overloadable)) convert_double2_rte(char2);
09343 double2 const_func __attribute__((overloadable)) convert_double2_rtz(char2);
09344 double2 const_func __attribute__((overloadable)) convert_double2_rtp(char2);
09345 double2 const_func __attribute__((overloadable)) convert_double2_rtn(char2);
09346 double2 const_func __attribute__((overloadable)) convert_double2(char2);
09347 double2 const_func __attribute__((overloadable)) convert_double2_rte(uchar2);
09348 double2 const_func __attribute__((overloadable)) convert_double2_rtz(uchar2);
09349 double2 const_func __attribute__((overloadable)) convert_double2_rtp(uchar2);
09350 double2 const_func __attribute__((overloadable)) convert_double2_rtn(uchar2);
09351 double2 const_func __attribute__((overloadable)) convert_double2(uchar2);
09352 double2 const_func __attribute__((overloadable)) convert_double2_rte(short2);
09353 double2 const_func __attribute__((overloadable)) convert_double2_rtz(short2);
09354 double2 const_func __attribute__((overloadable)) convert_double2_rtp(short2);
09355 double2 const_func __attribute__((overloadable)) convert_double2_rtn(short2);
09356 double2 const_func __attribute__((overloadable)) convert_double2(short2);
09357 double2 const_func __attribute__((overloadable)) convert_double2_rte(ushort2);
09358 double2 const_func __attribute__((overloadable)) convert_double2_rtz(ushort2);
09359 double2 const_func __attribute__((overloadable)) convert_double2_rtp(ushort2);
09360 double2 const_func __attribute__((overloadable)) convert_double2_rtn(ushort2);
09361 double2 const_func __attribute__((overloadable)) convert_double2(ushort2);
09362 double2 const_func __attribute__((overloadable)) convert_double2_rte(int2);
09363 double2 const_func __attribute__((overloadable)) convert_double2_rtz(int2);
09364 double2 const_func __attribute__((overloadable)) convert_double2_rtp(int2);
09365 double2 const_func __attribute__((overloadable)) convert_double2_rtn(int2);
09366 double2 const_func __attribute__((overloadable)) convert_double2(int2);
09367 double2 const_func __attribute__((overloadable)) convert_double2_rte(uint2);
09368 double2 const_func __attribute__((overloadable)) convert_double2_rtz(uint2);
09369 double2 const_func __attribute__((overloadable)) convert_double2_rtp(uint2);
09370 double2 const_func __attribute__((overloadable)) convert_double2_rtn(uint2);
09371 double2 const_func __attribute__((overloadable)) convert_double2(uint2);
09372 double2 const_func __attribute__((overloadable)) convert_double2_rte(long2);
09373 double2 const_func __attribute__((overloadable)) convert_double2_rtz(long2);
09374 double2 const_func __attribute__((overloadable)) convert_double2_rtp(long2);
09375 double2 const_func __attribute__((overloadable)) convert_double2_rtn(long2);
09376 double2 const_func __attribute__((overloadable)) convert_double2(long2);
09377 double2 const_func __attribute__((overloadable)) convert_double2_rte(ulong2);
09378 double2 const_func __attribute__((overloadable)) convert_double2_rtz(ulong2);
09379 double2 const_func __attribute__((overloadable)) convert_double2_rtp(ulong2);
09380 double2 const_func __attribute__((overloadable)) convert_double2_rtn(ulong2);
09381 double2 const_func __attribute__((overloadable)) convert_double2(ulong2);
09382 double2 const_func __attribute__((overloadable)) convert_double2_rte(float2);
09383 double2 const_func __attribute__((overloadable)) convert_double2_rtz(float2);
09384 double2 const_func __attribute__((overloadable)) convert_double2_rtp(float2);
09385 double2 const_func __attribute__((overloadable)) convert_double2_rtn(float2);
09386 double2 const_func __attribute__((overloadable)) convert_double2(float2);
09387 double2 const_func __attribute__((overloadable)) convert_double2_rte(double2);
09388 double2 const_func __attribute__((overloadable)) convert_double2_rtz(double2);
09389 double2 const_func __attribute__((overloadable)) convert_double2_rtp(double2);
09390 double2 const_func __attribute__((overloadable)) convert_double2_rtn(double2);
09391 double2 const_func __attribute__((overloadable)) convert_double2(double2);
09392
09393 char3 const_func __attribute__((overloadable)) convert_char3_rte(char3);
09394 char3 const_func __attribute__((overloadable)) convert_char3_sat_rte(char3);
09395 char3 const_func __attribute__((overloadable)) convert_char3_rtz(char3);
09396 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtz(char3);
09397 char3 const_func __attribute__((overloadable)) convert_char3_rtp(char3);
09398 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtp(char3);
09399 char3 const_func __attribute__((overloadable)) convert_char3_rtn(char3);
09400 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtn(char3);
09401 char3 const_func __attribute__((overloadable)) convert_char3(char3);
09402 char3 const_func __attribute__((overloadable)) convert_char3_sat(char3);
09403 char3 const_func __attribute__((overloadable)) convert_char3_rte(uchar3);
09404 char3 const_func __attribute__((overloadable)) convert_char3_sat_rte(uchar3);
09405 char3 const_func __attribute__((overloadable)) convert_char3_rtz(uchar3);
09406 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtz(uchar3);
09407 char3 const_func __attribute__((overloadable)) convert_char3_rtp(uchar3);
09408 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtp(uchar3);
```

```
09409 char3 const_func __attribute__((overloadable)) convert_char3_rtn(uchar3);
09410 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtn(uchar3);
09411 char3 const_func __attribute__((overloadable)) convert_char3(uchar3);
09412 char3 const_func __attribute__((overloadable)) convert_char3_sat(uchar3);
09413 char3 const_func __attribute__((overloadable)) convert_char3_rte(short3);
09414 char3 const_func __attribute__((overloadable)) convert_char3_sat_rte(short3);
09415 char3 const_func __attribute__((overloadable)) convert_char3_rtz(short3);
09416 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtz(short3);
09417 char3 const_func __attribute__((overloadable)) convert_char3_rtp(short3);
09418 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtp(short3);
09419 char3 const_func __attribute__((overloadable)) convert_char3_rtn(short3);
09420 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtn(short3);
09421 char3 const_func __attribute__((overloadable)) convert_char3(short3);
09422 char3 const_func __attribute__((overloadable)) convert_char3_sat(short3);
09423 char3 const_func __attribute__((overloadable)) convert_char3_rte(ushort3);
09424 char3 const_func __attribute__((overloadable)) convert_char3_sat_rte(ushort3);
09425 char3 const_func __attribute__((overloadable)) convert_char3_rtz(ushort3);
09426 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtz(ushort3);
09427 char3 const_func __attribute__((overloadable)) convert_char3_rtp(ushort3);
09428 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtp(ushort3);
09429 char3 const_func __attribute__((overloadable)) convert_char3_rtn(ushort3);
09430 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtn(ushort3);
09431 char3 const_func __attribute__((overloadable)) convert_char3(ushort3);
09432 char3 const_func __attribute__((overloadable)) convert_char3_sat(ushort3);
09433 char3 const_func __attribute__((overloadable)) convert_char3_rte(int3);
09434 char3 const_func __attribute__((overloadable)) convert_char3_sat_rte(int3);
09435 char3 const_func __attribute__((overloadable)) convert_char3_rtz(int3);
09436 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtz(int3);
09437 char3 const_func __attribute__((overloadable)) convert_char3_rtp(int3);
09438 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtp(int3);
09439 char3 const_func __attribute__((overloadable)) convert_char3_rtn(int3);
09440 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtn(int3);
09441 char3 const_func __attribute__((overloadable)) convert_char3(int3);
09442 char3 const_func __attribute__((overloadable)) convert_char3_sat(int3);
09443 char3 const_func __attribute__((overloadable)) convert_char3_rte(uint3);
09444 char3 const_func __attribute__((overloadable)) convert_char3_sat_rte(uint3);
09445 char3 const_func __attribute__((overloadable)) convert_char3_rtz(uint3);
09446 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtz(uint3);
09447 char3 const_func __attribute__((overloadable)) convert_char3_rtp(uint3);
09448 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtp(uint3);
09449 char3 const_func __attribute__((overloadable)) convert_char3_rtn(uint3);
09450 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtn(uint3);
09451 char3 const_func __attribute__((overloadable)) convert_char3(uint3);
09452 char3 const_func __attribute__((overloadable)) convert_char3_sat(uint3);
09453 char3 const_func __attribute__((overloadable)) convert_char3_rte(long3);
09454 char3 const_func __attribute__((overloadable)) convert_char3_sat_rte(long3);
09455 char3 const_func __attribute__((overloadable)) convert_char3_rtz(long3);
09456 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtz(long3);
09457 char3 const_func __attribute__((overloadable)) convert_char3_rtp(long3);
09458 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtp(long3);
09459 char3 const_func __attribute__((overloadable)) convert_char3_rtn(long3);
09460 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtn(long3);
09461 char3 const_func __attribute__((overloadable)) convert_char3(long3);
09462 char3 const_func __attribute__((overloadable)) convert_char3_sat(long3);
09463 char3 const_func __attribute__((overloadable)) convert_char3_rte(ulong3);
09464 char3 const_func __attribute__((overloadable)) convert_char3_sat_rte(ulong3);
09465 char3 const_func __attribute__((overloadable)) convert_char3_rtz(ulong3);
09466 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtz(ulong3);
09467 char3 const_func __attribute__((overloadable)) convert_char3_rtp(ulong3);
09468 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtp(ulong3);
09469 char3 const_func __attribute__((overloadable)) convert_char3_rtn(ulong3);
09470 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtn(ulong3);
09471 char3 const_func __attribute__((overloadable)) convert_char3(ulong3);
09472 char3 const_func __attribute__((overloadable)) convert_char3_sat(ulong3);
09473 char3 const_func __attribute__((overloadable)) convert_char3_rte(float3);
09474 char3 const_func __attribute__((overloadable)) convert_char3_sat_rte(float3);
09475 char3 const_func __attribute__((overloadable)) convert_char3_rtz(float3);
09476 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtz(float3);
09477 char3 const_func __attribute__((overloadable)) convert_char3_rtp(float3);
09478 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtp(float3);
09479 char3 const_func __attribute__((overloadable)) convert_char3_rtn(float3);
09480 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtn(float3);
09481 char3 const_func __attribute__((overloadable)) convert_char3(float3);
09482 char3 const_func __attribute__((overloadable)) convert_char3_sat(float3);
09483 char3 const_func __attribute__((overloadable)) convert_char3_rte(double3);
09484 char3 const_func __attribute__((overloadable)) convert_char3_sat_rte(double3);
09485 char3 const_func __attribute__((overloadable)) convert_char3_rtz(double3);
09486 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtz(double3);
09487 char3 const_func __attribute__((overloadable)) convert_char3_rtp(double3);
09488 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtp(double3);
09489 char3 const_func __attribute__((overloadable)) convert_char3_rtn(double3);
09490 char3 const_func __attribute__((overloadable)) convert_char3_sat_rtn(double3);
09491 char3 const_func __attribute__((overloadable)) convert_char3(double3);
09492 char3 const_func __attribute__((overloadable)) convert_char3_sat(double3);
09493
09494 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rte(char3);
09495 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rte(char3);
```

```
09496 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtz(char3);
09497 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtz(char3);
09498 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtp(char3);
09499 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtp(char3);
09500 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtn(char3);
09501 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtn(char3);
09502 uchar3 const_func __attribute__((overloadable)) convert_uchar3(char3);
09503 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat(char3);
09504 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rte(uchar3);
09505 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rte(uchar3);
09506 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtz(uchar3);
09507 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtz(uchar3);
09508 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtp(uchar3);
09509 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtp(uchar3);
09510 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtn(uchar3);
09511 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtn(uchar3);
09512 uchar3 const_func __attribute__((overloadable)) convert_uchar3(uchar3);
09513 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat(uchar3);
09514 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rte(short3);
09515 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rte(short3);
09516 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtz(short3);
09517 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtz(short3);
09518 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtp(short3);
09519 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtp(short3);
09520 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtn(short3);
09521 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtn(short3);
09522 uchar3 const_func __attribute__((overloadable)) convert_uchar3(short3);
09523 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat(short3);
09524 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rte(ushort3);
09525 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rte(ushort3);
09526 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtz(ushort3);
09527 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtz(ushort3);
09528 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtp(ushort3);
09529 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtp(ushort3);
09530 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtn(ushort3);
09531 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtn(ushort3);
09532 uchar3 const_func __attribute__((overloadable)) convert_uchar3(ushort3);
09533 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat(ushort3);
09534 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rte(int3);
09535 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rte(int3);
09536 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtz(int3);
09537 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtz(int3);
09538 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtp(int3);
09539 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtp(int3);
09540 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtn(int3);
09541 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtn(int3);
09542 uchar3 const_func __attribute__((overloadable)) convert_uchar3(int3);
09543 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat(int3);
09544 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rte(uint3);
09545 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rte(uint3);
09546 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtz(uint3);
09547 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtz(uint3);
09548 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtp(uint3);
09549 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtp(uint3);
09550 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtn(uint3);
09551 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtn(uint3);
09552 uchar3 const_func __attribute__((overloadable)) convert_uchar3(uint3);
09553 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat(uint3);
09554 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rte(long3);
09555 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rte(long3);
09556 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtz(long3);
09557 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtz(long3);
09558 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtp(long3);
09559 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtp(long3);
09560 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtn(long3);
09561 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtn(long3);
09562 uchar3 const_func __attribute__((overloadable)) convert_uchar3(long3);
09563 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat(long3);
09564 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rte(ulong3);
09565 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rte(ulong3);
09566 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtz(ulong3);
09567 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtz(ulong3);
09568 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtp(ulong3);
09569 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtp(ulong3);
09570 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtn(ulong3);
09571 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtn(ulong3);
09572 uchar3 const_func __attribute__((overloadable)) convert_uchar3(ulong3);
09573 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat(ulong3);
09574 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rte(float3);
09575 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rte(float3);
09576 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtz(float3);
09577 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtz(float3);
09578 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtp(float3);
09579 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtp(float3);
09580 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtn(float3);
09581 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtn(float3);
09582 uchar3 const_func __attribute__((overloadable)) convert_uchar3(float3);
```

```
09583 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat(float3);
09584 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rte(double3);
09585 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rte(double3);
09586 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtz(double3);
09587 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtz(double3);
09588 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtp(double3);
09589 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtp(double3);
09590 uchar3 const_func __attribute__((overloadable)) convert_uchar3_rtn(double3);
09591 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat_rtn(double3);
09592 uchar3 const_func __attribute__((overloadable)) convert_uchar3(double3);
09593 uchar3 const_func __attribute__((overloadable)) convert_uchar3_sat(double3);
09594
09595 short3 const_func __attribute__((overloadable)) convert_short3_rte(char3);
09596 short3 const_func __attribute__((overloadable)) convert_short3_sat_rte(char3);
09597 short3 const_func __attribute__((overloadable)) convert_short3_rtz(char3);
09598 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtz(char3);
09599 short3 const_func __attribute__((overloadable)) convert_short3_rtp(char3);
09600 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtp(char3);
09601 short3 const_func __attribute__((overloadable)) convert_short3_rtn(char3);
09602 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtn(char3);
09603 short3 const_func __attribute__((overloadable)) convert_short3(char3);
09604 short3 const_func __attribute__((overloadable)) convert_short3_sat(char3);
09605 short3 const_func __attribute__((overloadable)) convert_short3_rte(uchar3);
09606 short3 const_func __attribute__((overloadable)) convert_short3_sat_rte(uchar3);
09607 short3 const_func __attribute__((overloadable)) convert_short3_rtz(uchar3);
09608 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtz(uchar3);
09609 short3 const_func __attribute__((overloadable)) convert_short3_rtp(uchar3);
09610 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtp(uchar3);
09611 short3 const_func __attribute__((overloadable)) convert_short3_rtn(uchar3);
09612 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtn(uchar3);
09613 short3 const_func __attribute__((overloadable)) convert_short3(uchar3);
09614 short3 const_func __attribute__((overloadable)) convert_short3_sat(uchar3);
09615 short3 const_func __attribute__((overloadable)) convert_short3_rte(short3);
09616 short3 const_func __attribute__((overloadable)) convert_short3_sat_rte(short3);
09617 short3 const_func __attribute__((overloadable)) convert_short3_rtz(short3);
09618 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtz(short3);
09619 short3 const_func __attribute__((overloadable)) convert_short3_rtp(short3);
09620 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtp(short3);
09621 short3 const_func __attribute__((overloadable)) convert_short3_rtn(short3);
09622 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtn(short3);
09623 short3 const_func __attribute__((overloadable)) convert_short3(short3);
09624 short3 const_func __attribute__((overloadable)) convert_short3_sat(short3);
09625 short3 const_func __attribute__((overloadable)) convert_short3_rte(ushort3);
09626 short3 const_func __attribute__((overloadable)) convert_short3_sat_rte(ushort3);
09627 short3 const_func __attribute__((overloadable)) convert_short3_rtz(ushort3);
09628 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtz(ushort3);
09629 short3 const_func __attribute__((overloadable)) convert_short3_rtp(ushort3);
09630 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtp(ushort3);
09631 short3 const_func __attribute__((overloadable)) convert_short3_rtn(ushort3);
09632 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtn(ushort3);
09633 short3 const_func __attribute__((overloadable)) convert_short3(ushort3);
09634 short3 const_func __attribute__((overloadable)) convert_short3_sat(ushort3);
09635 short3 const_func __attribute__((overloadable)) convert_short3_rte(int3);
09636 short3 const_func __attribute__((overloadable)) convert_short3_sat_rte(int3);
09637 short3 const_func __attribute__((overloadable)) convert_short3_rtz(int3);
09638 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtz(int3);
09639 short3 const_func __attribute__((overloadable)) convert_short3_rtp(int3);
09640 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtp(int3);
09641 short3 const_func __attribute__((overloadable)) convert_short3_rtn(int3);
09642 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtn(int3);
09643 short3 const_func __attribute__((overloadable)) convert_short3(int3);
09644 short3 const_func __attribute__((overloadable)) convert_short3_sat(int3);
09645 short3 const_func __attribute__((overloadable)) convert_short3_rte(uint3);
09646 short3 const_func __attribute__((overloadable)) convert_short3_sat_rte(uint3);
09647 short3 const_func __attribute__((overloadable)) convert_short3_rtz(uint3);
09648 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtz(uint3);
09649 short3 const_func __attribute__((overloadable)) convert_short3_rtp(uint3);
09650 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtp(uint3);
09651 short3 const_func __attribute__((overloadable)) convert_short3_rtn(uint3);
09652 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtn(uint3);
09653 short3 const_func __attribute__((overloadable)) convert_short3(uint3);
09654 short3 const_func __attribute__((overloadable)) convert_short3_sat(uint3);
09655 short3 const_func __attribute__((overloadable)) convert_short3_rte(long3);
09656 short3 const_func __attribute__((overloadable)) convert_short3_sat_rte(long3);
09657 short3 const_func __attribute__((overloadable)) convert_short3_rtz(long3);
09658 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtz(long3);
09659 short3 const_func __attribute__((overloadable)) convert_short3_rtp(long3);
09660 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtp(long3);
09661 short3 const_func __attribute__((overloadable)) convert_short3_rtn(long3);
09662 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtn(long3);
09663 short3 const_func __attribute__((overloadable)) convert_short3(long3);
09664 short3 const_func __attribute__((overloadable)) convert_short3_sat(long3);
09665 short3 const_func __attribute__((overloadable)) convert_short3_rte(ulong3);
09666 short3 const_func __attribute__((overloadable)) convert_short3_sat_rte(ulong3);
09667 short3 const_func __attribute__((overloadable)) convert_short3_rtz(ulong3);
09668 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtz(ulong3);
09669 short3 const_func __attribute__((overloadable)) convert_short3_rtp(ulong3);
```

```
09670 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtp(ulong3);
09671 short3 const_func __attribute__((overloadable)) convert_short3_rtn(ulong3);
09672 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtn(ulong3);
09673 short3 const_func __attribute__((overloadable)) convert_short3(ulong3);
09674 short3 const_func __attribute__((overloadable)) convert_short3_sat(ulong3);
09675 short3 const_func __attribute__((overloadable)) convert_short3_rte(float3);
09676 short3 const_func __attribute__((overloadable)) convert_short3_sat_rte(float3);
09677 short3 const_func __attribute__((overloadable)) convert_short3_rtz(float3);
09678 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtz(float3);
09679 short3 const_func __attribute__((overloadable)) convert_short3_rtp(float3);
09680 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtp(float3);
09681 short3 const_func __attribute__((overloadable)) convert_short3_rtn(float3);
09682 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtn(float3);
09683 short3 const_func __attribute__((overloadable)) convert_short3(float3);
09684 short3 const_func __attribute__((overloadable)) convert_short3_sat(float3);
09685 short3 const_func __attribute__((overloadable)) convert_short3_rte(double3);
09686 short3 const_func __attribute__((overloadable)) convert_short3_sat_rte(double3);
09687 short3 const_func __attribute__((overloadable)) convert_short3_rtz(double3);
09688 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtz(double3);
09689 short3 const_func __attribute__((overloadable)) convert_short3_rtp(double3);
09690 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtp(double3);
09691 short3 const_func __attribute__((overloadable)) convert_short3_rtn(double3);
09692 short3 const_func __attribute__((overloadable)) convert_short3_sat_rtn(double3);
09693 short3 const_func __attribute__((overloadable)) convert_short3(double3);
09694 short3 const_func __attribute__((overloadable)) convert_short3_sat(double3);
09695
09696 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rte(char3);
09697 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rte(char3);
09698 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtz(char3);
09699 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtz(char3);
09700 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtp(char3);
09701 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtp(char3);
09702 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtn(char3);
09703 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtn(char3);
09704 ushort3 const_func __attribute__((overloadable)) convert_ushort3(char3);
09705 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat(char3);
09706 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rte(uchar3);
09707 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rte(uchar3);
09708 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtz(uchar3);
09709 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtz(uchar3);
09710 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtp(uchar3);
09711 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtp(uchar3);
09712 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtn(uchar3);
09713 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtn(uchar3);
09714 ushort3 const_func __attribute__((overloadable)) convert_ushort3(uchar3);
09715 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat(uchar3);
09716 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rte(short3);
09717 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rte(short3);
09718 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtz(short3);
09719 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtz(short3);
09720 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtp(short3);
09721 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtp(short3);
09722 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtn(short3);
09723 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtn(short3);
09724 ushort3 const_func __attribute__((overloadable)) convert_ushort3(short3);
09725 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat(short3);
09726 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rte(ushort3);
09727 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rte(ushort3);
09728 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtz(ushort3);
09729 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtz(ushort3);
09730 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtp(ushort3);
09731 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtp(ushort3);
09732 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtn(ushort3);
09733 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtn(ushort3);
09734 ushort3 const_func __attribute__((overloadable)) convert_ushort3(ushort3);
09735 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat(ushort3);
09736 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rte(int3);
09737 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rte(int3);
09738 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtz(int3);
09739 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtz(int3);
09740 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtp(int3);
09741 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtp(int3);
09742 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtn(int3);
09743 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtn(int3);
09744 ushort3 const_func __attribute__((overloadable)) convert_ushort3(int3);
09745 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat(int3);
09746 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rte(uint3);
09747 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rte(uint3);
09748 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtz(uint3);
09749 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtz(uint3);
09750 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtp(uint3);
09751 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtp(uint3);
09752 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtn(uint3);
09753 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtn(uint3);
09754 ushort3 const_func __attribute__((overloadable)) convert_ushort3(uint3);
09755 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat(uint3);
09756 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rte(long3);
```

```
09757 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rte(long3);
09758 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtz(long3);
09759 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtz(long3);
09760 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtp(long3);
09761 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtp(long3);
09762 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtn(long3);
09763 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtn(long3);
09764 ushort3 const_func __attribute__((overloadable)) convert_ushort3(long3);
09765 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat(long3);
09766 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rte(ulong3);
09767 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rte(ulong3);
09768 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtz(ulong3);
09769 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtz(ulong3);
09770 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtp(ulong3);
09771 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtp(ulong3);
09772 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtn(ulong3);
09773 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtn(ulong3);
09774 ushort3 const_func __attribute__((overloadable)) convert_ushort3(ulong3);
09775 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat(ulong3);
09776 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rte(float3);
09777 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rte(float3);
09778 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtz(float3);
09779 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtz(float3);
09780 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtp(float3);
09781 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtp(float3);
09782 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtn(float3);
09783 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtn(float3);
09784 ushort3 const_func __attribute__((overloadable)) convert_ushort3(float3);
09785 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat(float3);
09786 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rte(double3);
09787 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rte(double3);
09788 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtz(double3);
09789 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtz(double3);
09790 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtp(double3);
09791 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtp(double3);
09792 ushort3 const_func __attribute__((overloadable)) convert_ushort3_rtn(double3);
09793 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat_rtn(double3);
09794 ushort3 const_func __attribute__((overloadable)) convert_ushort3(double3);
09795 ushort3 const_func __attribute__((overloadable)) convert_ushort3_sat(double3);
09796
09797 int3 const_func __attribute__((overloadable)) convert_int3_rte(char3);
09798 int3 const_func __attribute__((overloadable)) convert_int3_sat_rte(char3);
09799 int3 const_func __attribute__((overloadable)) convert_int3_rtz(char3);
09800 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtz(char3);
09801 int3 const_func __attribute__((overloadable)) convert_int3_rtp(char3);
09802 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtp(char3);
09803 int3 const_func __attribute__((overloadable)) convert_int3_rtn(char3);
09804 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtn(char3);
09805 int3 const_func __attribute__((overloadable)) convert_int3(char3);
09806 int3 const_func __attribute__((overloadable)) convert_int3_sat(char3);
09807 int3 const_func __attribute__((overloadable)) convert_int3_rte(uchar3);
09808 int3 const_func __attribute__((overloadable)) convert_int3_sat_rte(uchar3);
09809 int3 const_func __attribute__((overloadable)) convert_int3_rtz(uchar3);
09810 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtz(uchar3);
09811 int3 const_func __attribute__((overloadable)) convert_int3_rtp(uchar3);
09812 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtp(uchar3);
09813 int3 const_func __attribute__((overloadable)) convert_int3_rtn(uchar3);
09814 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtn(uchar3);
09815 int3 const_func __attribute__((overloadable)) convert_int3(uchar3);
09816 int3 const_func __attribute__((overloadable)) convert_int3_sat(uchar3);
09817 int3 const_func __attribute__((overloadable)) convert_int3_rte(short3);
09818 int3 const_func __attribute__((overloadable)) convert_int3_sat_rte(short3);
09819 int3 const_func __attribute__((overloadable)) convert_int3_rtz(short3);
09820 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtz(short3);
09821 int3 const_func __attribute__((overloadable)) convert_int3_rtp(short3);
09822 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtp(short3);
09823 int3 const_func __attribute__((overloadable)) convert_int3_rtn(short3);
09824 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtn(short3);
09825 int3 const_func __attribute__((overloadable)) convert_int3(short3);
09826 int3 const_func __attribute__((overloadable)) convert_int3_sat(short3);
09827 int3 const_func __attribute__((overloadable)) convert_int3_rte(ushort3);
09828 int3 const_func __attribute__((overloadable)) convert_int3_sat_rte(ushort3);
09829 int3 const_func __attribute__((overloadable)) convert_int3_rtz(ushort3);
09830 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtz(ushort3);
09831 int3 const_func __attribute__((overloadable)) convert_int3_rtp(ushort3);
09832 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtp(ushort3);
09833 int3 const_func __attribute__((overloadable)) convert_int3_rtn(ushort3);
09834 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtn(ushort3);
09835 int3 const_func __attribute__((overloadable)) convert_int3(ushort3);
09836 int3 const_func __attribute__((overloadable)) convert_int3_sat(ushort3);
09837 int3 const_func __attribute__((overloadable)) convert_int3_rte(int3);
09838 int3 const_func __attribute__((overloadable)) convert_int3_sat_rte(int3);
09839 int3 const_func __attribute__((overloadable)) convert_int3_rtz(int3);
09840 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtz(int3);
09841 int3 const_func __attribute__((overloadable)) convert_int3_rtp(int3);
09842 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtp(int3);
09843 int3 const_func __attribute__((overloadable)) convert_int3_rtn(int3);
```

```
09844 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtn(int3);
09845 int3 const_func __attribute__((overloadable)) convert_int3(int3);
09846 int3 const_func __attribute__((overloadable)) convert_int3_sat(int3);
09847 int3 const_func __attribute__((overloadable)) convert_int3_rte(uint3);
09848 int3 const_func __attribute__((overloadable)) convert_int3_sat_rte(uint3);
09849 int3 const_func __attribute__((overloadable)) convert_int3_rtz(uint3);
09850 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtz(uint3);
09851 int3 const_func __attribute__((overloadable)) convert_int3_rtp(uint3);
09852 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtp(uint3);
09853 int3 const_func __attribute__((overloadable)) convert_int3_rtn(uint3);
09854 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtn(uint3);
09855 int3 const_func __attribute__((overloadable)) convert_int3(uint3);
09856 int3 const_func __attribute__((overloadable)) convert_int3_sat(uint3);
09857 int3 const_func __attribute__((overloadable)) convert_int3_rte(long3);
09858 int3 const_func __attribute__((overloadable)) convert_int3_sat_rte(long3);
09859 int3 const_func __attribute__((overloadable)) convert_int3_rtz(long3);
09860 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtz(long3);
09861 int3 const_func __attribute__((overloadable)) convert_int3_rtp(long3);
09862 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtp(long3);
09863 int3 const_func __attribute__((overloadable)) convert_int3_rtn(long3);
09864 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtn(long3);
09865 int3 const_func __attribute__((overloadable)) convert_int3(long3);
09866 int3 const_func __attribute__((overloadable)) convert_int3_sat(long3);
09867 int3 const_func __attribute__((overloadable)) convert_int3_rte(ulong3);
09868 int3 const_func __attribute__((overloadable)) convert_int3_sat_rte(ulong3);
09869 int3 const_func __attribute__((overloadable)) convert_int3_rtz(ulong3);
09870 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtz(ulong3);
09871 int3 const_func __attribute__((overloadable)) convert_int3_rtp(ulong3);
09872 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtp(ulong3);
09873 int3 const_func __attribute__((overloadable)) convert_int3_rtn(ulong3);
09874 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtn(ulong3);
09875 int3 const_func __attribute__((overloadable)) convert_int3(ulong3);
09876 int3 const_func __attribute__((overloadable)) convert_int3_sat(ulong3);
09877 int3 const_func __attribute__((overloadable)) convert_int3_rte(float3);
09878 int3 const_func __attribute__((overloadable)) convert_int3_sat_rte(float3);
09879 int3 const_func __attribute__((overloadable)) convert_int3_rtz(float3);
09880 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtz(float3);
09881 int3 const_func __attribute__((overloadable)) convert_int3_rtp(float3);
09882 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtp(float3);
09883 int3 const_func __attribute__((overloadable)) convert_int3_rtn(float3);
09884 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtn(float3);
09885 int3 const_func __attribute__((overloadable)) convert_int3(float3);
09886 int3 const_func __attribute__((overloadable)) convert_int3_sat(float3);
09887 int3 const_func __attribute__((overloadable)) convert_int3_rte(double3);
09888 int3 const_func __attribute__((overloadable)) convert_int3_sat_rte(double3);
09889 int3 const_func __attribute__((overloadable)) convert_int3_rtz(double3);
09890 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtz(double3);
09891 int3 const_func __attribute__((overloadable)) convert_int3_rtp(double3);
09892 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtp(double3);
09893 int3 const_func __attribute__((overloadable)) convert_int3_rtn(double3);
09894 int3 const_func __attribute__((overloadable)) convert_int3_sat_rtn(double3);
09895 int3 const_func __attribute__((overloadable)) convert_int3(double3);
09896 int3 const_func __attribute__((overloadable)) convert_int3_sat(double3);
09897
09898 uint3 const_func __attribute__((overloadable)) convert_uint3_rte(char3);
09899 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rte(char3);
09900 uint3 const_func __attribute__((overloadable)) convert_uint3_rtz(char3);
09901 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtz(char3);
09902 uint3 const_func __attribute__((overloadable)) convert_uint3_rtp(char3);
09903 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtp(char3);
09904 uint3 const_func __attribute__((overloadable)) convert_uint3_rtn(char3);
09905 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtn(char3);
09906 uint3 const_func __attribute__((overloadable)) convert_uint3(char3);
09907 uint3 const_func __attribute__((overloadable)) convert_uint3_sat(char3);
09908 uint3 const_func __attribute__((overloadable)) convert_uint3_rte(uchar3);
09909 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rte(uchar3);
09910 uint3 const_func __attribute__((overloadable)) convert_uint3_rtz(uchar3);
09911 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtz(uchar3);
09912 uint3 const_func __attribute__((overloadable)) convert_uint3_rtp(uchar3);
09913 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtp(uchar3);
09914 uint3 const_func __attribute__((overloadable)) convert_uint3_rtn(uchar3);
09915 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtn(uchar3);
09916 uint3 const_func __attribute__((overloadable)) convert_uint3(uchar3);
09917 uint3 const_func __attribute__((overloadable)) convert_uint3_sat(uchar3);
09918 uint3 const_func __attribute__((overloadable)) convert_uint3_rte(short3);
09919 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rte(short3);
09920 uint3 const_func __attribute__((overloadable)) convert_uint3_rtz(short3);
09921 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtz(short3);
09922 uint3 const_func __attribute__((overloadable)) convert_uint3_rtp(short3);
09923 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtp(short3);
09924 uint3 const_func __attribute__((overloadable)) convert_uint3_rtn(short3);
09925 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtn(short3);
09926 uint3 const_func __attribute__((overloadable)) convert_uint3(short3);
09927 uint3 const_func __attribute__((overloadable)) convert_uint3_sat(short3);
09928 uint3 const_func __attribute__((overloadable)) convert_uint3_rte(ushort3);
09929 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rte(ushort3);
09930 uint3 const_func __attribute__((overloadable)) convert_uint3_rtz(ushort3);
```

```
09931 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtz(ushort3);
09932 uint3 const_func __attribute__((overloadable)) convert_uint3_rtp(ushort3);
09933 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtp(ushort3);
09934 uint3 const_func __attribute__((overloadable)) convert_uint3_rtn(ushort3);
09935 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtn(ushort3);
09936 uint3 const_func __attribute__((overloadable)) convert_uint3(ushort3);
09937 uint3 const_func __attribute__((overloadable)) convert_uint3_sat(ushort3);
09938 uint3 const_func __attribute__((overloadable)) convert_uint3_rte(int3);
09939 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rte(int3);
09940 uint3 const_func __attribute__((overloadable)) convert_uint3_rtz(int3);
09941 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtz(int3);
09942 uint3 const_func __attribute__((overloadable)) convert_uint3_rtp(int3);
09943 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtp(int3);
09944 uint3 const_func __attribute__((overloadable)) convert_uint3_rtn(int3);
09945 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtn(int3);
09946 uint3 const_func __attribute__((overloadable)) convert_uint3(int3);
09947 uint3 const_func __attribute__((overloadable)) convert_uint3_sat(int3);
09948 uint3 const_func __attribute__((overloadable)) convert_uint3_rte(uint3);
09949 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rte(uint3);
09950 uint3 const_func __attribute__((overloadable)) convert_uint3_rtz(uint3);
09951 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtz(uint3);
09952 uint3 const_func __attribute__((overloadable)) convert_uint3_rtp(uint3);
09953 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtp(uint3);
09954 uint3 const_func __attribute__((overloadable)) convert_uint3_rtn(uint3);
09955 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtn(uint3);
09956 uint3 const_func __attribute__((overloadable)) convert_uint3(uint3);
09957 uint3 const_func __attribute__((overloadable)) convert_uint3_sat(uint3);
09958 uint3 const_func __attribute__((overloadable)) convert_uint3_rte(long3);
09959 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rte(long3);
09960 uint3 const_func __attribute__((overloadable)) convert_uint3_rtz(long3);
09961 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtz(long3);
09962 uint3 const_func __attribute__((overloadable)) convert_uint3_rtp(long3);
09963 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtp(long3);
09964 uint3 const_func __attribute__((overloadable)) convert_uint3_rtn(long3);
09965 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtn(long3);
09966 uint3 const_func __attribute__((overloadable)) convert_uint3(long3);
09967 uint3 const_func __attribute__((overloadable)) convert_uint3_sat(long3);
09968 uint3 const_func __attribute__((overloadable)) convert_uint3_rte(ulong3);
09969 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rte(ulong3);
09970 uint3 const_func __attribute__((overloadable)) convert_uint3_rtz(ulong3);
09971 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtz(ulong3);
09972 uint3 const_func __attribute__((overloadable)) convert_uint3_rtp(ulong3);
09973 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtp(ulong3);
09974 uint3 const_func __attribute__((overloadable)) convert_uint3_rtn(ulong3);
09975 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtn(ulong3);
09976 uint3 const_func __attribute__((overloadable)) convert_uint3(ulong3);
09977 uint3 const_func __attribute__((overloadable)) convert_uint3_sat(ulong3);
09978 uint3 const_func __attribute__((overloadable)) convert_uint3_rte(float3);
09979 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rte(float3);
09980 uint3 const_func __attribute__((overloadable)) convert_uint3_rtz(float3);
09981 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtz(float3);
09982 uint3 const_func __attribute__((overloadable)) convert_uint3_rtp(float3);
09983 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtp(float3);
09984 uint3 const_func __attribute__((overloadable)) convert_uint3_rtn(float3);
09985 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtn(float3);
09986 uint3 const_func __attribute__((overloadable)) convert_uint3(float3);
09987 uint3 const_func __attribute__((overloadable)) convert_uint3_sat(float3);
09988 uint3 const_func __attribute__((overloadable)) convert_uint3_rte(double3);
09989 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rte(double3);
09990 uint3 const_func __attribute__((overloadable)) convert_uint3_rtz(double3);
09991 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtz(double3);
09992 uint3 const_func __attribute__((overloadable)) convert_uint3_rtp(double3);
09993 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtp(double3);
09994 uint3 const_func __attribute__((overloadable)) convert_uint3_rtn(double3);
09995 uint3 const_func __attribute__((overloadable)) convert_uint3_sat_rtn(double3);
09996 uint3 const_func __attribute__((overloadable)) convert_uint3(double3);
09997 uint3 const_func __attribute__((overloadable)) convert_uint3_sat(double3);
09998 long3 const_func __attribute__((overloadable)) convert_long3_rte(char3);
09999 long3 const_func __attribute__((overloadable)) convert_long3_sat_rte(char3);
10000 long3 const_func __attribute__((overloadable)) convert_long3_rtz(char3);
10001 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtz(char3);
10002 long3 const_func __attribute__((overloadable)) convert_long3_rtp(char3);
10003 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtp(char3);
10004 long3 const_func __attribute__((overloadable)) convert_long3_rtn(char3);
10005 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtn(char3);
10006 long3 const_func __attribute__((overloadable)) convert_long3(char3);
10007 long3 const_func __attribute__((overloadable)) convert_long3_sat(char3);
10008 long3 const_func __attribute__((overloadable)) convert_long3_rte(uchar3);
10009 long3 const_func __attribute__((overloadable)) convert_long3_sat_rte(uchar3);
10010 long3 const_func __attribute__((overloadable)) convert_long3_rtz(uchar3);
10011 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtz(uchar3);
10012 long3 const_func __attribute__((overloadable)) convert_long3_rtp(uchar3);
10013 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtp(uchar3);
10014 long3 const_func __attribute__((overloadable)) convert_long3_rtn(uchar3);
10015 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtn(uchar3);
10016 long3 const_func __attribute__((overloadable)) convert_long3(uchar3);
10017 long3 const_func __attribute__((overloadable)) convert_long3_sat(uchar3);
```

```
10018 long3 const_func __attribute__((overloadable)) convert_long3_rte(short3);
10019 long3 const_func __attribute__((overloadable)) convert_long3_sat_rte(short3);
10020 long3 const_func __attribute__((overloadable)) convert_long3_rtz(short3);
10021 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtz(short3);
10022 long3 const_func __attribute__((overloadable)) convert_long3_rtp(short3);
10023 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtp(short3);
10024 long3 const_func __attribute__((overloadable)) convert_long3_rtn(short3);
10025 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtn(short3);
10026 long3 const_func __attribute__((overloadable)) convert_long3(short3);
10027 long3 const_func __attribute__((overloadable)) convert_long3_sat(short3);
10028 long3 const_func __attribute__((overloadable)) convert_long3_rte(ushort3);
10029 long3 const_func __attribute__((overloadable)) convert_long3_sat_rte(ushort3);
10030 long3 const_func __attribute__((overloadable)) convert_long3_rtz(ushort3);
10031 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtz(ushort3);
10032 long3 const_func __attribute__((overloadable)) convert_long3_rtp(ushort3);
10033 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtp(ushort3);
10034 long3 const_func __attribute__((overloadable)) convert_long3_rtn(ushort3);
10035 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtn(ushort3);
10036 long3 const_func __attribute__((overloadable)) convert_long3(ushort3);
10037 long3 const_func __attribute__((overloadable)) convert_long3_sat(ushort3);
10038 long3 const_func __attribute__((overloadable)) convert_long3_rte(int3);
10039 long3 const_func __attribute__((overloadable)) convert_long3_sat_rte(int3);
10040 long3 const_func __attribute__((overloadable)) convert_long3_rtz(int3);
10041 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtz(int3);
10042 long3 const_func __attribute__((overloadable)) convert_long3_rtp(int3);
10043 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtp(int3);
10044 long3 const_func __attribute__((overloadable)) convert_long3_rtn(int3);
10045 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtn(int3);
10046 long3 const_func __attribute__((overloadable)) convert_long3(int3);
10047 long3 const_func __attribute__((overloadable)) convert_long3_sat(int3);
10048 long3 const_func __attribute__((overloadable)) convert_long3_rte(uint3);
10049 long3 const_func __attribute__((overloadable)) convert_long3_sat_rte(uint3);
10050 long3 const_func __attribute__((overloadable)) convert_long3_rtz(uint3);
10051 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtz(uint3);
10052 long3 const_func __attribute__((overloadable)) convert_long3_rtp(uint3);
10053 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtp(uint3);
10054 long3 const_func __attribute__((overloadable)) convert_long3_rtn(uint3);
10055 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtn(uint3);
10056 long3 const_func __attribute__((overloadable)) convert_long3(uint3);
10057 long3 const_func __attribute__((overloadable)) convert_long3_sat(uint3);
10058 long3 const_func __attribute__((overloadable)) convert_long3_rte(long3);
10059 long3 const_func __attribute__((overloadable)) convert_long3_sat_rte(long3);
10060 long3 const_func __attribute__((overloadable)) convert_long3_rtz(long3);
10061 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtz(long3);
10062 long3 const_func __attribute__((overloadable)) convert_long3_rtp(long3);
10063 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtp(long3);
10064 long3 const_func __attribute__((overloadable)) convert_long3_rtn(long3);
10065 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtn(long3);
10066 long3 const_func __attribute__((overloadable)) convert_long3(long3);
10067 long3 const_func __attribute__((overloadable)) convert_long3_sat(long3);
10068 long3 const_func __attribute__((overloadable)) convert_long3_rte(ulong3);
10069 long3 const_func __attribute__((overloadable)) convert_long3_sat_rte(ulong3);
10070 long3 const_func __attribute__((overloadable)) convert_long3_rtz(ulong3);
10071 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtz(ulong3);
10072 long3 const_func __attribute__((overloadable)) convert_long3_rtp(ulong3);
10073 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtp(ulong3);
10074 long3 const_func __attribute__((overloadable)) convert_long3_rtn(ulong3);
10075 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtn(ulong3);
10076 long3 const_func __attribute__((overloadable)) convert_long3(ulong3);
10077 long3 const_func __attribute__((overloadable)) convert_long3_sat(ulong3);
10078 long3 const_func __attribute__((overloadable)) convert_long3_rte(float3);
10079 long3 const_func __attribute__((overloadable)) convert_long3_sat_rte(float3);
10080 long3 const_func __attribute__((overloadable)) convert_long3_rtz(float3);
10081 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtz(float3);
10082 long3 const_func __attribute__((overloadable)) convert_long3_rtp(float3);
10083 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtp(float3);
10084 long3 const_func __attribute__((overloadable)) convert_long3_rtn(float3);
10085 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtn(float3);
10086 long3 const_func __attribute__((overloadable)) convert_long3(float3);
10087 long3 const_func __attribute__((overloadable)) convert_long3_sat(float3);
10088 long3 const_func __attribute__((overloadable)) convert_long3_rte(double3);
10089 long3 const_func __attribute__((overloadable)) convert_long3_sat_rte(double3);
10090 long3 const_func __attribute__((overloadable)) convert_long3_rtz(double3);
10091 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtz(double3);
10092 long3 const_func __attribute__((overloadable)) convert_long3_rtp(double3);
10093 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtp(double3);
10094 long3 const_func __attribute__((overloadable)) convert_long3_rtn(double3);
10095 long3 const_func __attribute__((overloadable)) convert_long3_sat_rtn(double3);
10096 long3 const_func __attribute__((overloadable)) convert_long3(double3);
10097 long3 const_func __attribute__((overloadable)) convert_long3_sat(double3);
10098 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rte(char3);
10099 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rte(char3);
10100 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtz(char3);
10101 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtz(char3);
10102 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtp(char3);
10103 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtp(char3);
10104 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtn(char3);
```

```
10105 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtn(char3);
10106 ulong3 const_func __attribute__((overloadable)) convert_ulong3(char3);
10107 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat(char3);
10108 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rte(uchar3);
10109 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rte(uchar3);
10110 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtz(uchar3);
10111 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtz(uchar3);
10112 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtp(uchar3);
10113 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtp(uchar3);
10114 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtn(uchar3);
10115 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtn(uchar3);
10116 ulong3 const_func __attribute__((overloadable)) convert_ulong3(uchar3);
10117 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat(uchar3);
10118 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rte(short3);
10119 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rte(short3);
10120 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtz(short3);
10121 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtz(short3);
10122 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtp(short3);
10123 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtp(short3);
10124 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtn(short3);
10125 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtn(short3);
10126 ulong3 const_func __attribute__((overloadable)) convert_ulong3(short3);
10127 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat(short3);
10128 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rte(ushort3);
10129 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rte(ushort3);
10130 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtz(ushort3);
10131 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtz(ushort3);
10132 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtp(ushort3);
10133 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtp(ushort3);
10134 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtn(ushort3);
10135 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtn(ushort3);
10136 ulong3 const_func __attribute__((overloadable)) convert_ulong3(ushort3);
10137 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat(ushort3);
10138 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rte(int3);
10139 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rte(int3);
10140 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtz(int3);
10141 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtz(int3);
10142 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtp(int3);
10143 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtp(int3);
10144 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtn(int3);
10145 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtn(int3);
10146 ulong3 const_func __attribute__((overloadable)) convert_ulong3(int3);
10147 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat(int3);
10148 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rte(uint3);
10149 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rte(uint3);
10150 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtz(uint3);
10151 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtz(uint3);
10152 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtp(uint3);
10153 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtp(uint3);
10154 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtn(uint3);
10155 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtn(uint3);
10156 ulong3 const_func __attribute__((overloadable)) convert_ulong3(uint3);
10157 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat(uint3);
10158 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rte(long3);
10159 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rte(long3);
10160 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtz(long3);
10161 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtz(long3);
10162 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtp(long3);
10163 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtp(long3);
10164 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtn(long3);
10165 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtn(long3);
10166 ulong3 const_func __attribute__((overloadable)) convert_ulong3(long3);
10167 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat(long3);
10168 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rte(ulong3);
10169 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rte(ulong3);
10170 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtz(ulong3);
10171 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtz(ulong3);
10172 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtp(ulong3);
10173 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtp(ulong3);
10174 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtn(ulong3);
10175 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtn(ulong3);
10176 ulong3 const_func __attribute__((overloadable)) convert_ulong3(ulong3);
10177 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat(ulong3);
10178 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rte(float3);
10179 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rte(float3);
10180 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtz(float3);
10181 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtz(float3);
10182 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtp(float3);
10183 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtp(float3);
10184 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtn(float3);
10185 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtn(float3);
10186 ulong3 const_func __attribute__((overloadable)) convert_ulong3(float3);
10187 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat(float3);
10188 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rte(double3);
10189 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rte(double3);
10190 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtz(double3);
10191 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtz(double3);
```

```
10192 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtp(double3);
10193 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtp(double3);
10194 ulong3 const_func __attribute__((overloadable)) convert_ulong3_rtn(double3);
10195 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat_rtn(double3);
10196 ulong3 const_func __attribute__((overloadable)) convert_ulong3(double3);
10197 ulong3 const_func __attribute__((overloadable)) convert_ulong3_sat(double3);
10198 float3 const_func __attribute__((overloadable)) convert_float3_rte(char3);
10199 float3 const_func __attribute__((overloadable)) convert_float3_rtz(char3);
10200 float3 const_func __attribute__((overloadable)) convert_float3_rtp(char3);
10201 float3 const_func __attribute__((overloadable)) convert_float3_rtn(char3);
10202 float3 const_func __attribute__((overloadable)) convert_float3(char3);
10203 float3 const_func __attribute__((overloadable)) convert_float3_rte(uchar3);
10204 float3 const_func __attribute__((overloadable)) convert_float3_rtz(uchar3);
10205 float3 const_func __attribute__((overloadable)) convert_float3_rtp(uchar3);
10206 float3 const_func __attribute__((overloadable)) convert_float3_rtn(uchar3);
10207 float3 const_func __attribute__((overloadable)) convert_float3(uchar3);
10208 float3 const_func __attribute__((overloadable)) convert_float3_rte(short3);
10209 float3 const_func __attribute__((overloadable)) convert_float3_rtz(short3);
10210 float3 const_func __attribute__((overloadable)) convert_float3_rtp(short3);
10211 float3 const_func __attribute__((overloadable)) convert_float3_rtn(short3);
10212 float3 const_func __attribute__((overloadable)) convert_float3(short3);
10213 float3 const_func __attribute__((overloadable)) convert_float3_rte(ushort3);
10214 float3 const_func __attribute__((overloadable)) convert_float3_rtz(ushort3);
10215 float3 const_func __attribute__((overloadable)) convert_float3_rtp(ushort3);
10216 float3 const_func __attribute__((overloadable)) convert_float3_rtn(ushort3);
10217 float3 const_func __attribute__((overloadable)) convert_float3(ushort3);
10218 float3 const_func __attribute__((overloadable)) convert_float3_rte(int3);
10219 float3 const_func __attribute__((overloadable)) convert_float3_rtz(int3);
10220 float3 const_func __attribute__((overloadable)) convert_float3_rtp(int3);
10221 float3 const_func __attribute__((overloadable)) convert_float3_rtn(int3);
10222 float3 const_func __attribute__((overloadable)) convert_float3(int3);
10223 float3 const_func __attribute__((overloadable)) convert_float3_rte(uint3);
10224 float3 const_func __attribute__((overloadable)) convert_float3_rtz(uint3);
10225 float3 const_func __attribute__((overloadable)) convert_float3_rtp(uint3);
10226 float3 const_func __attribute__((overloadable)) convert_float3_rtn(uint3);
10227 float3 const_func __attribute__((overloadable)) convert_float3(uint3);
10228 float3 const_func __attribute__((overloadable)) convert_float3_rte(long3);
10229 float3 const_func __attribute__((overloadable)) convert_float3_rtz(long3);
10230 float3 const_func __attribute__((overloadable)) convert_float3_rtp(long3);
10231 float3 const_func __attribute__((overloadable)) convert_float3_rtn(long3);
10232 float3 const_func __attribute__((overloadable)) convert_float3(long3);
10233 float3 const_func __attribute__((overloadable)) convert_float3_rte(ulong3);
10234 float3 const_func __attribute__((overloadable)) convert_float3_rtz(ulong3);
10235 float3 const_func __attribute__((overloadable)) convert_float3_rtp(ulong3);
10236 float3 const_func __attribute__((overloadable)) convert_float3_rtn(ulong3);
10237 float3 const_func __attribute__((overloadable)) convert_float3(ulong3);
10238 float3 const_func __attribute__((overloadable)) convert_float3_rte(float3);
10239 float3 const_func __attribute__((overloadable)) convert_float3_rtz(float3);
10240 float3 const_func __attribute__((overloadable)) convert_float3_rtp(float3);
10241 float3 const_func __attribute__((overloadable)) convert_float3_rtn(float3);
10242 float3 const_func __attribute__((overloadable)) convert_float3(float3);
10243 float3 const_func __attribute__((overloadable)) convert_float3_rte(double3);
10244 float3 const_func __attribute__((overloadable)) convert_float3_rtz(double3);
10245 float3 const_func __attribute__((overloadable)) convert_float3_rtp(double3);
10246 float3 const_func __attribute__((overloadable)) convert_float3_rtn(double3);
10247 float3 const_func __attribute__((overloadable)) convert_float3(double3);
10248
10249 double3 const_func __attribute__((overloadable)) convert_double3_rte(char3);
10250 double3 const_func __attribute__((overloadable)) convert_double3_rtz(char3);
10251 double3 const_func __attribute__((overloadable)) convert_double3_rtp(char3);
10252 double3 const_func __attribute__((overloadable)) convert_double3_rtn(char3);
10253 double3 const_func __attribute__((overloadable)) convert_double3(char3);
10254 double3 const_func __attribute__((overloadable)) convert_double3_rte(uchar3);
10255 double3 const_func __attribute__((overloadable)) convert_double3_rtz(uchar3);
10256 double3 const_func __attribute__((overloadable)) convert_double3_rtp(uchar3);
10257 double3 const_func __attribute__((overloadable)) convert_double3_rtn(uchar3);
10258 double3 const_func __attribute__((overloadable)) convert_double3(uchar3);
10259 double3 const_func __attribute__((overloadable)) convert_double3_rte(short3);
10260 double3 const_func __attribute__((overloadable)) convert_double3_rtz(short3);
10261 double3 const_func __attribute__((overloadable)) convert_double3_rtp(short3);
10262 double3 const_func __attribute__((overloadable)) convert_double3_rtn(short3);
10263 double3 const_func __attribute__((overloadable)) convert_double3(short3);
10264 double3 const_func __attribute__((overloadable)) convert_double3_rte(ushort3);
10265 double3 const_func __attribute__((overloadable)) convert_double3_rtz(ushort3);
10266 double3 const_func __attribute__((overloadable)) convert_double3_rtp(ushort3);
10267 double3 const_func __attribute__((overloadable)) convert_double3_rtn(ushort3);
10268 double3 const_func __attribute__((overloadable)) convert_double3(ushort3);
10269 double3 const_func __attribute__((overloadable)) convert_double3_rte(int3);
10270 double3 const_func __attribute__((overloadable)) convert_double3_rtz(int3);
10271 double3 const_func __attribute__((overloadable)) convert_double3_rtp(int3);
10272 double3 const_func __attribute__((overloadable)) convert_double3_rtn(int3);
10273 double3 const_func __attribute__((overloadable)) convert_double3(int3);
10274 double3 const_func __attribute__((overloadable)) convert_double3_rte(uint3);
10275 double3 const_func __attribute__((overloadable)) convert_double3_rtz(uint3);
10276 double3 const_func __attribute__((overloadable)) convert_double3_rtp(uint3);
10277 double3 const_func __attribute__((overloadable)) convert_double3_rtn(uint3);
10278 double3 const_func __attribute__((overloadable)) convert_double3(uint3);
```

```
10279 double3 const_func __attribute__((overloadable)) convert_double3_rte(long3);
10280 double3 const_func __attribute__((overloadable)) convert_double3_rtz(long3);
10281 double3 const_func __attribute__((overloadable)) convert_double3_rtp(long3);
10282 double3 const_func __attribute__((overloadable)) convert_double3_rtn(long3);
10283 double3 const_func __attribute__((overloadable)) convert_double3(long3);
10284 double3 const_func __attribute__((overloadable)) convert_double3_rte(ulong3);
10285 double3 const_func __attribute__((overloadable)) convert_double3_rtz(ulong3);
10286 double3 const_func __attribute__((overloadable)) convert_double3_rtp(ulong3);
10287 double3 const_func __attribute__((overloadable)) convert_double3_rtn(ulong3);
10288 double3 const_func __attribute__((overloadable)) convert_double3(ulong3);
10289 double3 const_func __attribute__((overloadable)) convert_double3_rte(float3);
10290 double3 const_func __attribute__((overloadable)) convert_double3_rtz(float3);
10291 double3 const_func __attribute__((overloadable)) convert_double3_rtp(float3);
10292 double3 const_func __attribute__((overloadable)) convert_double3_rtn(float3);
10293 double3 const_func __attribute__((overloadable)) convert_double3(float3);
10294 double3 const_func __attribute__((overloadable)) convert_double3_rte(double3);
10295 double3 const_func __attribute__((overloadable)) convert_double3_rtz(double3);
10296 double3 const_func __attribute__((overloadable)) convert_double3_rtp(double3);
10297 double3 const_func __attribute__((overloadable)) convert_double3_rtn(double3);
10298 double3 const_func __attribute__((overloadable)) convert_double3(double3);
10299
10300 char4 const_func __attribute__((overloadable)) convert_char4_rte(char4);
10301 char4 const_func __attribute__((overloadable)) convert_char4_sat_rte(char4);
10302 char4 const_func __attribute__((overloadable)) convert_char4_rtz(char4);
10303 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtz(char4);
10304 char4 const_func __attribute__((overloadable)) convert_char4_rtp(char4);
10305 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtp(char4);
10306 char4 const_func __attribute__((overloadable)) convert_char4_rtn(char4);
10307 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtn(char4);
10308 char4 const_func __attribute__((overloadable)) convert_char4(char4);
10309 char4 const_func __attribute__((overloadable)) convert_char4_sat(char4);
10310 char4 const_func __attribute__((overloadable)) convert_char4_rte(uchar4);
10311 char4 const_func __attribute__((overloadable)) convert_char4_sat_rte(uchar4);
10312 char4 const_func __attribute__((overloadable)) convert_char4_rtz(uchar4);
10313 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtz(uchar4);
10314 char4 const_func __attribute__((overloadable)) convert_char4_rtp(uchar4);
10315 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtp(uchar4);
10316 char4 const_func __attribute__((overloadable)) convert_char4_rtn(uchar4);
10317 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtn(uchar4);
10318 char4 const_func __attribute__((overloadable)) convert_char4(uchar4);
10319 char4 const_func __attribute__((overloadable)) convert_char4_sat(uchar4);
10320 char4 const_func __attribute__((overloadable)) convert_char4_rte(short4);
10321 char4 const_func __attribute__((overloadable)) convert_char4_sat_rte(short4);
10322 char4 const_func __attribute__((overloadable)) convert_char4_rtz(short4);
10323 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtz(short4);
10324 char4 const_func __attribute__((overloadable)) convert_char4_rtp(short4);
10325 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtp(short4);
10326 char4 const_func __attribute__((overloadable)) convert_char4_rtn(short4);
10327 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtn(short4);
10328 char4 const_func __attribute__((overloadable)) convert_char4(short4);
10329 char4 const_func __attribute__((overloadable)) convert_char4_sat(short4);
10330 char4 const_func __attribute__((overloadable)) convert_char4_rte(ushort4);
10331 char4 const_func __attribute__((overloadable)) convert_char4_sat_rte(ushort4);
10332 char4 const_func __attribute__((overloadable)) convert_char4_rtz(ushort4);
10333 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtz(ushort4);
10334 char4 const_func __attribute__((overloadable)) convert_char4_rtp(ushort4);
10335 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtp(ushort4);
10336 char4 const_func __attribute__((overloadable)) convert_char4_rtn(ushort4);
10337 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtn(ushort4);
10338 char4 const_func __attribute__((overloadable)) convert_char4(ushort4);
10339 char4 const_func __attribute__((overloadable)) convert_char4_sat(ushort4);
10340 char4 const_func __attribute__((overloadable)) convert_char4_rte(int4);
10341 char4 const_func __attribute__((overloadable)) convert_char4_sat_rte(int4);
10342 char4 const_func __attribute__((overloadable)) convert_char4_rtz(int4);
10343 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtz(int4);
10344 char4 const_func __attribute__((overloadable)) convert_char4_rtp(int4);
10345 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtp(int4);
10346 char4 const_func __attribute__((overloadable)) convert_char4_rtn(int4);
10347 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtn(int4);
10348 char4 const_func __attribute__((overloadable)) convert_char4(int4);
10349 char4 const_func __attribute__((overloadable)) convert_char4_sat(int4);
10350 char4 const_func __attribute__((overloadable)) convert_char4_rte(uint4);
10351 char4 const_func __attribute__((overloadable)) convert_char4_sat_rte(uint4);
10352 char4 const_func __attribute__((overloadable)) convert_char4_rtz(uint4);
10353 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtz(uint4);
10354 char4 const_func __attribute__((overloadable)) convert_char4_rtp(uint4);
10355 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtp(uint4);
10356 char4 const_func __attribute__((overloadable)) convert_char4_rtn(uint4);
10357 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtn(uint4);
10358 char4 const_func __attribute__((overloadable)) convert_char4(uint4);
10359 char4 const_func __attribute__((overloadable)) convert_char4_sat(uint4);
10360 char4 const_func __attribute__((overloadable)) convert_char4_rte(long4);
10361 char4 const_func __attribute__((overloadable)) convert_char4_sat_rte(long4);
10362 char4 const_func __attribute__((overloadable)) convert_char4_rtz(long4);
10363 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtz(long4);
10364 char4 const_func __attribute__((overloadable)) convert_char4_rtp(long4);
10365 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtp(long4);
```

```
10366 char4 const_func __attribute__((overloadable)) convert_char4_rtn(long4);
10367 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtn(long4);
10368 char4 const_func __attribute__((overloadable)) convert_char4(long4);
10369 char4 const_func __attribute__((overloadable)) convert_char4_sat(long4);
10370 char4 const_func __attribute__((overloadable)) convert_char4_rte(ulong4);
10371 char4 const_func __attribute__((overloadable)) convert_char4_sat_rte(ulong4);
10372 char4 const_func __attribute__((overloadable)) convert_char4_rtz(ulong4);
10373 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtz(ulong4);
10374 char4 const_func __attribute__((overloadable)) convert_char4_rtp(ulong4);
10375 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtp(ulong4);
10376 char4 const_func __attribute__((overloadable)) convert_char4_rtn(ulong4);
10377 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtn(ulong4);
10378 char4 const_func __attribute__((overloadable)) convert_char4(ulong4);
10379 char4 const_func __attribute__((overloadable)) convert_char4_sat(ulong4);
10380 char4 const_func __attribute__((overloadable)) convert_char4_rte(float4);
10381 char4 const_func __attribute__((overloadable)) convert_char4_sat_rte(float4);
10382 char4 const_func __attribute__((overloadable)) convert_char4_rtz(float4);
10383 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtz(float4);
10384 char4 const_func __attribute__((overloadable)) convert_char4_rtp(float4);
10385 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtp(float4);
10386 char4 const_func __attribute__((overloadable)) convert_char4_rtn(float4);
10387 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtn(float4);
10388 char4 const_func __attribute__((overloadable)) convert_char4(float4);
10389 char4 const_func __attribute__((overloadable)) convert_char4_sat(float4);
10390 char4 const_func __attribute__((overloadable)) convert_char4_rte(double4);
10391 char4 const_func __attribute__((overloadable)) convert_char4_sat_rte(double4);
10392 char4 const_func __attribute__((overloadable)) convert_char4_rtz(double4);
10393 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtz(double4);
10394 char4 const_func __attribute__((overloadable)) convert_char4_rtp(double4);
10395 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtp(double4);
10396 char4 const_func __attribute__((overloadable)) convert_char4_rtn(double4);
10397 char4 const_func __attribute__((overloadable)) convert_char4_sat_rtn(double4);
10398 char4 const_func __attribute__((overloadable)) convert_char4(double4);
10399 char4 const_func __attribute__((overloadable)) convert_char4_sat(double4);
10400
10401 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rte(char4);
10402 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rte(char4);
10403 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtz(char4);
10404 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtz(char4);
10405 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtp(char4);
10406 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtp(char4);
10407 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtn(char4);
10408 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtn(char4);
10409 uchar4 const_func __attribute__((overloadable)) convert_uchar4(char4);
10410 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat(char4);
10411 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rte(uchar4);
10412 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rte(uchar4);
10413 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtz(uchar4);
10414 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtz(uchar4);
10415 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtp(uchar4);
10416 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtp(uchar4);
10417 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtn(uchar4);
10418 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtn(uchar4);
10419 uchar4 const_func __attribute__((overloadable)) convert_uchar4(uchar4);
10420 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat(uchar4);
10421 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rte(short4);
10422 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rte(short4);
10423 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtz(short4);
10424 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtz(short4);
10425 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtp(short4);
10426 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtp(short4);
10427 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtn(short4);
10428 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtn(short4);
10429 uchar4 const_func __attribute__((overloadable)) convert_uchar4(short4);
10430 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat(short4);
10431 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rte(ushort4);
10432 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rte(ushort4);
10433 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtz(ushort4);
10434 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtz(ushort4);
10435 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtp(ushort4);
10436 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtp(ushort4);
10437 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtn(ushort4);
10438 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtn(ushort4);
10439 uchar4 const_func __attribute__((overloadable)) convert_uchar4(ushort4);
10440 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat(ushort4);
10441 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rte(int4);
10442 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rte(int4);
10443 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtz(int4);
10444 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtz(int4);
10445 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtp(int4);
10446 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtp(int4);
10447 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtn(int4);
10448 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtn(int4);
10449 uchar4 const_func __attribute__((overloadable)) convert_uchar4(int4);
10450 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat(int4);
10451 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rte(uint4);
10452 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rte(uint4);
```

```
10453 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtz(uint4);
10454 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtz(uint4);
10455 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtp(uint4);
10456 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtp(uint4);
10457 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtn(uint4);
10458 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtn(uint4);
10459 uchar4 const_func __attribute__((overloadable)) convert_uchar4(uint4);
10460 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat(uint4);
10461 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rte(long4);
10462 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rte(long4);
10463 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtz(long4);
10464 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtz(long4);
10465 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtp(long4);
10466 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtp(long4);
10467 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtn(long4);
10468 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtn(long4);
10469 uchar4 const_func __attribute__((overloadable)) convert_uchar4(long4);
10470 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat(long4);
10471 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rte(ulong4);
10472 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rte(ulong4);
10473 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtz(ulong4);
10474 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtz(ulong4);
10475 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtp(ulong4);
10476 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtp(ulong4);
10477 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtn(ulong4);
10478 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtn(ulong4);
10479 uchar4 const_func __attribute__((overloadable)) convert_uchar4(ulong4);
10480 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat(ulong4);
10481 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rte(float4);
10482 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rte(float4);
10483 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtz(float4);
10484 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtz(float4);
10485 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtp(float4);
10486 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtp(float4);
10487 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtn(float4);
10488 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtn(float4);
10489 uchar4 const_func __attribute__((overloadable)) convert_uchar4(float4);
10490 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat(float4);
10491 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rte(double4);
10492 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rte(double4);
10493 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtz(double4);
10494 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtz(double4);
10495 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtp(double4);
10496 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtp(double4);
10497 uchar4 const_func __attribute__((overloadable)) convert_uchar4_rtn(double4);
10498 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat_rtn(double4);
10499 uchar4 const_func __attribute__((overloadable)) convert_uchar4(double4);
10500 uchar4 const_func __attribute__((overloadable)) convert_uchar4_sat(double4);
10501
10502 short4 const_func __attribute__((overloadable)) convert_short4_rte(char4);
10503 short4 const_func __attribute__((overloadable)) convert_short4_sat_rte(char4);
10504 short4 const_func __attribute__((overloadable)) convert_short4_rtz(char4);
10505 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtz(char4);
10506 short4 const_func __attribute__((overloadable)) convert_short4_rtp(char4);
10507 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtp(char4);
10508 short4 const_func __attribute__((overloadable)) convert_short4_rtn(char4);
10509 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtn(char4);
10510 short4 const_func __attribute__((overloadable)) convert_short4(char4);
10511 short4 const_func __attribute__((overloadable)) convert_short4_sat(char4);
10512 short4 const_func __attribute__((overloadable)) convert_short4_rte(uchar4);
10513 short4 const_func __attribute__((overloadable)) convert_short4_sat_rte(uchar4);
10514 short4 const_func __attribute__((overloadable)) convert_short4_rtz(uchar4);
10515 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtz(uchar4);
10516 short4 const_func __attribute__((overloadable)) convert_short4_rtp(uchar4);
10517 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtp(uchar4);
10518 short4 const_func __attribute__((overloadable)) convert_short4_rtn(uchar4);
10519 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtn(uchar4);
10520 short4 const_func __attribute__((overloadable)) convert_short4(uchar4);
10521 short4 const_func __attribute__((overloadable)) convert_short4_sat(uchar4);
10522 short4 const_func __attribute__((overloadable)) convert_short4_rte(short4);
10523 short4 const_func __attribute__((overloadable)) convert_short4_sat_rte(short4);
10524 short4 const_func __attribute__((overloadable)) convert_short4_rtz(short4);
10525 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtz(short4);
10526 short4 const_func __attribute__((overloadable)) convert_short4_rtp(short4);
10527 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtp(short4);
10528 short4 const_func __attribute__((overloadable)) convert_short4_rtn(short4);
10529 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtn(short4);
10530 short4 const_func __attribute__((overloadable)) convert_short4(short4);
10531 short4 const_func __attribute__((overloadable)) convert_short4_sat(short4);
10532 short4 const_func __attribute__((overloadable)) convert_short4_rte(ushort4);
10533 short4 const_func __attribute__((overloadable)) convert_short4_sat_rte(ushort4);
10534 short4 const_func __attribute__((overloadable)) convert_short4_rtz(ushort4);
10535 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtz(ushort4);
10536 short4 const_func __attribute__((overloadable)) convert_short4_rtp(ushort4);
10537 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtp(ushort4);
10538 short4 const_func __attribute__((overloadable)) convert_short4_rtn(ushort4);
10539 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtn(ushort4);
```

```
10540 short4 const_func __attribute__((overloadable)) convert_short4(ushort4);
10541 short4 const_func __attribute__((overloadable)) convert_short4_sat(ushort4);
10542 short4 const_func __attribute__((overloadable)) convert_short4_rte(int4);
10543 short4 const_func __attribute__((overloadable)) convert_short4_sat_rte(int4);
10544 short4 const_func __attribute__((overloadable)) convert_short4_rtz(int4);
10545 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtz(int4);
10546 short4 const_func __attribute__((overloadable)) convert_short4_rtp(int4);
10547 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtp(int4);
10548 short4 const_func __attribute__((overloadable)) convert_short4_rtn(int4);
10549 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtn(int4);
10550 short4 const_func __attribute__((overloadable)) convert_short4(int4);
10551 short4 const_func __attribute__((overloadable)) convert_short4_sat(int4);
10552 short4 const_func __attribute__((overloadable)) convert_short4_rte(uint4);
10553 short4 const_func __attribute__((overloadable)) convert_short4_sat_rte(uint4);
10554 short4 const_func __attribute__((overloadable)) convert_short4_rtz(uint4);
10555 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtz(uint4);
10556 short4 const_func __attribute__((overloadable)) convert_short4_rtp(uint4);
10557 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtp(uint4);
10558 short4 const_func __attribute__((overloadable)) convert_short4_rtn(uint4);
10559 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtn(uint4);
10560 short4 const_func __attribute__((overloadable)) convert_short4(uint4);
10561 short4 const_func __attribute__((overloadable)) convert_short4_sat(uint4);
10562 short4 const_func __attribute__((overloadable)) convert_short4_rte(long4);
10563 short4 const_func __attribute__((overloadable)) convert_short4_sat_rte(long4);
10564 short4 const_func __attribute__((overloadable)) convert_short4_rtz(long4);
10565 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtz(long4);
10566 short4 const_func __attribute__((overloadable)) convert_short4_rtp(long4);
10567 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtp(long4);
10568 short4 const_func __attribute__((overloadable)) convert_short4_rtn(long4);
10569 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtn(long4);
10570 short4 const_func __attribute__((overloadable)) convert_short4(long4);
10571 short4 const_func __attribute__((overloadable)) convert_short4_sat(long4);
10572 short4 const_func __attribute__((overloadable)) convert_short4_rte(ulong4);
10573 short4 const_func __attribute__((overloadable)) convert_short4_sat_rte(ulong4);
10574 short4 const_func __attribute__((overloadable)) convert_short4_rtz(ulong4);
10575 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtz(ulong4);
10576 short4 const_func __attribute__((overloadable)) convert_short4_rtp(ulong4);
10577 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtp(ulong4);
10578 short4 const_func __attribute__((overloadable)) convert_short4_rtn(ulong4);
10579 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtn(ulong4);
10580 short4 const_func __attribute__((overloadable)) convert_short4(ulong4);
10581 short4 const_func __attribute__((overloadable)) convert_short4_sat(ulong4);
10582 short4 const_func __attribute__((overloadable)) convert_short4_rte(float4);
10583 short4 const_func __attribute__((overloadable)) convert_short4_sat_rte(float4);
10584 short4 const_func __attribute__((overloadable)) convert_short4_rtz(float4);
10585 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtz(float4);
10586 short4 const_func __attribute__((overloadable)) convert_short4_rtp(float4);
10587 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtp(float4);
10588 short4 const_func __attribute__((overloadable)) convert_short4_rtn(float4);
10589 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtn(float4);
10590 short4 const_func __attribute__((overloadable)) convert_short4(float4);
10591 short4 const_func __attribute__((overloadable)) convert_short4_sat(float4);
10592 short4 const_func __attribute__((overloadable)) convert_short4_rte(double4);
10593 short4 const_func __attribute__((overloadable)) convert_short4_sat_rte(double4);
10594 short4 const_func __attribute__((overloadable)) convert_short4_rtz(double4);
10595 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtz(double4);
10596 short4 const_func __attribute__((overloadable)) convert_short4_rtp(double4);
10597 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtp(double4);
10598 short4 const_func __attribute__((overloadable)) convert_short4_rtn(double4);
10599 short4 const_func __attribute__((overloadable)) convert_short4_sat_rtn(double4);
10600 short4 const_func __attribute__((overloadable)) convert_short4(double4);
10601 short4 const_func __attribute__((overloadable)) convert_short4_sat(double4);
10602
10603 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rte(char4);
10604 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rte(char4);
10605 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtz(char4);
10606 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtz(char4);
10607 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtp(char4);
10608 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtp(char4);
10609 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtn(char4);
10610 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtn(char4);
10611 ushort4 const_func __attribute__((overloadable)) convert_ushort4(char4);
10612 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat(char4);
10613 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rte(uchar4);
10614 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rte(uchar4);
10615 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtz(uchar4);
10616 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtz(uchar4);
10617 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtp(uchar4);
10618 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtp(uchar4);
10619 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtn(uchar4);
10620 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtn(uchar4);
10621 ushort4 const_func __attribute__((overloadable)) convert_ushort4(uchar4);
10622 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat(uchar4);
10623 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rte(short4);
10624 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rte(short4);
10625 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtz(short4);
10626 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtz(short4);
```

```
10627 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtp(short4);
10628 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtp(short4);
10629 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtn(short4);
10630 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtn(short4);
10631 ushort4 const_func __attribute__((overloadable)) convert_ushort4(short4);
10632 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat(short4);
10633 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rte(ushort4);
10634 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rte(ushort4);
10635 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtz(ushort4);
10636 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtz(ushort4);
10637 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtp(ushort4);
10638 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtp(ushort4);
10639 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtn(ushort4);
10640 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtn(ushort4);
10641 ushort4 const_func __attribute__((overloadable)) convert_ushort4(ushort4);
10642 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat(ushort4);
10643 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rte(int4);
10644 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rte(int4);
10645 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtz(int4);
10646 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtz(int4);
10647 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtp(int4);
10648 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtp(int4);
10649 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtn(int4);
10650 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtn(int4);
10651 ushort4 const_func __attribute__((overloadable)) convert_ushort4(int4);
10652 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat(int4);
10653 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rte(uint4);
10654 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rte(uint4);
10655 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtz(uint4);
10656 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtz(uint4);
10657 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtp(uint4);
10658 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtp(uint4);
10659 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtn(uint4);
10660 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtn(uint4);
10661 ushort4 const_func __attribute__((overloadable)) convert_ushort4(uint4);
10662 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat(uint4);
10663 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rte(long4);
10664 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rte(long4);
10665 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtz(long4);
10666 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtz(long4);
10667 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtp(long4);
10668 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtp(long4);
10669 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtn(long4);
10670 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtn(long4);
10671 ushort4 const_func __attribute__((overloadable)) convert_ushort4(long4);
10672 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat(long4);
10673 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rte(ulong4);
10674 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rte(ulong4);
10675 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtz(ulong4);
10676 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtz(ulong4);
10677 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtp(ulong4);
10678 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtp(ulong4);
10679 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtn(ulong4);
10680 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtn(ulong4);
10681 ushort4 const_func __attribute__((overloadable)) convert_ushort4(ulong4);
10682 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat(ulong4);
10683 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rte(float4);
10684 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rte(float4);
10685 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtz(float4);
10686 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtz(float4);
10687 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtp(float4);
10688 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtp(float4);
10689 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtn(float4);
10690 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtn(float4);
10691 ushort4 const_func __attribute__((overloadable)) convert_ushort4(float4);
10692 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat(float4);
10693 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rte(double4);
10694 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rte(double4);
10695 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtz(double4);
10696 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtz(double4);
10697 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtp(double4);
10698 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtp(double4);
10699 ushort4 const_func __attribute__((overloadable)) convert_ushort4_rtn(double4);
10700 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat_rtn(double4);
10701 ushort4 const_func __attribute__((overloadable)) convert_ushort4(double4);
10702 ushort4 const_func __attribute__((overloadable)) convert_ushort4_sat(double4);
10703
10704 int4 const_func __attribute__((overloadable)) convert_int4_rte(char4);
10705 int4 const_func __attribute__((overloadable)) convert_int4_sat_rte(char4);
10706 int4 const_func __attribute__((overloadable)) convert_int4_rtz(char4);
10707 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtz(char4);
10708 int4 const_func __attribute__((overloadable)) convert_int4_rtp(char4);
10709 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtp(char4);
10710 int4 const_func __attribute__((overloadable)) convert_int4_rtn(char4);
10711 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtn(char4);
10712 int4 const_func __attribute__((overloadable)) convert_int4(char4);
10713 int4 const_func __attribute__((overloadable)) convert_int4_sat(char4);
```

```
10714 int4 const_func __attribute__((overloadable)) convert_int4_rte(uchar4);
10715 int4 const_func __attribute__((overloadable)) convert_int4_sat_rte(uchar4);
10716 int4 const_func __attribute__((overloadable)) convert_int4_rtz(uchar4);
10717 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtz(uchar4);
10718 int4 const_func __attribute__((overloadable)) convert_int4_rtp(uchar4);
10719 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtp(uchar4);
10720 int4 const_func __attribute__((overloadable)) convert_int4_rtn(uchar4);
10721 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtn(uchar4);
10722 int4 const_func __attribute__((overloadable)) convert_int4(uchar4);
10723 int4 const_func __attribute__((overloadable)) convert_int4_sat(uchar4);
10724 int4 const_func __attribute__((overloadable)) convert_int4_rte(short4);
10725 int4 const_func __attribute__((overloadable)) convert_int4_sat_rte(short4);
10726 int4 const_func __attribute__((overloadable)) convert_int4_rtz(short4);
10727 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtz(short4);
10728 int4 const_func __attribute__((overloadable)) convert_int4_rtp(short4);
10729 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtp(short4);
10730 int4 const_func __attribute__((overloadable)) convert_int4_rtn(short4);
10731 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtn(short4);
10732 int4 const_func __attribute__((overloadable)) convert_int4(short4);
10733 int4 const_func __attribute__((overloadable)) convert_int4_sat(short4);
10734 int4 const_func __attribute__((overloadable)) convert_int4_rte(ushort4);
10735 int4 const_func __attribute__((overloadable)) convert_int4_sat_rte(ushort4);
10736 int4 const_func __attribute__((overloadable)) convert_int4_rtz(ushort4);
10737 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtz(ushort4);
10738 int4 const_func __attribute__((overloadable)) convert_int4_rtp(ushort4);
10739 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtp(ushort4);
10740 int4 const_func __attribute__((overloadable)) convert_int4_rtn(ushort4);
10741 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtn(ushort4);
10742 int4 const_func __attribute__((overloadable)) convert_int4(ushort4);
10743 int4 const_func __attribute__((overloadable)) convert_int4_sat(ushort4);
10744 int4 const_func __attribute__((overloadable)) convert_int4_rte(int4);
10745 int4 const_func __attribute__((overloadable)) convert_int4_sat_rte(int4);
10746 int4 const_func __attribute__((overloadable)) convert_int4_rtz(int4);
10747 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtz(int4);
10748 int4 const_func __attribute__((overloadable)) convert_int4_rtp(int4);
10749 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtp(int4);
10750 int4 const_func __attribute__((overloadable)) convert_int4_rtn(int4);
10751 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtn(int4);
10752 int4 const_func __attribute__((overloadable)) convert_int4(int4);
10753 int4 const_func __attribute__((overloadable)) convert_int4_sat(int4);
10754 int4 const_func __attribute__((overloadable)) convert_int4_rte(uint4);
10755 int4 const_func __attribute__((overloadable)) convert_int4_sat_rte(uint4);
10756 int4 const_func __attribute__((overloadable)) convert_int4_rtz(uint4);
10757 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtz(uint4);
10758 int4 const_func __attribute__((overloadable)) convert_int4_rtp(uint4);
10759 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtp(uint4);
10760 int4 const_func __attribute__((overloadable)) convert_int4_rtn(uint4);
10761 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtn(uint4);
10762 int4 const_func __attribute__((overloadable)) convert_int4(uint4);
10763 int4 const_func __attribute__((overloadable)) convert_int4_sat(uint4);
10764 int4 const_func __attribute__((overloadable)) convert_int4_rte(long4);
10765 int4 const_func __attribute__((overloadable)) convert_int4_sat_rte(long4);
10766 int4 const_func __attribute__((overloadable)) convert_int4_rtz(long4);
10767 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtz(long4);
10768 int4 const_func __attribute__((overloadable)) convert_int4_rtp(long4);
10769 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtp(long4);
10770 int4 const_func __attribute__((overloadable)) convert_int4_rtn(long4);
10771 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtn(long4);
10772 int4 const_func __attribute__((overloadable)) convert_int4(long4);
10773 int4 const_func __attribute__((overloadable)) convert_int4_sat(long4);
10774 int4 const_func __attribute__((overloadable)) convert_int4_rte(ulong4);
10775 int4 const_func __attribute__((overloadable)) convert_int4_sat_rte(ulong4);
10776 int4 const_func __attribute__((overloadable)) convert_int4_rtz(ulong4);
10777 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtz(ulong4);
10778 int4 const_func __attribute__((overloadable)) convert_int4_rtp(ulong4);
10779 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtp(ulong4);
10780 int4 const_func __attribute__((overloadable)) convert_int4_rtn(ulong4);
10781 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtn(ulong4);
10782 int4 const_func __attribute__((overloadable)) convert_int4(ulong4);
10783 int4 const_func __attribute__((overloadable)) convert_int4_sat(ulong4);
10784 int4 const_func __attribute__((overloadable)) convert_int4_rte(float4);
10785 int4 const_func __attribute__((overloadable)) convert_int4_sat_rte(float4);
10786 int4 const_func __attribute__((overloadable)) convert_int4_rtz(float4);
10787 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtz(float4);
10788 int4 const_func __attribute__((overloadable)) convert_int4_rtp(float4);
10789 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtp(float4);
10790 int4 const_func __attribute__((overloadable)) convert_int4_rtn(float4);
10791 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtn(float4);
10792 int4 const_func __attribute__((overloadable)) convert_int4(float4);
10793 int4 const_func __attribute__((overloadable)) convert_int4_sat(float4);
10794 int4 const_func __attribute__((overloadable)) convert_int4_rte(double4);
10795 int4 const_func __attribute__((overloadable)) convert_int4_sat_rte(double4);
10796 int4 const_func __attribute__((overloadable)) convert_int4_rtz(double4);
10797 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtz(double4);
10798 int4 const_func __attribute__((overloadable)) convert_int4_rtp(double4);
10799 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtp(double4);
10800 int4 const_func __attribute__((overloadable)) convert_int4_rtn(double4);
```

```
10801 int4 const_func __attribute__((overloadable)) convert_int4_sat_rtn(double4);
10802 int4 const_func __attribute__((overloadable)) convert_int4(double4);
10803 int4 const_func __attribute__((overloadable)) convert_int4_sat(double4);
10804
10805 uint4 const_func __attribute__((overloadable)) convert_uint4_rte(char4);
10806 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rte(char4);
10807 uint4 const_func __attribute__((overloadable)) convert_uint4_rtz(char4);
10808 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtz(char4);
10809 uint4 const_func __attribute__((overloadable)) convert_uint4_rtp(char4);
10810 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtp(char4);
10811 uint4 const_func __attribute__((overloadable)) convert_uint4_rtn(char4);
10812 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtn(char4);
10813 uint4 const_func __attribute__((overloadable)) convert_uint4(char4);
10814 uint4 const_func __attribute__((overloadable)) convert_uint4_sat(char4);
10815 uint4 const_func __attribute__((overloadable)) convert_uint4_rte(uchar4);
10816 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rte(uchar4);
10817 uint4 const_func __attribute__((overloadable)) convert_uint4_rtz(uchar4);
10818 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtz(uchar4);
10819 uint4 const_func __attribute__((overloadable)) convert_uint4_rtp(uchar4);
10820 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtp(uchar4);
10821 uint4 const_func __attribute__((overloadable)) convert_uint4_rtn(uchar4);
10822 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtn(uchar4);
10823 uint4 const_func __attribute__((overloadable)) convert_uint4(uchar4);
10824 uint4 const_func __attribute__((overloadable)) convert_uint4_sat(uchar4);
10825 uint4 const_func __attribute__((overloadable)) convert_uint4_rte(short4);
10826 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rte(short4);
10827 uint4 const_func __attribute__((overloadable)) convert_uint4_rtz(short4);
10828 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtz(short4);
10829 uint4 const_func __attribute__((overloadable)) convert_uint4_rtp(short4);
10830 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtp(short4);
10831 uint4 const_func __attribute__((overloadable)) convert_uint4_rtn(short4);
10832 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtn(short4);
10833 uint4 const_func __attribute__((overloadable)) convert_uint4(short4);
10834 uint4 const_func __attribute__((overloadable)) convert_uint4_sat(short4);
10835 uint4 const_func __attribute__((overloadable)) convert_uint4_rte(ushort4);
10836 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rte(ushort4);
10837 uint4 const_func __attribute__((overloadable)) convert_uint4_rtz(ushort4);
10838 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtz(ushort4);
10839 uint4 const_func __attribute__((overloadable)) convert_uint4_rtp(ushort4);
10840 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtp(ushort4);
10841 uint4 const_func __attribute__((overloadable)) convert_uint4_rtn(ushort4);
10842 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtn(ushort4);
10843 uint4 const_func __attribute__((overloadable)) convert_uint4(ushort4);
10844 uint4 const_func __attribute__((overloadable)) convert_uint4_sat(ushort4);
10845 uint4 const_func __attribute__((overloadable)) convert_uint4_rte(int4);
10846 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rte(int4);
10847 uint4 const_func __attribute__((overloadable)) convert_uint4_rtz(int4);
10848 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtz(int4);
10849 uint4 const_func __attribute__((overloadable)) convert_uint4_rtp(int4);
10850 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtp(int4);
10851 uint4 const_func __attribute__((overloadable)) convert_uint4_rtn(int4);
10852 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtn(int4);
10853 uint4 const_func __attribute__((overloadable)) convert_uint4(int4);
10854 uint4 const_func __attribute__((overloadable)) convert_uint4_sat(int4);
10855 uint4 const_func __attribute__((overloadable)) convert_uint4_rte(uint4);
10856 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rte(uint4);
10857 uint4 const_func __attribute__((overloadable)) convert_uint4_rtz(uint4);
10858 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtz(uint4);
10859 uint4 const_func __attribute__((overloadable)) convert_uint4_rtp(uint4);
10860 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtp(uint4);
10861 uint4 const_func __attribute__((overloadable)) convert_uint4_rtn(uint4);
10862 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtn(uint4);
10863 uint4 const_func __attribute__((overloadable)) convert_uint4(uint4);
10864 uint4 const_func __attribute__((overloadable)) convert_uint4_sat(uint4);
10865 uint4 const_func __attribute__((overloadable)) convert_uint4_rte(long4);
10866 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rte(long4);
10867 uint4 const_func __attribute__((overloadable)) convert_uint4_rtz(long4);
10868 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtz(long4);
10869 uint4 const_func __attribute__((overloadable)) convert_uint4_rtp(long4);
10870 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtp(long4);
10871 uint4 const_func __attribute__((overloadable)) convert_uint4_rtn(long4);
10872 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtn(long4);
10873 uint4 const_func __attribute__((overloadable)) convert_uint4(long4);
10874 uint4 const_func __attribute__((overloadable)) convert_uint4_sat(long4);
10875 uint4 const_func __attribute__((overloadable)) convert_uint4_rte(ulong4);
10876 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rte(ulong4);
10877 uint4 const_func __attribute__((overloadable)) convert_uint4_rtz(ulong4);
10878 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtz(ulong4);
10879 uint4 const_func __attribute__((overloadable)) convert_uint4_rtp(ulong4);
10880 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtp(ulong4);
10881 uint4 const_func __attribute__((overloadable)) convert_uint4_rtn(ulong4);
10882 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtn(ulong4);
10883 uint4 const_func __attribute__((overloadable)) convert_uint4(ulong4);
10884 uint4 const_func __attribute__((overloadable)) convert_uint4_sat(ulong4);
10885 uint4 const_func __attribute__((overloadable)) convert_uint4_rte(float4);
10886 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rte(float4);
10887 uint4 const_func __attribute__((overloadable)) convert_uint4_rtz(float4);
```

```
10888 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtz(float4);
10889 uint4 const_func __attribute__((overloadable)) convert_uint4_rtp(float4);
10890 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtp(float4);
10891 uint4 const_func __attribute__((overloadable)) convert_uint4_rtn(float4);
10892 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtn(float4);
10893 uint4 const_func __attribute__((overloadable)) convert_uint4(float4);
10894 uint4 const_func __attribute__((overloadable)) convert_uint4_sat(float4);
10895 uint4 const_func __attribute__((overloadable)) convert_uint4_rte(double4);
10896 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rte(double4);
10897 uint4 const_func __attribute__((overloadable)) convert_uint4_rtz(double4);
10898 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtz(double4);
10899 uint4 const_func __attribute__((overloadable)) convert_uint4_rtp(double4);
10900 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtp(double4);
10901 uint4 const_func __attribute__((overloadable)) convert_uint4_rtn(double4);
10902 uint4 const_func __attribute__((overloadable)) convert_uint4_sat_rtn(double4);
10903 uint4 const_func __attribute__((overloadable)) convert_uint4(double4);
10904 uint4 const_func __attribute__((overloadable)) convert_uint4_sat(double4);
10905 long4 const_func __attribute__((overloadable)) convert_long4_rte(char4);
10906 long4 const_func __attribute__((overloadable)) convert_long4_sat_rte(char4);
10907 long4 const_func __attribute__((overloadable)) convert_long4_rtz(char4);
10908 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtz(char4);
10909 long4 const_func __attribute__((overloadable)) convert_long4_rtp(char4);
10910 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtp(char4);
10911 long4 const_func __attribute__((overloadable)) convert_long4_rtn(char4);
10912 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtn(char4);
10913 long4 const_func __attribute__((overloadable)) convert_long4(char4);
10914 long4 const_func __attribute__((overloadable)) convert_long4_sat(char4);
10915 long4 const_func __attribute__((overloadable)) convert_long4_rte(uchar4);
10916 long4 const_func __attribute__((overloadable)) convert_long4_sat_rte(uchar4);
10917 long4 const_func __attribute__((overloadable)) convert_long4_rtz(uchar4);
10918 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtz(uchar4);
10919 long4 const_func __attribute__((overloadable)) convert_long4_rtp(uchar4);
10920 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtp(uchar4);
10921 long4 const_func __attribute__((overloadable)) convert_long4_rtn(uchar4);
10922 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtn(uchar4);
10923 long4 const_func __attribute__((overloadable)) convert_long4(uchar4);
10924 long4 const_func __attribute__((overloadable)) convert_long4_sat(uchar4);
10925 long4 const_func __attribute__((overloadable)) convert_long4_rte(short4);
10926 long4 const_func __attribute__((overloadable)) convert_long4_sat_rte(short4);
10927 long4 const_func __attribute__((overloadable)) convert_long4_rtz(short4);
10928 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtz(short4);
10929 long4 const_func __attribute__((overloadable)) convert_long4_rtp(short4);
10930 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtp(short4);
10931 long4 const_func __attribute__((overloadable)) convert_long4_rtn(short4);
10932 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtn(short4);
10933 long4 const_func __attribute__((overloadable)) convert_long4(short4);
10934 long4 const_func __attribute__((overloadable)) convert_long4_sat(short4);
10935 long4 const_func __attribute__((overloadable)) convert_long4_rte(ushort4);
10936 long4 const_func __attribute__((overloadable)) convert_long4_sat_rte(ushort4);
10937 long4 const_func __attribute__((overloadable)) convert_long4_rtz(ushort4);
10938 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtz(ushort4);
10939 long4 const_func __attribute__((overloadable)) convert_long4_rtp(ushort4);
10940 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtp(ushort4);
10941 long4 const_func __attribute__((overloadable)) convert_long4_rtn(ushort4);
10942 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtn(ushort4);
10943 long4 const_func __attribute__((overloadable)) convert_long4(ushort4);
10944 long4 const_func __attribute__((overloadable)) convert_long4_sat(ushort4);
10945 long4 const_func __attribute__((overloadable)) convert_long4_rte(int4);
10946 long4 const_func __attribute__((overloadable)) convert_long4_sat_rte(int4);
10947 long4 const_func __attribute__((overloadable)) convert_long4_rtz(int4);
10948 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtz(int4);
10949 long4 const_func __attribute__((overloadable)) convert_long4_rtp(int4);
10950 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtp(int4);
10951 long4 const_func __attribute__((overloadable)) convert_long4_rtn(int4);
10952 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtn(int4);
10953 long4 const_func __attribute__((overloadable)) convert_long4(int4);
10954 long4 const_func __attribute__((overloadable)) convert_long4_sat(int4);
10955 long4 const_func __attribute__((overloadable)) convert_long4_rte(uint4);
10956 long4 const_func __attribute__((overloadable)) convert_long4_sat_rte(uint4);
10957 long4 const_func __attribute__((overloadable)) convert_long4_rtz(uint4);
10958 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtz(uint4);
10959 long4 const_func __attribute__((overloadable)) convert_long4_rtp(uint4);
10960 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtp(uint4);
10961 long4 const_func __attribute__((overloadable)) convert_long4_rtn(uint4);
10962 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtn(uint4);
10963 long4 const_func __attribute__((overloadable)) convert_long4(uint4);
10964 long4 const_func __attribute__((overloadable)) convert_long4_sat(uint4);
10965 long4 const_func __attribute__((overloadable)) convert_long4_rte(long4);
10966 long4 const_func __attribute__((overloadable)) convert_long4_sat_rte(long4);
10967 long4 const_func __attribute__((overloadable)) convert_long4_rtz(long4);
10968 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtz(long4);
10969 long4 const_func __attribute__((overloadable)) convert_long4_rtp(long4);
10970 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtp(long4);
10971 long4 const_func __attribute__((overloadable)) convert_long4_rtn(long4);
10972 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtn(long4);
10973 long4 const_func __attribute__((overloadable)) convert_long4(long4);
10974 long4 const_func __attribute__((overloadable)) convert_long4_sat(long4);
```

```
10975 long4 const_func __attribute__((overloadable)) convert_long4_rte(ulong4);
10976 long4 const_func __attribute__((overloadable)) convert_long4_sat_rte(ulong4);
10977 long4 const_func __attribute__((overloadable)) convert_long4_rtz(ulong4);
10978 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtz(ulong4);
10979 long4 const_func __attribute__((overloadable)) convert_long4_rtp(ulong4);
10980 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtp(ulong4);
10981 long4 const_func __attribute__((overloadable)) convert_long4_rtn(ulong4);
10982 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtn(ulong4);
10983 long4 const_func __attribute__((overloadable)) convert_long4(ulong4);
10984 long4 const_func __attribute__((overloadable)) convert_long4_sat(ulong4);
10985 long4 const_func __attribute__((overloadable)) convert_long4_rte(float4);
10986 long4 const_func __attribute__((overloadable)) convert_long4_sat_rte(float4);
10987 long4 const_func __attribute__((overloadable)) convert_long4_rtz(float4);
10988 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtz(float4);
10989 long4 const_func __attribute__((overloadable)) convert_long4_rtp(float4);
10990 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtp(float4);
10991 long4 const_func __attribute__((overloadable)) convert_long4_rtn(float4);
10992 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtn(float4);
10993 long4 const_func __attribute__((overloadable)) convert_long4(float4);
10994 long4 const_func __attribute__((overloadable)) convert_long4_sat(float4);
10995 long4 const_func __attribute__((overloadable)) convert_long4_rte(double4);
10996 long4 const_func __attribute__((overloadable)) convert_long4_sat_rte(double4);
10997 long4 const_func __attribute__((overloadable)) convert_long4_rtz(double4);
10998 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtz(double4);
10999 long4 const_func __attribute__((overloadable)) convert_long4_rtp(double4);
11000 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtp(double4);
11001 long4 const_func __attribute__((overloadable)) convert_long4_rtn(double4);
11002 long4 const_func __attribute__((overloadable)) convert_long4_sat_rtn(double4);
11003 long4 const_func __attribute__((overloadable)) convert_long4(double4);
11004 long4 const_func __attribute__((overloadable)) convert_long4_sat(double4);
11005
11006 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rte(char4);
11007 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rte(char4);
11008 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtz(char4);
11009 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtz(char4);
11010 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtp(char4);
11011 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtp(char4);
11012 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtn(char4);
11013 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtn(char4);
11014 ulong4 const_func __attribute__((overloadable)) convert_ulong4(char4);
11015 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat(char4);
11016 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rte(uchar4);
11017 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rte(uchar4);
11018 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtz(uchar4);
11019 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtz(uchar4);
11020 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtp(uchar4);
11021 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtp(uchar4);
11022 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtn(uchar4);
11023 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtn(uchar4);
11024 ulong4 const_func __attribute__((overloadable)) convert_ulong4(uchar4);
11025 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat(uchar4);
11026 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rte(short4);
11027 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rte(short4);
11028 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtz(short4);
11029 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtz(short4);
11030 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtp(short4);
11031 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtp(short4);
11032 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtn(short4);
11033 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtn(short4);
11034 ulong4 const_func __attribute__((overloadable)) convert_ulong4(short4);
11035 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat(short4);
11036 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rte(ushort4);
11037 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rte(ushort4);
11038 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtz(ushort4);
11039 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtz(ushort4);
11040 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtp(ushort4);
11041 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtp(ushort4);
11042 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtn(ushort4);
11043 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtn(ushort4);
11044 ulong4 const_func __attribute__((overloadable)) convert_ulong4(ushort4);
11045 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat(ushort4);
11046 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rte(int4);
11047 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rte(int4);
11048 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtz(int4);
11049 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtz(int4);
11050 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtp(int4);
11051 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtp(int4);
11052 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtn(int4);
11053 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtn(int4);
11054 ulong4 const_func __attribute__((overloadable)) convert_ulong4(int4);
11055 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat(int4);
11056 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rte(uint4);
11057 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rte(uint4);
11058 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtz(uint4);
11059 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtz(uint4);
11060 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtp(uint4);
11061 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtp(uint4);
```

```
11062 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtn(uint4);
11063 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtn(uint4);
11064 ulong4 const_func __attribute__((overloadable)) convert_ulong4(uint4);
11065 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat(uint4);
11066 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rte(long4);
11067 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rte(long4);
11068 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtz(long4);
11069 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtz(long4);
11070 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtp(long4);
11071 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtp(long4);
11072 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtn(long4);
11073 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtn(long4);
11074 ulong4 const_func __attribute__((overloadable)) convert_ulong4(long4);
11075 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat(long4);
11076 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rte(ulong4);
11077 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rte(ulong4);
11078 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtz(ulong4);
11079 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtz(ulong4);
11080 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtp(ulong4);
11081 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtp(ulong4);
11082 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtn(ulong4);
11083 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtn(ulong4);
11084 ulong4 const_func __attribute__((overloadable)) convert_ulong4(ulong4);
11085 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat(ulong4);
11086 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rte(float4);
11087 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rte(float4);
11088 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtz(float4);
11089 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtz(float4);
11090 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtp(float4);
11091 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtp(float4);
11092 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtn(float4);
11093 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtn(float4);
11094 ulong4 const_func __attribute__((overloadable)) convert_ulong4(float4);
11095 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat(float4);
11096 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rte(double4);
11097 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rte(double4);
11098 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtz(double4);
11099 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtz(double4);
11100 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtp(double4);
11101 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtp(double4);
11102 ulong4 const_func __attribute__((overloadable)) convert_ulong4_rtn(double4);
11103 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat_rtn(double4);
11104 ulong4 const_func __attribute__((overloadable)) convert_ulong4(double4);
11105 ulong4 const_func __attribute__((overloadable)) convert_ulong4_sat(double4);
11106
11107 float4 const_func __attribute__((overloadable)) convert_float4_rte(char4);
11108 float4 const_func __attribute__((overloadable)) convert_float4_rtz(char4);
11109 float4 const_func __attribute__((overloadable)) convert_float4_rtp(char4);
11110 float4 const_func __attribute__((overloadable)) convert_float4_rtn(char4);
11111 float4 const_func __attribute__((overloadable)) convert_float4(char4);
11112 float4 const_func __attribute__((overloadable)) convert_float4_rte(uchar4);
11113 float4 const_func __attribute__((overloadable)) convert_float4_rtz(uchar4);
11114 float4 const_func __attribute__((overloadable)) convert_float4_rtp(uchar4);
11115 float4 const_func __attribute__((overloadable)) convert_float4_rtn(uchar4);
11116 float4 const_func __attribute__((overloadable)) convert_float4(uchar4);
11117 float4 const_func __attribute__((overloadable)) convert_float4_rte(short4);
11118 float4 const_func __attribute__((overloadable)) convert_float4_rtz(short4);
11119 float4 const_func __attribute__((overloadable)) convert_float4_rtp(short4);
11120 float4 const_func __attribute__((overloadable)) convert_float4_rtn(short4);
11121 float4 const_func __attribute__((overloadable)) convert_float4(short4);
11122 float4 const_func __attribute__((overloadable)) convert_float4_rte(ushort4);
11123 float4 const_func __attribute__((overloadable)) convert_float4_rtz(ushort4);
11124 float4 const_func __attribute__((overloadable)) convert_float4_rtp(ushort4);
11125 float4 const_func __attribute__((overloadable)) convert_float4_rtn(ushort4);
11126 float4 const_func __attribute__((overloadable)) convert_float4(ushort4);
11127 float4 const_func __attribute__((overloadable)) convert_float4_rte(int4);
11128 float4 const_func __attribute__((overloadable)) convert_float4_rtz(int4);
11129 float4 const_func __attribute__((overloadable)) convert_float4_rtp(int4);
11130 float4 const_func __attribute__((overloadable)) convert_float4_rtn(int4);
11131 float4 const_func __attribute__((overloadable)) convert_float4(int4);
11132 float4 const_func __attribute__((overloadable)) convert_float4_rte(uint4);
11133 float4 const_func __attribute__((overloadable)) convert_float4_rtz(uint4);
11134 float4 const_func __attribute__((overloadable)) convert_float4_rtp(uint4);
11135 float4 const_func __attribute__((overloadable)) convert_float4_rtn(uint4);
11136 float4 const_func __attribute__((overloadable)) convert_float4(uint4);
11137 float4 const_func __attribute__((overloadable)) convert_float4_rte(long4);
11138 float4 const_func __attribute__((overloadable)) convert_float4_rtz(long4);
11139 float4 const_func __attribute__((overloadable)) convert_float4_rtp(long4);
11140 float4 const_func __attribute__((overloadable)) convert_float4_rtn(long4);
11141 float4 const_func __attribute__((overloadable)) convert_float4(long4);
11142 float4 const_func __attribute__((overloadable)) convert_float4_rte(ulong4);
11143 float4 const_func __attribute__((overloadable)) convert_float4_rtz(ulong4);
11144 float4 const_func __attribute__((overloadable)) convert_float4_rtp(ulong4);
11145 float4 const_func __attribute__((overloadable)) convert_float4_rtn(ulong4);
11146 float4 const_func __attribute__((overloadable)) convert_float4(ulong4);
11147 float4 const_func __attribute__((overloadable)) convert_float4_rte(float4);
11148 float4 const_func __attribute__((overloadable)) convert_float4_rtz(float4);
```

```
11149 float4 const_func __attribute__((overloadable)) convert_float4_rtp(float4);
11150 float4 const_func __attribute__((overloadable)) convert_float4_rtn(float4);
11151 float4 const_func __attribute__((overloadable)) convert_float4(float4);
11152 float4 const_func __attribute__((overloadable)) convert_float4_rte(double4);
11153 float4 const_func __attribute__((overloadable)) convert_float4_rtz(double4);
11154 float4 const_func __attribute__((overloadable)) convert_float4_rtp(double4);
11155 float4 const_func __attribute__((overloadable)) convert_float4_rtn(double4);
11156 float4 const_func __attribute__((overloadable)) convert_float4(double4);
11157
11158 double4 const_func __attribute__((overloadable)) convert_double4_rte(char4);
11159 double4 const_func __attribute__((overloadable)) convert_double4_rtz(char4);
11160 double4 const_func __attribute__((overloadable)) convert_double4_rtp(char4);
11161 double4 const_func __attribute__((overloadable)) convert_double4_rtn(char4);
11162 double4 const_func __attribute__((overloadable)) convert_double4(char4);
11163 double4 const_func __attribute__((overloadable)) convert_double4_rte(uchar4);
11164 double4 const_func __attribute__((overloadable)) convert_double4_rtz(uchar4);
11165 double4 const_func __attribute__((overloadable)) convert_double4_rtp(uchar4);
11166 double4 const_func __attribute__((overloadable)) convert_double4_rtn(uchar4);
11167 double4 const_func __attribute__((overloadable)) convert_double4(uchar4);
11168 double4 const_func __attribute__((overloadable)) convert_double4_rte(short4);
11169 double4 const_func __attribute__((overloadable)) convert_double4_rtz(short4);
11170 double4 const_func __attribute__((overloadable)) convert_double4_rtp(short4);
11171 double4 const_func __attribute__((overloadable)) convert_double4_rtn(short4);
11172 double4 const_func __attribute__((overloadable)) convert_double4(short4);
11173 double4 const_func __attribute__((overloadable)) convert_double4_rte(ushort4);
11174 double4 const_func __attribute__((overloadable)) convert_double4_rtz(ushort4);
11175 double4 const_func __attribute__((overloadable)) convert_double4_rtp(ushort4);
11176 double4 const_func __attribute__((overloadable)) convert_double4_rtn(ushort4);
11177 double4 const_func __attribute__((overloadable)) convert_double4(ushort4);
11178 double4 const_func __attribute__((overloadable)) convert_double4_rte(int4);
11179 double4 const_func __attribute__((overloadable)) convert_double4_rtz(int4);
11180 double4 const_func __attribute__((overloadable)) convert_double4_rtp(int4);
11181 double4 const_func __attribute__((overloadable)) convert_double4_rtn(int4);
11182 double4 const_func __attribute__((overloadable)) convert_double4(int4);
11183 double4 const_func __attribute__((overloadable)) convert_double4_rte(uint4);
11184 double4 const_func __attribute__((overloadable)) convert_double4_rtz(uint4);
11185 double4 const_func __attribute__((overloadable)) convert_double4_rtp(uint4);
11186 double4 const_func __attribute__((overloadable)) convert_double4_rtn(uint4);
11187 double4 const_func __attribute__((overloadable)) convert_double4(uint4);
11188 double4 const_func __attribute__((overloadable)) convert_double4_rte(long4);
11189 double4 const_func __attribute__((overloadable)) convert_double4_rtz(long4);
11190 double4 const_func __attribute__((overloadable)) convert_double4_rtp(long4);
11191 double4 const_func __attribute__((overloadable)) convert_double4_rtn(long4);
11192 double4 const_func __attribute__((overloadable)) convert_double4(long4);
11193 double4 const_func __attribute__((overloadable)) convert_double4_rte(ulong4);
11194 double4 const_func __attribute__((overloadable)) convert_double4_rtz(ulong4);
11195 double4 const_func __attribute__((overloadable)) convert_double4_rtp(ulong4);
11196 double4 const_func __attribute__((overloadable)) convert_double4_rtn(ulong4);
11197 double4 const_func __attribute__((overloadable)) convert_double4(ulong4);
11198 double4 const_func __attribute__((overloadable)) convert_double4_rte(float4);
11199 double4 const_func __attribute__((overloadable)) convert_double4_rtz(float4);
11200 double4 const_func __attribute__((overloadable)) convert_double4_rtp(float4);
11201 double4 const_func __attribute__((overloadable)) convert_double4_rtn(float4);
11202 double4 const_func __attribute__((overloadable)) convert_double4(float4);
11203 double4 const_func __attribute__((overloadable)) convert_double4_rte(double4);
11204 double4 const_func __attribute__((overloadable)) convert_double4_rtz(double4);
11205 double4 const_func __attribute__((overloadable)) convert_double4_rtp(double4);
11206 double4 const_func __attribute__((overloadable)) convert_double4_rtn(double4);
11207 double4 const_func __attribute__((overloadable)) convert_double4(double4);
11208
11209 char8 const_func __attribute__((overloadable)) convert_char8_rte(char8);
11210 char8 const_func __attribute__((overloadable)) convert_char8_sat_rte(char8);
11211 char8 const_func __attribute__((overloadable)) convert_char8_rtz(char8);
11212 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtz(char8);
11213 char8 const_func __attribute__((overloadable)) convert_char8_rtp(char8);
11214 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtp(char8);
11215 char8 const_func __attribute__((overloadable)) convert_char8_rtn(char8);
11216 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtn(char8);
11217 char8 const_func __attribute__((overloadable)) convert_char8(char8);
11218 char8 const_func __attribute__((overloadable)) convert_char8_sat(char8);
11219 char8 const_func __attribute__((overloadable)) convert_char8_rte(uchar8);
11220 char8 const_func __attribute__((overloadable)) convert_char8_sat_rte(uchar8);
11221 char8 const_func __attribute__((overloadable)) convert_char8_rtz(uchar8);
11222 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtz(uchar8);
11223 char8 const_func __attribute__((overloadable)) convert_char8_rtp(uchar8);
11224 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtp(uchar8);
11225 char8 const_func __attribute__((overloadable)) convert_char8_rtn(uchar8);
11226 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtn(uchar8);
11227 char8 const_func __attribute__((overloadable)) convert_char8(uchar8);
11228 char8 const_func __attribute__((overloadable)) convert_char8_sat(uchar8);
11229 char8 const_func __attribute__((overloadable)) convert_char8_rte(short8);
11230 char8 const_func __attribute__((overloadable)) convert_char8_sat_rte(short8);
11231 char8 const_func __attribute__((overloadable)) convert_char8_rtz(short8);
11232 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtz(short8);
11233 char8 const_func __attribute__((overloadable)) convert_char8_rtp(short8);
11234 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtp(short8);
11235 char8 const_func __attribute__((overloadable)) convert_char8_rtn(short8);
```

```
11236 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtn(short8);
11237 char8 const_func __attribute__((overloadable)) convert_char8(short8);
11238 char8 const_func __attribute__((overloadable)) convert_char8_sat(short8);
11239 char8 const_func __attribute__((overloadable)) convert_char8_rte(ushort8);
11240 char8 const_func __attribute__((overloadable)) convert_char8_sat_rte(ushort8);
11241 char8 const_func __attribute__((overloadable)) convert_char8_rtz(ushort8);
11242 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtz(ushort8);
11243 char8 const_func __attribute__((overloadable)) convert_char8_rtp(ushort8);
11244 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtp(ushort8);
11245 char8 const_func __attribute__((overloadable)) convert_char8_rtn(ushort8);
11246 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtn(ushort8);
11247 char8 const_func __attribute__((overloadable)) convert_char8(ushort8);
11248 char8 const_func __attribute__((overloadable)) convert_char8_sat(ushort8);
11249 char8 const_func __attribute__((overloadable)) convert_char8_rte(int8);
11250 char8 const_func __attribute__((overloadable)) convert_char8_sat_rte(int8);
11251 char8 const_func __attribute__((overloadable)) convert_char8_rtz(int8);
11252 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtz(int8);
11253 char8 const_func __attribute__((overloadable)) convert_char8_rtp(int8);
11254 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtp(int8);
11255 char8 const_func __attribute__((overloadable)) convert_char8_rtn(int8);
11256 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtn(int8);
11257 char8 const_func __attribute__((overloadable)) convert_char8(int8);
11258 char8 const_func __attribute__((overloadable)) convert_char8_sat(int8);
11259 char8 const_func __attribute__((overloadable)) convert_char8_rte(uint8);
11260 char8 const_func __attribute__((overloadable)) convert_char8_sat_rte(uint8);
11261 char8 const_func __attribute__((overloadable)) convert_char8_rtz(uint8);
11262 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtz(uint8);
11263 char8 const_func __attribute__((overloadable)) convert_char8_rtp(uint8);
11264 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtp(uint8);
11265 char8 const_func __attribute__((overloadable)) convert_char8_rtn(uint8);
11266 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtn(uint8);
11267 char8 const_func __attribute__((overloadable)) convert_char8(uint8);
11268 char8 const_func __attribute__((overloadable)) convert_char8_sat(uint8);
11269 char8 const_func __attribute__((overloadable)) convert_char8_rte(long8);
11270 char8 const_func __attribute__((overloadable)) convert_char8_sat_rte(long8);
11271 char8 const_func __attribute__((overloadable)) convert_char8_rtz(long8);
11272 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtz(long8);
11273 char8 const_func __attribute__((overloadable)) convert_char8_rtp(long8);
11274 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtp(long8);
11275 char8 const_func __attribute__((overloadable)) convert_char8_rtn(long8);
11276 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtn(long8);
11277 char8 const_func __attribute__((overloadable)) convert_char8(long8);
11278 char8 const_func __attribute__((overloadable)) convert_char8_sat(long8);
11279 char8 const_func __attribute__((overloadable)) convert_char8_rte(ulong8);
11280 char8 const_func __attribute__((overloadable)) convert_char8_sat_rte(ulong8);
11281 char8 const_func __attribute__((overloadable)) convert_char8_rtz(ulong8);
11282 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtz(ulong8);
11283 char8 const_func __attribute__((overloadable)) convert_char8_rtp(ulong8);
11284 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtp(ulong8);
11285 char8 const_func __attribute__((overloadable)) convert_char8_rtn(ulong8);
11286 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtn(ulong8);
11287 char8 const_func __attribute__((overloadable)) convert_char8(ulong8);
11288 char8 const_func __attribute__((overloadable)) convert_char8_sat(ulong8);
11289 char8 const_func __attribute__((overloadable)) convert_char8_rte(float8);
11290 char8 const_func __attribute__((overloadable)) convert_char8_sat_rte(float8);
11291 char8 const_func __attribute__((overloadable)) convert_char8_rtz(float8);
11292 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtz(float8);
11293 char8 const_func __attribute__((overloadable)) convert_char8_rtp(float8);
11294 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtp(float8);
11295 char8 const_func __attribute__((overloadable)) convert_char8_rtn(float8);
11296 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtn(float8);
11297 char8 const_func __attribute__((overloadable)) convert_char8(float8);
11298 char8 const_func __attribute__((overloadable)) convert_char8_sat(float8);
11299 char8 const_func __attribute__((overloadable)) convert_char8_rte(double8);
11300 char8 const_func __attribute__((overloadable)) convert_char8_sat_rte(double8);
11301 char8 const_func __attribute__((overloadable)) convert_char8_rtz(double8);
11302 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtz(double8);
11303 char8 const_func __attribute__((overloadable)) convert_char8_rtp(double8);
11304 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtp(double8);
11305 char8 const_func __attribute__((overloadable)) convert_char8_rtn(double8);
11306 char8 const_func __attribute__((overloadable)) convert_char8_sat_rtn(double8);
11307 char8 const_func __attribute__((overloadable)) convert_char8(double8);
11308 char8 const_func __attribute__((overloadable)) convert_char8_sat(double8);
11309
11310 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rte(char8);
11311 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rte(char8);
11312 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtz(char8);
11313 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtz(char8);
11314 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtp(char8);
11315 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtp(char8);
11316 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtn(char8);
11317 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtn(char8);
11318 uchar8 const_func __attribute__((overloadable)) convert_uchar8(char8);
11319 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat(char8);
11320 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rte(uchar8);
11321 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rte(uchar8);
11322 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtz(uchar8);
```

```
11323 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtz(uchar8);
11324 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtp(uchar8);
11325 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtp(uchar8);
11326 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtn(uchar8);
11327 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtn(uchar8);
11328 uchar8 const_func __attribute__((overloadable)) convert_uchar8(uchar8);
11329 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat(uchar8);
11330 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rte(short8);
11331 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rte(short8);
11332 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtz(short8);
11333 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtz(short8);
11334 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtp(short8);
11335 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtp(short8);
11336 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtn(short8);
11337 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtn(short8);
11338 uchar8 const_func __attribute__((overloadable)) convert_uchar8(short8);
11339 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat(short8);
11340 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rte(ushort8);
11341 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rte(ushort8);
11342 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtz(ushort8);
11343 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtz(ushort8);
11344 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtp(ushort8);
11345 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtp(ushort8);
11346 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtn(ushort8);
11347 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtn(ushort8);
11348 uchar8 const_func __attribute__((overloadable)) convert_uchar8(ushort8);
11349 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat(ushort8);
11350 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rte(int8);
11351 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rte(int8);
11352 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtz(int8);
11353 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtz(int8);
11354 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtp(int8);
11355 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtp(int8);
11356 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtn(int8);
11357 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtn(int8);
11358 uchar8 const_func __attribute__((overloadable)) convert_uchar8(int8);
11359 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat(int8);
11360 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rte(uint8);
11361 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rte(uint8);
11362 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtz(uint8);
11363 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtz(uint8);
11364 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtp(uint8);
11365 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtp(uint8);
11366 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtn(uint8);
11367 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtn(uint8);
11368 uchar8 const_func __attribute__((overloadable)) convert_uchar8(uint8);
11369 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat(uint8);
11370 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rte(long8);
11371 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rte(long8);
11372 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtz(long8);
11373 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtz(long8);
11374 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtp(long8);
11375 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtp(long8);
11376 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtn(long8);
11377 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtn(long8);
11378 uchar8 const_func __attribute__((overloadable)) convert_uchar8(long8);
11379 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat(long8);
11380 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rte(ulong8);
11381 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rte(ulong8);
11382 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtz(ulong8);
11383 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtz(ulong8);
11384 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtp(ulong8);
11385 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtp(ulong8);
11386 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtn(ulong8);
11387 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtn(ulong8);
11388 uchar8 const_func __attribute__((overloadable)) convert_uchar8(ulong8);
11389 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat(ulong8);
11390 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rte(float8);
11391 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rte(float8);
11392 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtz(float8);
11393 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtz(float8);
11394 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtp(float8);
11395 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtp(float8);
11396 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtn(float8);
11397 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtn(float8);
11398 uchar8 const_func __attribute__((overloadable)) convert_uchar8(float8);
11399 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat(float8);
11400 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rte(double8);
11401 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rte(double8);
11402 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtz(double8);
11403 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtz(double8);
11404 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtp(double8);
11405 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtp(double8);
11406 uchar8 const_func __attribute__((overloadable)) convert_uchar8_rtn(double8);
11407 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat_rtn(double8);
11408 uchar8 const_func __attribute__((overloadable)) convert_uchar8(double8);
11409 uchar8 const_func __attribute__((overloadable)) convert_uchar8_sat(double8);
```

```
11410
11411 short8 const_func __attribute__((overloadable)) convert_short8_rte(char8);
11412 short8 const_func __attribute__((overloadable)) convert_short8_sat_rte(char8);
11413 short8 const_func __attribute__((overloadable)) convert_short8_rtz(char8);
11414 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtz(char8);
11415 short8 const_func __attribute__((overloadable)) convert_short8_rtp(char8);
11416 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtp(char8);
11417 short8 const_func __attribute__((overloadable)) convert_short8_rtn(char8);
11418 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtn(char8);
11419 short8 const_func __attribute__((overloadable)) convert_short8(char8);
11420 short8 const_func __attribute__((overloadable)) convert_short8_sat(char8);
11421 short8 const_func __attribute__((overloadable)) convert_short8_rte(uchar8);
11422 short8 const_func __attribute__((overloadable)) convert_short8_sat_rte(uchar8);
11423 short8 const_func __attribute__((overloadable)) convert_short8_rtz(uchar8);
11424 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtz(uchar8);
11425 short8 const_func __attribute__((overloadable)) convert_short8_rtp(uchar8);
11426 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtp(uchar8);
11427 short8 const_func __attribute__((overloadable)) convert_short8_rtn(uchar8);
11428 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtn(uchar8);
11429 short8 const_func __attribute__((overloadable)) convert_short8(uchar8);
11430 short8 const_func __attribute__((overloadable)) convert_short8_sat(uchar8);
11431 short8 const_func __attribute__((overloadable)) convert_short8_rte(short8);
11432 short8 const_func __attribute__((overloadable)) convert_short8_sat_rte(short8);
11433 short8 const_func __attribute__((overloadable)) convert_short8_rtz(short8);
11434 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtz(short8);
11435 short8 const_func __attribute__((overloadable)) convert_short8_rtp(short8);
11436 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtp(short8);
11437 short8 const_func __attribute__((overloadable)) convert_short8_rtn(short8);
11438 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtn(short8);
11439 short8 const_func __attribute__((overloadable)) convert_short8(short8);
11440 short8 const_func __attribute__((overloadable)) convert_short8_sat(short8);
11441 short8 const_func __attribute__((overloadable)) convert_short8_rte(ushort8);
11442 short8 const_func __attribute__((overloadable)) convert_short8_sat_rte(ushort8);
11443 short8 const_func __attribute__((overloadable)) convert_short8_rtz(ushort8);
11444 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtz(ushort8);
11445 short8 const_func __attribute__((overloadable)) convert_short8_rtp(ushort8);
11446 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtp(ushort8);
11447 short8 const_func __attribute__((overloadable)) convert_short8_rtn(ushort8);
11448 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtn(ushort8);
11449 short8 const_func __attribute__((overloadable)) convert_short8(ushort8);
11450 short8 const_func __attribute__((overloadable)) convert_short8_sat(ushort8);
11451 short8 const_func __attribute__((overloadable)) convert_short8_rte(int8);
11452 short8 const_func __attribute__((overloadable)) convert_short8_sat_rte(int8);
11453 short8 const_func __attribute__((overloadable)) convert_short8_rtz(int8);
11454 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtz(int8);
11455 short8 const_func __attribute__((overloadable)) convert_short8_rtp(int8);
11456 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtp(int8);
11457 short8 const_func __attribute__((overloadable)) convert_short8_rtn(int8);
11458 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtn(int8);
11459 short8 const_func __attribute__((overloadable)) convert_short8(int8);
11460 short8 const_func __attribute__((overloadable)) convert_short8_sat(int8);
11461 short8 const_func __attribute__((overloadable)) convert_short8_rte(uint8);
11462 short8 const_func __attribute__((overloadable)) convert_short8_sat_rte(uint8);
11463 short8 const_func __attribute__((overloadable)) convert_short8_rtz(uint8);
11464 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtz(uint8);
11465 short8 const_func __attribute__((overloadable)) convert_short8_rtp(uint8);
11466 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtp(uint8);
11467 short8 const_func __attribute__((overloadable)) convert_short8_rtn(uint8);
11468 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtn(uint8);
11469 short8 const_func __attribute__((overloadable)) convert_short8(uint8);
11470 short8 const_func __attribute__((overloadable)) convert_short8_sat(uint8);
11471 short8 const_func __attribute__((overloadable)) convert_short8_rte(long8);
11472 short8 const_func __attribute__((overloadable)) convert_short8_sat_rte(long8);
11473 short8 const_func __attribute__((overloadable)) convert_short8_rtz(long8);
11474 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtz(long8);
11475 short8 const_func __attribute__((overloadable)) convert_short8_rtp(long8);
11476 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtp(long8);
11477 short8 const_func __attribute__((overloadable)) convert_short8_rtn(long8);
11478 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtn(long8);
11479 short8 const_func __attribute__((overloadable)) convert_short8(long8);
11480 short8 const_func __attribute__((overloadable)) convert_short8_sat(long8);
11481 short8 const_func __attribute__((overloadable)) convert_short8_rte(ulong8);
11482 short8 const_func __attribute__((overloadable)) convert_short8_sat_rte(ulong8);
11483 short8 const_func __attribute__((overloadable)) convert_short8_rtz(ulong8);
11484 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtz(ulong8);
11485 short8 const_func __attribute__((overloadable)) convert_short8_rtp(ulong8);
11486 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtp(ulong8);
11487 short8 const_func __attribute__((overloadable)) convert_short8_rtn(ulong8);
11488 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtn(ulong8);
11489 short8 const_func __attribute__((overloadable)) convert_short8(ulong8);
11490 short8 const_func __attribute__((overloadable)) convert_short8_sat(ulong8);
11491 short8 const_func __attribute__((overloadable)) convert_short8_rte(float8);
11492 short8 const_func __attribute__((overloadable)) convert_short8_sat_rte(float8);
11493 short8 const_func __attribute__((overloadable)) convert_short8_rtz(float8);
11494 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtz(float8);
11495 short8 const_func __attribute__((overloadable)) convert_short8_rtp(float8);
11496 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtp(float8);
```

```
11497 short8 const_func __attribute__((overloadable)) convert_short8_rtn(float8);
11498 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtn(float8);
11499 short8 const_func __attribute__((overloadable)) convert_short8(float8);
11500 short8 const_func __attribute__((overloadable)) convert_short8_sat(float8);
11501 short8 const_func __attribute__((overloadable)) convert_short8_rte(double8);
11502 short8 const_func __attribute__((overloadable)) convert_short8_sat_rte(double8);
11503 short8 const_func __attribute__((overloadable)) convert_short8_rtz(double8);
11504 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtz(double8);
11505 short8 const_func __attribute__((overloadable)) convert_short8_rtp(double8);
11506 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtp(double8);
11507 short8 const_func __attribute__((overloadable)) convert_short8_rtn(double8);
11508 short8 const_func __attribute__((overloadable)) convert_short8_sat_rtn(double8);
11509 short8 const_func __attribute__((overloadable)) convert_short8(double8);
11510 short8 const_func __attribute__((overloadable)) convert_short8_sat(double8);
11511
11512 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rte(char8);
11513 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rte(char8);
11514 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtz(char8);
11515 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtz(char8);
11516 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtp(char8);
11517 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtp(char8);
11518 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtn(char8);
11519 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtn(char8);
11520 ushort8 const_func __attribute__((overloadable)) convert_ushort8(char8);
11521 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat(char8);
11522 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rte(uchar8);
11523 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rte(uchar8);
11524 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtz(uchar8);
11525 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtz(uchar8);
11526 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtp(uchar8);
11527 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtp(uchar8);
11528 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtn(uchar8);
11529 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtn(uchar8);
11530 ushort8 const_func __attribute__((overloadable)) convert_ushort8(uchar8);
11531 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat(uchar8);
11532 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rte(short8);
11533 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rte(short8);
11534 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtz(short8);
11535 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtz(short8);
11536 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtp(short8);
11537 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtp(short8);
11538 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtn(short8);
11539 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtn(short8);
11540 ushort8 const_func __attribute__((overloadable)) convert_ushort8(short8);
11541 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat(short8);
11542 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rte(ushort8);
11543 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rte(ushort8);
11544 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtz(ushort8);
11545 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtz(ushort8);
11546 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtp(ushort8);
11547 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtp(ushort8);
11548 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtn(ushort8);
11549 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtn(ushort8);
11550 ushort8 const_func __attribute__((overloadable)) convert_ushort8(ushort8);
11551 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat(ushort8);
11552 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rte(int8);
11553 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rte(int8);
11554 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtz(int8);
11555 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtz(int8);
11556 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtp(int8);
11557 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtp(int8);
11558 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtn(int8);
11559 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtn(int8);
11560 ushort8 const_func __attribute__((overloadable)) convert_ushort8(int8);
11561 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat(int8);
11562 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rte(uint8);
11563 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rte(uint8);
11564 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtz(uint8);
11565 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtz(uint8);
11566 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtp(uint8);
11567 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtp(uint8);
11568 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtn(uint8);
11569 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtn(uint8);
11570 ushort8 const_func __attribute__((overloadable)) convert_ushort8(uint8);
11571 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat(uint8);
11572 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rte(long8);
11573 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rte(long8);
11574 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtz(long8);
11575 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtz(long8);
11576 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtp(long8);
11577 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtp(long8);
11578 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtn(long8);
11579 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtn(long8);
11580 ushort8 const_func __attribute__((overloadable)) convert_ushort8(long8);
11581 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat(long8);
11582 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rte(ulong8);
11583 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rte(ulong8);
```

```
11584 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtz(ulong8);
11585 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtz(ulong8);
11586 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtp(ulong8);
11587 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtp(ulong8);
11588 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtn(ulong8);
11589 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtn(ulong8);
11590 ushort8 const_func __attribute__((overloadable)) convert_ushort8(ulong8);
11591 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat(ulong8);
11592 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rte(float8);
11593 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rte(float8);
11594 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtz(float8);
11595 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtz(float8);
11596 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtp(float8);
11597 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtp(float8);
11598 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtn(float8);
11599 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtn(float8);
11600 ushort8 const_func __attribute__((overloadable)) convert_ushort8(float8);
11601 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat(float8);
11602 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rte(double8);
11603 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rte(double8);
11604 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtz(double8);
11605 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtz(double8);
11606 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtp(double8);
11607 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtp(double8);
11608 ushort8 const_func __attribute__((overloadable)) convert_ushort8_rtn(double8);
11609 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat_rtn(double8);
11610 ushort8 const_func __attribute__((overloadable)) convert_ushort8(double8);
11611 ushort8 const_func __attribute__((overloadable)) convert_ushort8_sat(double8);
11612
11613 int8 const_func __attribute__((overloadable)) convert_int8_rte(char8);
11614 int8 const_func __attribute__((overloadable)) convert_int8_sat_rte(char8);
11615 int8 const_func __attribute__((overloadable)) convert_int8_rtz(char8);
11616 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtz(char8);
11617 int8 const_func __attribute__((overloadable)) convert_int8_rtp(char8);
11618 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtp(char8);
11619 int8 const_func __attribute__((overloadable)) convert_int8_rtn(char8);
11620 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtn(char8);
11621 int8 const_func __attribute__((overloadable)) convert_int8(char8);
11622 int8 const_func __attribute__((overloadable)) convert_int8_sat(char8);
11623 int8 const_func __attribute__((overloadable)) convert_int8_rte(uchar8);
11624 int8 const_func __attribute__((overloadable)) convert_int8_sat_rte(uchar8);
11625 int8 const_func __attribute__((overloadable)) convert_int8_rtz(uchar8);
11626 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtz(uchar8);
11627 int8 const_func __attribute__((overloadable)) convert_int8_rtp(uchar8);
11628 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtp(uchar8);
11629 int8 const_func __attribute__((overloadable)) convert_int8_rtn(uchar8);
11630 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtn(uchar8);
11631 int8 const_func __attribute__((overloadable)) convert_int8(uchar8);
11632 int8 const_func __attribute__((overloadable)) convert_int8_sat(uchar8);
11633 int8 const_func __attribute__((overloadable)) convert_int8_rte(short8);
11634 int8 const_func __attribute__((overloadable)) convert_int8_sat_rte(short8);
11635 int8 const_func __attribute__((overloadable)) convert_int8_rtz(short8);
11636 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtz(short8);
11637 int8 const_func __attribute__((overloadable)) convert_int8_rtp(short8);
11638 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtp(short8);
11639 int8 const_func __attribute__((overloadable)) convert_int8_rtn(short8);
11640 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtn(short8);
11641 int8 const_func __attribute__((overloadable)) convert_int8(short8);
11642 int8 const_func __attribute__((overloadable)) convert_int8_sat(short8);
11643 int8 const_func __attribute__((overloadable)) convert_int8_rte(ushort8);
11644 int8 const_func __attribute__((overloadable)) convert_int8_sat_rte(ushort8);
11645 int8 const_func __attribute__((overloadable)) convert_int8_rtz(ushort8);
11646 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtz(ushort8);
11647 int8 const_func __attribute__((overloadable)) convert_int8_rtp(ushort8);
11648 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtp(ushort8);
11649 int8 const_func __attribute__((overloadable)) convert_int8_rtn(ushort8);
11650 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtn(ushort8);
11651 int8 const_func __attribute__((overloadable)) convert_int8(ushort8);
11652 int8 const_func __attribute__((overloadable)) convert_int8_sat(ushort8);
11653 int8 const_func __attribute__((overloadable)) convert_int8_rte(int8);
11654 int8 const_func __attribute__((overloadable)) convert_int8_sat_rte(int8);
11655 int8 const_func __attribute__((overloadable)) convert_int8_rtz(int8);
11656 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtz(int8);
11657 int8 const_func __attribute__((overloadable)) convert_int8_rtp(int8);
11658 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtp(int8);
11659 int8 const_func __attribute__((overloadable)) convert_int8_rtn(int8);
11660 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtn(int8);
11661 int8 const_func __attribute__((overloadable)) convert_int8(int8);
11662 int8 const_func __attribute__((overloadable)) convert_int8_sat(int8);
11663 int8 const_func __attribute__((overloadable)) convert_int8_rte(uint8);
11664 int8 const_func __attribute__((overloadable)) convert_int8_sat_rte(uint8);
11665 int8 const_func __attribute__((overloadable)) convert_int8_rtz(uint8);
11666 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtz(uint8);
11667 int8 const_func __attribute__((overloadable)) convert_int8_rtp(uint8);
11668 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtp(uint8);
11669 int8 const_func __attribute__((overloadable)) convert_int8_rtn(uint8);
11670 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtn(uint8);
```

```
11671 int8 const_func __attribute__((overloadable)) convert_int8(uint8);
11672 int8 const_func __attribute__((overloadable)) convert_int8_sat(uint8);
11673 int8 const_func __attribute__((overloadable)) convert_int8_rte(long8);
11674 int8 const_func __attribute__((overloadable)) convert_int8_sat_rte(long8);
11675 int8 const_func __attribute__((overloadable)) convert_int8_rtz(long8);
11676 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtz(long8);
11677 int8 const_func __attribute__((overloadable)) convert_int8_rtp(long8);
11678 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtp(long8);
11679 int8 const_func __attribute__((overloadable)) convert_int8_rtn(long8);
11680 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtn(long8);
11681 int8 const_func __attribute__((overloadable)) convert_int8(long8);
11682 int8 const_func __attribute__((overloadable)) convert_int8_sat(long8);
11683 int8 const_func __attribute__((overloadable)) convert_int8_rte(ulong8);
11684 int8 const_func __attribute__((overloadable)) convert_int8_sat_rte(ulong8);
11685 int8 const_func __attribute__((overloadable)) convert_int8_rtz(ulong8);
11686 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtz(ulong8);
11687 int8 const_func __attribute__((overloadable)) convert_int8_rtp(ulong8);
11688 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtp(ulong8);
11689 int8 const_func __attribute__((overloadable)) convert_int8_rtn(ulong8);
11690 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtn(ulong8);
11691 int8 const_func __attribute__((overloadable)) convert_int8(ulong8);
11692 int8 const_func __attribute__((overloadable)) convert_int8_sat(ulong8);
11693 int8 const_func __attribute__((overloadable)) convert_int8_rte(float8);
11694 int8 const_func __attribute__((overloadable)) convert_int8_sat_rte(float8);
11695 int8 const_func __attribute__((overloadable)) convert_int8_rtz(float8);
11696 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtz(float8);
11697 int8 const_func __attribute__((overloadable)) convert_int8_rtp(float8);
11698 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtp(float8);
11699 int8 const_func __attribute__((overloadable)) convert_int8_rtn(float8);
11700 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtn(float8);
11701 int8 const_func __attribute__((overloadable)) convert_int8(float8);
11702 int8 const_func __attribute__((overloadable)) convert_int8_sat(float8);
11703 int8 const_func __attribute__((overloadable)) convert_int8_rte(double8);
11704 int8 const_func __attribute__((overloadable)) convert_int8_sat_rte(double8);
11705 int8 const_func __attribute__((overloadable)) convert_int8_rtz(double8);
11706 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtz(double8);
11707 int8 const_func __attribute__((overloadable)) convert_int8_rtp(double8);
11708 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtp(double8);
11709 int8 const_func __attribute__((overloadable)) convert_int8_rtn(double8);
11710 int8 const_func __attribute__((overloadable)) convert_int8_sat_rtn(double8);
11711 int8 const_func __attribute__((overloadable)) convert_int8(double8);
11712 int8 const_func __attribute__((overloadable)) convert_int8_sat(double8);
11713
11714 uint8 const_func __attribute__((overloadable)) convert_uint8_rte(char8);
11715 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rte(char8);
11716 uint8 const_func __attribute__((overloadable)) convert_uint8_rtz(char8);
11717 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtz(char8);
11718 uint8 const_func __attribute__((overloadable)) convert_uint8_rtp(char8);
11719 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtp(char8);
11720 uint8 const_func __attribute__((overloadable)) convert_uint8_rtn(char8);
11721 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtn(char8);
11722 uint8 const_func __attribute__((overloadable)) convert_uint8(char8);
11723 uint8 const_func __attribute__((overloadable)) convert_uint8_sat(char8);
11724 uint8 const_func __attribute__((overloadable)) convert_uint8_rte(uchar8);
11725 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rte(uchar8);
11726 uint8 const_func __attribute__((overloadable)) convert_uint8_rtz(uchar8);
11727 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtz(uchar8);
11728 uint8 const_func __attribute__((overloadable)) convert_uint8_rtp(uchar8);
11729 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtp(uchar8);
11730 uint8 const_func __attribute__((overloadable)) convert_uint8_rtn(uchar8);
11731 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtn(uchar8);
11732 uint8 const_func __attribute__((overloadable)) convert_uint8(uchar8);
11733 uint8 const_func __attribute__((overloadable)) convert_uint8_sat(uchar8);
11734 uint8 const_func __attribute__((overloadable)) convert_uint8_rte(short8);
11735 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rte(short8);
11736 uint8 const_func __attribute__((overloadable)) convert_uint8_rtz(short8);
11737 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtz(short8);
11738 uint8 const_func __attribute__((overloadable)) convert_uint8_rtp(short8);
11739 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtp(short8);
11740 uint8 const_func __attribute__((overloadable)) convert_uint8_rtn(short8);
11741 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtn(short8);
11742 uint8 const_func __attribute__((overloadable)) convert_uint8(short8);
11743 uint8 const_func __attribute__((overloadable)) convert_uint8_sat(short8);
11744 uint8 const_func __attribute__((overloadable)) convert_uint8_rte(ushort8);
11745 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rte(ushort8);
11746 uint8 const_func __attribute__((overloadable)) convert_uint8_rtz(ushort8);
11747 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtz(ushort8);
11748 uint8 const_func __attribute__((overloadable)) convert_uint8_rtp(ushort8);
11749 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtp(ushort8);
11750 uint8 const_func __attribute__((overloadable)) convert_uint8_rtn(ushort8);
11751 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtn(ushort8);
11752 uint8 const_func __attribute__((overloadable)) convert_uint8(ushort8);
11753 uint8 const_func __attribute__((overloadable)) convert_uint8_sat(ushort8);
11754 uint8 const_func __attribute__((overloadable)) convert_uint8_rte(int8);
11755 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rte(int8);
11756 uint8 const_func __attribute__((overloadable)) convert_uint8_rtz(int8);
11757 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtz(int8);
```

```
11758 uint8 const_func __attribute__((overloadable)) convert_uint8_rtp(int8);
11759 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtp(int8);
11760 uint8 const_func __attribute__((overloadable)) convert_uint8_rtn(int8);
11761 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtn(int8);
11762 uint8 const_func __attribute__((overloadable)) convert_uint8(int8);
11763 uint8 const_func __attribute__((overloadable)) convert_uint8_sat(int8);
11764 uint8 const_func __attribute__((overloadable)) convert_uint8_rte(uint8);
11765 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rte(uint8);
11766 uint8 const_func __attribute__((overloadable)) convert_uint8_rtz(uint8);
11767 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtz(uint8);
11768 uint8 const_func __attribute__((overloadable)) convert_uint8_rtp(uint8);
11769 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtp(uint8);
11770 uint8 const_func __attribute__((overloadable)) convert_uint8_rtn(uint8);
11771 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtn(uint8);
11772 uint8 const_func __attribute__((overloadable)) convert_uint8(uint8);
11773 uint8 const_func __attribute__((overloadable)) convert_uint8_sat(uint8);
11774 uint8 const_func __attribute__((overloadable)) convert_uint8_rte(long8);
11775 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rte(long8);
11776 uint8 const_func __attribute__((overloadable)) convert_uint8_rtz(long8);
11777 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtz(long8);
11778 uint8 const_func __attribute__((overloadable)) convert_uint8_rtp(long8);
11779 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtp(long8);
11780 uint8 const_func __attribute__((overloadable)) convert_uint8_rtn(long8);
11781 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtn(long8);
11782 uint8 const_func __attribute__((overloadable)) convert_uint8(long8);
11783 uint8 const_func __attribute__((overloadable)) convert_uint8_sat(long8);
11784 uint8 const_func __attribute__((overloadable)) convert_uint8_rte(ulong8);
11785 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rte(ulong8);
11786 uint8 const_func __attribute__((overloadable)) convert_uint8_rtz(ulong8);
11787 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtz(ulong8);
11788 uint8 const_func __attribute__((overloadable)) convert_uint8_rtp(ulong8);
11789 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtp(ulong8);
11790 uint8 const_func __attribute__((overloadable)) convert_uint8_rtn(ulong8);
11791 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtn(ulong8);
11792 uint8 const_func __attribute__((overloadable)) convert_uint8(ulong8);
11793 uint8 const_func __attribute__((overloadable)) convert_uint8_sat(ulong8);
11794 uint8 const_func __attribute__((overloadable)) convert_uint8_rte(float8);
11795 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rte(float8);
11796 uint8 const_func __attribute__((overloadable)) convert_uint8_rtz(float8);
11797 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtz(float8);
11798 uint8 const_func __attribute__((overloadable)) convert_uint8_rtp(float8);
11799 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtp(float8);
11800 uint8 const_func __attribute__((overloadable)) convert_uint8_rtn(float8);
11801 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtn(float8);
11802 uint8 const_func __attribute__((overloadable)) convert_uint8(float8);
11803 uint8 const_func __attribute__((overloadable)) convert_uint8_sat(float8);
11804 uint8 const_func __attribute__((overloadable)) convert_uint8_rte(double8);
11805 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rte(double8);
11806 uint8 const_func __attribute__((overloadable)) convert_uint8_rtz(double8);
11807 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtz(double8);
11808 uint8 const_func __attribute__((overloadable)) convert_uint8_rtp(double8);
11809 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtp(double8);
11810 uint8 const_func __attribute__((overloadable)) convert_uint8_rtn(double8);
11811 uint8 const_func __attribute__((overloadable)) convert_uint8_sat_rtn(double8);
11812 uint8 const_func __attribute__((overloadable)) convert_uint8(double8);
11813 uint8 const_func __attribute__((overloadable)) convert_uint8_sat(double8);
11814 long8 const_func __attribute__((overloadable)) convert_long8_rte(char8);
11815 long8 const_func __attribute__((overloadable)) convert_long8_sat_rte(char8);
11816 long8 const_func __attribute__((overloadable)) convert_long8_rtz(char8);
11817 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtz(char8);
11818 long8 const_func __attribute__((overloadable)) convert_long8_rtp(char8);
11819 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtp(char8);
11820 long8 const_func __attribute__((overloadable)) convert_long8_rtn(char8);
11821 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtn(char8);
11822 long8 const_func __attribute__((overloadable)) convert_long8(char8);
11823 long8 const_func __attribute__((overloadable)) convert_long8_sat(char8);
11824 long8 const_func __attribute__((overloadable)) convert_long8_rte(uchar8);
11825 long8 const_func __attribute__((overloadable)) convert_long8_sat_rte(uchar8);
11826 long8 const_func __attribute__((overloadable)) convert_long8_rtz(uchar8);
11827 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtz(uchar8);
11828 long8 const_func __attribute__((overloadable)) convert_long8_rtp(uchar8);
11829 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtp(uchar8);
11830 long8 const_func __attribute__((overloadable)) convert_long8_rtn(uchar8);
11831 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtn(uchar8);
11832 long8 const_func __attribute__((overloadable)) convert_long8(uchar8);
11833 long8 const_func __attribute__((overloadable)) convert_long8_sat(uchar8);
11834 long8 const_func __attribute__((overloadable)) convert_long8_rte(short8);
11835 long8 const_func __attribute__((overloadable)) convert_long8_sat_rte(short8);
11836 long8 const_func __attribute__((overloadable)) convert_long8_rtz(short8);
11837 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtz(short8);
11838 long8 const_func __attribute__((overloadable)) convert_long8_rtp(short8);
11839 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtp(short8);
11840 long8 const_func __attribute__((overloadable)) convert_long8_rtn(short8);
11841 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtn(short8);
11842 long8 const_func __attribute__((overloadable)) convert_long8(short8);
11843 long8 const_func __attribute__((overloadable)) convert_long8_sat(short8);
11844 long8 const_func __attribute__((overloadable)) convert_long8_rte(ushort8);
```

```
11845 long8 const_func __attribute__((overloadable)) convert_long8_sat_rte(ushort8);
11846 long8 const_func __attribute__((overloadable)) convert_long8_rtz(ushort8);
11847 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtz(ushort8);
11848 long8 const_func __attribute__((overloadable)) convert_long8_rtp(ushort8);
11849 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtp(ushort8);
11850 long8 const_func __attribute__((overloadable)) convert_long8_rtn(ushort8);
11851 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtn(ushort8);
11852 long8 const_func __attribute__((overloadable)) convert_long8(ushort8);
11853 long8 const_func __attribute__((overloadable)) convert_long8_sat(ushort8);
11854 long8 const_func __attribute__((overloadable)) convert_long8_rte(int8);
11855 long8 const_func __attribute__((overloadable)) convert_long8_sat_rte(int8);
11856 long8 const_func __attribute__((overloadable)) convert_long8_rtz(int8);
11857 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtz(int8);
11858 long8 const_func __attribute__((overloadable)) convert_long8_rtp(int8);
11859 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtp(int8);
11860 long8 const_func __attribute__((overloadable)) convert_long8_rtn(int8);
11861 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtn(int8);
11862 long8 const_func __attribute__((overloadable)) convert_long8(int8);
11863 long8 const_func __attribute__((overloadable)) convert_long8_sat(int8);
11864 long8 const_func __attribute__((overloadable)) convert_long8_rte(uint8);
11865 long8 const_func __attribute__((overloadable)) convert_long8_sat_rte(uint8);
11866 long8 const_func __attribute__((overloadable)) convert_long8_rtz(uint8);
11867 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtz(uint8);
11868 long8 const_func __attribute__((overloadable)) convert_long8_rtp(uint8);
11869 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtp(uint8);
11870 long8 const_func __attribute__((overloadable)) convert_long8_rtn(uint8);
11871 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtn(uint8);
11872 long8 const_func __attribute__((overloadable)) convert_long8(uint8);
11873 long8 const_func __attribute__((overloadable)) convert_long8_sat(uint8);
11874 long8 const_func __attribute__((overloadable)) convert_long8_rte(long8);
11875 long8 const_func __attribute__((overloadable)) convert_long8_sat_rte(long8);
11876 long8 const_func __attribute__((overloadable)) convert_long8_rtz(long8);
11877 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtz(long8);
11878 long8 const_func __attribute__((overloadable)) convert_long8_rtp(long8);
11879 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtp(long8);
11880 long8 const_func __attribute__((overloadable)) convert_long8_rtn(long8);
11881 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtn(long8);
11882 long8 const_func __attribute__((overloadable)) convert_long8(long8);
11883 long8 const_func __attribute__((overloadable)) convert_long8_sat(long8);
11884 long8 const_func __attribute__((overloadable)) convert_long8_rte(ulong8);
11885 long8 const_func __attribute__((overloadable)) convert_long8_sat_rte(ulong8);
11886 long8 const_func __attribute__((overloadable)) convert_long8_rtz(ulong8);
11887 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtz(ulong8);
11888 long8 const_func __attribute__((overloadable)) convert_long8_rtp(ulong8);
11889 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtp(ulong8);
11890 long8 const_func __attribute__((overloadable)) convert_long8_rtn(ulong8);
11891 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtn(ulong8);
11892 long8 const_func __attribute__((overloadable)) convert_long8(ulong8);
11893 long8 const_func __attribute__((overloadable)) convert_long8_sat(ulong8);
11894 long8 const_func __attribute__((overloadable)) convert_long8_rte(float8);
11895 long8 const_func __attribute__((overloadable)) convert_long8_sat_rte(float8);
11896 long8 const_func __attribute__((overloadable)) convert_long8_rtz(float8);
11897 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtz(float8);
11898 long8 const_func __attribute__((overloadable)) convert_long8_rtp(float8);
11899 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtp(float8);
11900 long8 const_func __attribute__((overloadable)) convert_long8_rtn(float8);
11901 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtn(float8);
11902 long8 const_func __attribute__((overloadable)) convert_long8(float8);
11903 long8 const_func __attribute__((overloadable)) convert_long8_sat(float8);
11904 long8 const_func __attribute__((overloadable)) convert_long8_rte(double8);
11905 long8 const_func __attribute__((overloadable)) convert_long8_sat_rte(double8);
11906 long8 const_func __attribute__((overloadable)) convert_long8_rtz(double8);
11907 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtz(double8);
11908 long8 const_func __attribute__((overloadable)) convert_long8_rtp(double8);
11909 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtp(double8);
11910 long8 const_func __attribute__((overloadable)) convert_long8_rtn(double8);
11911 long8 const_func __attribute__((overloadable)) convert_long8_sat_rtn(double8);
11912 long8 const_func __attribute__((overloadable)) convert_long8(double8);
11913 long8 const_func __attribute__((overloadable)) convert_long8_sat(double8);
11914
11915 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rte(char8);
11916 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rte(char8);
11917 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtz(char8);
11918 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtz(char8);
11919 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtp(char8);
11920 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtp(char8);
11921 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtn(char8);
11922 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtn(char8);
11923 ulong8 const_func __attribute__((overloadable)) convert_ulong8(char8);
11924 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat(char8);
11925 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rte(uchar8);
11926 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rte(uchar8);
11927 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtz(uchar8);
11928 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtz(uchar8);
11929 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtp(uchar8);
11930 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtp(uchar8);
11931 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtn(uchar8);
```

```
11932 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtn(uchar8);
11933 ulong8 const_func __attribute__((overloadable)) convert_ulong8(uchar8);
11934 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat(uchar8);
11935 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rte(short8);
11936 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rte(short8);
11937 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtz(short8);
11938 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtz(short8);
11939 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtp(short8);
11940 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtp(short8);
11941 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtn(short8);
11942 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtn(short8);
11943 ulong8 const_func __attribute__((overloadable)) convert_ulong8(short8);
11944 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat(short8);
11945 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rte(ushort8);
11946 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rte(ushort8);
11947 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtz(ushort8);
11948 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtz(ushort8);
11949 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtp(ushort8);
11950 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtp(ushort8);
11951 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtn(ushort8);
11952 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtn(ushort8);
11953 ulong8 const_func __attribute__((overloadable)) convert_ulong8(ushort8);
11954 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat(ushort8);
11955 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rte(int8);
11956 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rte(int8);
11957 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtz(int8);
11958 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtz(int8);
11959 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtp(int8);
11960 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtp(int8);
11961 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtn(int8);
11962 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtn(int8);
11963 ulong8 const_func __attribute__((overloadable)) convert_ulong8(int8);
11964 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat(int8);
11965 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rte(uint8);
11966 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rte(uint8);
11967 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtz(uint8);
11968 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtz(uint8);
11969 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtp(uint8);
11970 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtp(uint8);
11971 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtn(uint8);
11972 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtn(uint8);
11973 ulong8 const_func __attribute__((overloadable)) convert_ulong8(uint8);
11974 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat(uint8);
11975 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rte(long8);
11976 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rte(long8);
11977 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtz(long8);
11978 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtz(long8);
11979 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtp(long8);
11980 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtp(long8);
11981 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtn(long8);
11982 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtn(long8);
11983 ulong8 const_func __attribute__((overloadable)) convert_ulong8(long8);
11984 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat(long8);
11985 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rte(ulong8);
11986 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rte(ulong8);
11987 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtz(ulong8);
11988 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtz(ulong8);
11989 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtp(ulong8);
11990 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtp(ulong8);
11991 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtn(ulong8);
11992 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtn(ulong8);
11993 ulong8 const_func __attribute__((overloadable)) convert_ulong8(ulong8);
11994 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat(ulong8);
11995 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rte(float8);
11996 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rte(float8);
11997 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtz(float8);
11998 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtz(float8);
11999 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtp(float8);
12000 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtp(float8);
12001 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtn(float8);
12002 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtn(float8);
12003 ulong8 const_func __attribute__((overloadable)) convert_ulong8(float8);
12004 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat(float8);
12005 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rte(double8);
12006 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rte(double8);
12007 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtz(double8);
12008 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtz(double8);
12009 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtp(double8);
12010 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtp(double8);
12011 ulong8 const_func __attribute__((overloadable)) convert_ulong8_rtn(double8);
12012 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat_rtn(double8);
12013 ulong8 const_func __attribute__((overloadable)) convert_ulong8(double8);
12014 ulong8 const_func __attribute__((overloadable)) convert_ulong8_sat(double8);
12015
12016 float8 const_func __attribute__((overloadable)) convert_float8_rte(char8);
12017 float8 const_func __attribute__((overloadable)) convert_float8_rtz(char8);
12018 float8 const_func __attribute__((overloadable)) convert_float8_rtp(char8);
```

```
12019 float8 const_func __attribute__((overloadable)) convert_float8_rtn(char8);
12020 float8 const_func __attribute__((overloadable)) convert_float8(char8);
12021 float8 const_func __attribute__((overloadable)) convert_float8_rte(uchar8);
12022 float8 const_func __attribute__((overloadable)) convert_float8_rtz(uchar8);
12023 float8 const_func __attribute__((overloadable)) convert_float8_rtp(uchar8);
12024 float8 const_func __attribute__((overloadable)) convert_float8_rtn(uchar8);
12025 float8 const_func __attribute__((overloadable)) convert_float8(uchar8);
12026 float8 const_func __attribute__((overloadable)) convert_float8_rte(short8);
12027 float8 const_func __attribute__((overloadable)) convert_float8_rtz(short8);
12028 float8 const_func __attribute__((overloadable)) convert_float8_rtp(short8);
12029 float8 const_func __attribute__((overloadable)) convert_float8_rtn(short8);
12030 float8 const_func __attribute__((overloadable)) convert_float8(short8);
12031 float8 const_func __attribute__((overloadable)) convert_float8_rte(ushort8);
12032 float8 const_func __attribute__((overloadable)) convert_float8_rtz(ushort8);
12033 float8 const_func __attribute__((overloadable)) convert_float8_rtp(ushort8);
12034 float8 const_func __attribute__((overloadable)) convert_float8_rtn(ushort8);
12035 float8 const_func __attribute__((overloadable)) convert_float8(ushort8);
12036 float8 const_func __attribute__((overloadable)) convert_float8_rte(int8);
12037 float8 const_func __attribute__((overloadable)) convert_float8_rtz(int8);
12038 float8 const_func __attribute__((overloadable)) convert_float8_rtp(int8);
12039 float8 const_func __attribute__((overloadable)) convert_float8_rtn(int8);
12040 float8 const_func __attribute__((overloadable)) convert_float8(int8);
12041 float8 const_func __attribute__((overloadable)) convert_float8_rte(uint8);
12042 float8 const_func __attribute__((overloadable)) convert_float8_rtz(uint8);
12043 float8 const_func __attribute__((overloadable)) convert_float8_rtp(uint8);
12044 float8 const_func __attribute__((overloadable)) convert_float8_rtn(uint8);
12045 float8 const_func __attribute__((overloadable)) convert_float8(uint8);
12046 float8 const_func __attribute__((overloadable)) convert_float8_rte(long8);
12047 float8 const_func __attribute__((overloadable)) convert_float8_rtz(long8);
12048 float8 const_func __attribute__((overloadable)) convert_float8_rtp(long8);
12049 float8 const_func __attribute__((overloadable)) convert_float8_rtn(long8);
12050 float8 const_func __attribute__((overloadable)) convert_float8(long8);
12051 float8 const_func __attribute__((overloadable)) convert_float8_rte(ulong8);
12052 float8 const_func __attribute__((overloadable)) convert_float8_rtz(ulong8);
12053 float8 const_func __attribute__((overloadable)) convert_float8_rtp(ulong8);
12054 float8 const_func __attribute__((overloadable)) convert_float8_rtn(ulong8);
12055 float8 const_func __attribute__((overloadable)) convert_float8(ulong8);
12056 float8 const_func __attribute__((overloadable)) convert_float8_rte(float8);
12057 float8 const_func __attribute__((overloadable)) convert_float8_rtz(float8);
12058 float8 const_func __attribute__((overloadable)) convert_float8_rtp(float8);
12059 float8 const_func __attribute__((overloadable)) convert_float8_rtn(float8);
12060 float8 const_func __attribute__((overloadable)) convert_float8(float8);
12061 float8 const_func __attribute__((overloadable)) convert_float8_rte(double8);
12062 float8 const_func __attribute__((overloadable)) convert_float8_rtz(double8);
12063 float8 const_func __attribute__((overloadable)) convert_float8_rtp(double8);
12064 float8 const_func __attribute__((overloadable)) convert_float8_rtn(double8);
12065 float8 const_func __attribute__((overloadable)) convert_float8(double8);
12066
12067 double8 const_func __attribute__((overloadable)) convert_double8_rte(char8);
12068 double8 const_func __attribute__((overloadable)) convert_double8_rtz(char8);
12069 double8 const_func __attribute__((overloadable)) convert_double8_rtp(char8);
12070 double8 const_func __attribute__((overloadable)) convert_double8_rtn(char8);
12071 double8 const_func __attribute__((overloadable)) convert_double8(char8);
12072 double8 const_func __attribute__((overloadable)) convert_double8_rte(uchar8);
12073 double8 const_func __attribute__((overloadable)) convert_double8_rtz(uchar8);
12074 double8 const_func __attribute__((overloadable)) convert_double8_rtp(uchar8);
12075 double8 const_func __attribute__((overloadable)) convert_double8_rtn(uchar8);
12076 double8 const_func __attribute__((overloadable)) convert_double8(uchar8);
12077 double8 const_func __attribute__((overloadable)) convert_double8_rte(short8);
12078 double8 const_func __attribute__((overloadable)) convert_double8_rtz(short8);
12079 double8 const_func __attribute__((overloadable)) convert_double8_rtp(short8);
12080 double8 const_func __attribute__((overloadable)) convert_double8_rtn(short8);
12081 double8 const_func __attribute__((overloadable)) convert_double8(short8);
12082 double8 const_func __attribute__((overloadable)) convert_double8_rte(ushort8);
12083 double8 const_func __attribute__((overloadable)) convert_double8_rtz(ushort8);
12084 double8 const_func __attribute__((overloadable)) convert_double8_rtp(ushort8);
12085 double8 const_func __attribute__((overloadable)) convert_double8_rtn(ushort8);
12086 double8 const_func __attribute__((overloadable)) convert_double8(ushort8);
12087 double8 const_func __attribute__((overloadable)) convert_double8_rte(int8);
12088 double8 const_func __attribute__((overloadable)) convert_double8_rtz(int8);
12089 double8 const_func __attribute__((overloadable)) convert_double8_rtp(int8);
12090 double8 const_func __attribute__((overloadable)) convert_double8_rtn(int8);
12091 double8 const_func __attribute__((overloadable)) convert_double8(int8);
12092 double8 const_func __attribute__((overloadable)) convert_double8_rte(uint8);
12093 double8 const_func __attribute__((overloadable)) convert_double8_rtz(uint8);
12094 double8 const_func __attribute__((overloadable)) convert_double8_rtp(uint8);
12095 double8 const_func __attribute__((overloadable)) convert_double8_rtn(uint8);
12096 double8 const_func __attribute__((overloadable)) convert_double8(uint8);
12097 double8 const_func __attribute__((overloadable)) convert_double8_rte(long8);
12098 double8 const_func __attribute__((overloadable)) convert_double8_rtz(long8);
12099 double8 const_func __attribute__((overloadable)) convert_double8_rtp(long8);
12100 double8 const_func __attribute__((overloadable)) convert_double8_rtn(long8);
12101 double8 const_func __attribute__((overloadable)) convert_double8(long8);
12102 double8 const_func __attribute__((overloadable)) convert_double8_rte(ulong8);
12103 double8 const_func __attribute__((overloadable)) convert_double8_rtz(ulong8);
12104 double8 const_func __attribute__((overloadable)) convert_double8_rtp(ulong8);
12105 double8 const_func __attribute__((overloadable)) convert_double8_rtn(ulong8);
```

```
12106 double8 const_func __attribute__((overloadable)) convert_double8(ulong8);
12107 double8 const_func __attribute__((overloadable)) convert_double8_rte(float8);
12108 double8 const_func __attribute__((overloadable)) convert_double8_rtz(float8);
12109 double8 const_func __attribute__((overloadable)) convert_double8_rtp(float8);
12110 double8 const_func __attribute__((overloadable)) convert_double8_rtn(float8);
12111 double8 const_func __attribute__((overloadable)) convert_double8(float8);
12112 double8 const_func __attribute__((overloadable)) convert_double8_rte(double8);
12113 double8 const_func __attribute__((overloadable)) convert_double8_rtz(double8);
12114 double8 const_func __attribute__((overloadable)) convert_double8_rtp(double8);
12115 double8 const_func __attribute__((overloadable)) convert_double8_rtn(double8);
12116 double8 const_func __attribute__((overloadable)) convert_double8(double8);
12117 char16 const_func __attribute__((overloadable)) convert_char16_rte(char16);
12118 char16 const_func __attribute__((overloadable)) convert_char16_sat_rte(char16);
12119 char16 const_func __attribute__((overloadable)) convert_char16_rtz(char16);
12120 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtz(char16);
12121 char16 const_func __attribute__((overloadable)) convert_char16_rtp(char16);
12122 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtp(char16);
12123 char16 const_func __attribute__((overloadable)) convert_char16_rtn(char16);
12124 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtn(char16);
12125 char16 const_func __attribute__((overloadable)) convert_char16(char16);
12126 char16 const_func __attribute__((overloadable)) convert_char16_sat(char16);
12127 char16 const_func __attribute__((overloadable)) convert_char16_rte(uchar16);
12128 char16 const_func __attribute__((overloadable)) convert_char16_sat_rte(uchar16);
12129 char16 const_func __attribute__((overloadable)) convert_char16_rtz(uchar16);
12130 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtz(uchar16);
12131 char16 const_func __attribute__((overloadable)) convert_char16_rtp(uchar16);
12132 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtp(uchar16);
12133 char16 const_func __attribute__((overloadable)) convert_char16_rtn(uchar16);
12134 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtn(uchar16);
12135 char16 const_func __attribute__((overloadable)) convert_char16(uchar16);
12136 char16 const_func __attribute__((overloadable)) convert_char16_sat(uchar16);
12137 char16 const_func __attribute__((overloadable)) convert_char16_rte(short16);
12138 char16 const_func __attribute__((overloadable)) convert_char16_sat_rte(short16);
12139 char16 const_func __attribute__((overloadable)) convert_char16_rtz(short16);
12140 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtz(short16);
12141 char16 const_func __attribute__((overloadable)) convert_char16_rtp(short16);
12142 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtp(short16);
12143 char16 const_func __attribute__((overloadable)) convert_char16_rtn(short16);
12144 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtn(short16);
12145 char16 const_func __attribute__((overloadable)) convert_char16(short16);
12146 char16 const_func __attribute__((overloadable)) convert_char16_sat(short16);
12147 char16 const_func __attribute__((overloadable)) convert_char16_rte(ushort16);
12148 char16 const_func __attribute__((overloadable)) convert_char16_sat_rte(ushort16);
12149 char16 const_func __attribute__((overloadable)) convert_char16_rtz(ushort16);
12150 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtz(ushort16);
12151 char16 const_func __attribute__((overloadable)) convert_char16_rtp(ushort16);
12152 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtp(ushort16);
12153 char16 const_func __attribute__((overloadable)) convert_char16_rtn(ushort16);
12154 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtn(ushort16);
12155 char16 const_func __attribute__((overloadable)) convert_char16(ushort16);
12156 char16 const_func __attribute__((overloadable)) convert_char16_sat(ushort16);
12157 char16 const_func __attribute__((overloadable)) convert_char16_rte(int16);
12158 char16 const_func __attribute__((overloadable)) convert_char16_sat_rte(int16);
12159 char16 const_func __attribute__((overloadable)) convert_char16_rtz(int16);
12160 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtz(int16);
12161 char16 const_func __attribute__((overloadable)) convert_char16_rtp(int16);
12162 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtp(int16);
12163 char16 const_func __attribute__((overloadable)) convert_char16_rtn(int16);
12164 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtn(int16);
12165 char16 const_func __attribute__((overloadable)) convert_char16(int16);
12166 char16 const_func __attribute__((overloadable)) convert_char16_sat(int16);
12167 char16 const_func __attribute__((overloadable)) convert_char16_rte(uint16);
12168 char16 const_func __attribute__((overloadable)) convert_char16_sat_rte(uint16);
12169 char16 const_func __attribute__((overloadable)) convert_char16_rtz(uint16);
12170 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtz(uint16);
12171 char16 const_func __attribute__((overloadable)) convert_char16_rtp(uint16);
12172 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtp(uint16);
12173 char16 const_func __attribute__((overloadable)) convert_char16_rtn(uint16);
12174 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtn(uint16);
12175 char16 const_func __attribute__((overloadable)) convert_char16(uint16);
12176 char16 const_func __attribute__((overloadable)) convert_char16_sat(uint16);
12177 char16 const_func __attribute__((overloadable)) convert_char16_rte(long16);
12178 char16 const_func __attribute__((overloadable)) convert_char16_sat_rte(long16);
12179 char16 const_func __attribute__((overloadable)) convert_char16_rtz(long16);
12180 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtz(long16);
12181 char16 const_func __attribute__((overloadable)) convert_char16_rtp(long16);
12182 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtp(long16);
12183 char16 const_func __attribute__((overloadable)) convert_char16_rtn(long16);
12184 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtn(long16);
12185 char16 const_func __attribute__((overloadable)) convert_char16(long16);
12186 char16 const_func __attribute__((overloadable)) convert_char16_sat(long16);
12187 char16 const_func __attribute__((overloadable)) convert_char16_rte(ulong16);
12188 char16 const_func __attribute__((overloadable)) convert_char16_sat_rte(ulong16);
12189 char16 const_func __attribute__((overloadable)) convert_char16_rtz(ulong16);
12190 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtz(ulong16);
12191 char16 const_func __attribute__((overloadable)) convert_char16_rtp(ulong16);
12192 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtp(ulong16);
```

```
12193 char16 const_func __attribute__((overloadable)) convert_char16_rtn(ulong16);
12194 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtn(ulong16);
12195 char16 const_func __attribute__((overloadable)) convert_char16(ulong16);
12196 char16 const_func __attribute__((overloadable)) convert_char16_sat(ulong16);
12197 char16 const_func __attribute__((overloadable)) convert_char16_rte(float16);
12198 char16 const_func __attribute__((overloadable)) convert_char16_sat_rte(float16);
12199 char16 const_func __attribute__((overloadable)) convert_char16_rtz(float16);
12200 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtz(float16);
12201 char16 const_func __attribute__((overloadable)) convert_char16_rtp(float16);
12202 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtp(float16);
12203 char16 const_func __attribute__((overloadable)) convert_char16_rtn(float16);
12204 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtn(float16);
12205 char16 const_func __attribute__((overloadable)) convert_char16(float16);
12206 char16 const_func __attribute__((overloadable)) convert_char16_sat(float16);
12207 char16 const_func __attribute__((overloadable)) convert_char16_rte(double16);
12208 char16 const_func __attribute__((overloadable)) convert_char16_sat_rte(double16);
12209 char16 const_func __attribute__((overloadable)) convert_char16_rtz(double16);
12210 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtz(double16);
12211 char16 const_func __attribute__((overloadable)) convert_char16_rtp(double16);
12212 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtp(double16);
12213 char16 const_func __attribute__((overloadable)) convert_char16_rtn(double16);
12214 char16 const_func __attribute__((overloadable)) convert_char16_sat_rtn(double16);
12215 char16 const_func __attribute__((overloadable)) convert_char16(double16);
12216 char16 const_func __attribute__((overloadable)) convert_char16_sat(double16);
12217
12218 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rte(char16);
12219 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rte(char16);
12220 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtz(char16);
12221 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtz(char16);
12222 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtp(char16);
12223 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtp(char16);
12224 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtn(char16);
12225 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtn(char16);
12226 uchar16 const_func __attribute__((overloadable)) convert_uchar16(char16);
12227 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat(char16);
12228 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rte(uchar16);
12229 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rte(uchar16);
12230 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtz(uchar16);
12231 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtz(uchar16);
12232 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtp(uchar16);
12233 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtp(uchar16);
12234 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtn(uchar16);
12235 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtn(uchar16);
12236 uchar16 const_func __attribute__((overloadable)) convert_uchar16(uchar16);
12237 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat(uchar16);
12238 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rte(short16);
12239 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rte(short16);
12240 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtz(short16);
12241 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtz(short16);
12242 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtp(short16);
12243 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtp(short16);
12244 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtn(short16);
12245 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtn(short16);
12246 uchar16 const_func __attribute__((overloadable)) convert_uchar16(short16);
12247 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat(short16);
12248 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rte(ushort16);
12249 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rte(ushort16);
12250 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtz(ushort16);
12251 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtz(ushort16);
12252 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtp(ushort16);
12253 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtp(ushort16);
12254 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtn(ushort16);
12255 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtn(ushort16);
12256 uchar16 const_func __attribute__((overloadable)) convert_uchar16(ushort16);
12257 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat(ushort16);
12258 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rte(int16);
12259 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rte(int16);
12260 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtz(int16);
12261 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtz(int16);
12262 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtp(int16);
12263 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtp(int16);
12264 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtn(int16);
12265 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtn(int16);
12266 uchar16 const_func __attribute__((overloadable)) convert_uchar16(int16);
12267 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat(int16);
12268 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rte(uint16);
12269 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rte(uint16);
12270 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtz(uint16);
12271 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtz(uint16);
12272 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtp(uint16);
12273 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtp(uint16);
12274 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtn(uint16);
12275 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtn(uint16);
12276 uchar16 const_func __attribute__((overloadable)) convert_uchar16(uint16);
12277 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat(uint16);
12278 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rte(long16);
12279 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rte(long16);
```

```
12280 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtz(long16);
12281 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtz(long16);
12282 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtp(long16);
12283 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtp(long16);
12284 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtn(long16);
12285 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtn(long16);
12286 uchar16 const_func __attribute__((overloadable)) convert_uchar16(long16);
12287 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat(long16);
12288 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rte(ulong16);
12289 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rte(ulong16);
12290 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtz(ulong16);
12291 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtz(ulong16);
12292 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtp(ulong16);
12293 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtp(ulong16);
12294 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtn(ulong16);
12295 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtn(ulong16);
12296 uchar16 const_func __attribute__((overloadable)) convert_uchar16(ulong16);
12297 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat(ulong16);
12298 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rte(float16);
12299 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rte(float16);
12300 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtz(float16);
12301 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtz(float16);
12302 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtp(float16);
12303 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtp(float16);
12304 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtn(float16);
12305 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtn(float16);
12306 uchar16 const_func __attribute__((overloadable)) convert_uchar16(float16);
12307 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat(float16);
12308 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rte(double16);
12309 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rte(double16);
12310 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtz(double16);
12311 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtz(double16);
12312 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtp(double16);
12313 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtp(double16);
12314 uchar16 const_func __attribute__((overloadable)) convert_uchar16_rtn(double16);
12315 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat_rtn(double16);
12316 uchar16 const_func __attribute__((overloadable)) convert_uchar16(double16);
12317 uchar16 const_func __attribute__((overloadable)) convert_uchar16_sat(double16);
12318
12319 short16 const_func __attribute__((overloadable)) convert_short16_rte(char16);
12320 short16 const_func __attribute__((overloadable)) convert_short16_sat_rte(char16);
12321 short16 const_func __attribute__((overloadable)) convert_short16_rtz(char16);
12322 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtz(char16);
12323 short16 const_func __attribute__((overloadable)) convert_short16_rtp(char16);
12324 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtp(char16);
12325 short16 const_func __attribute__((overloadable)) convert_short16_rtn(char16);
12326 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtn(char16);
12327 short16 const_func __attribute__((overloadable)) convert_short16(char16);
12328 short16 const_func __attribute__((overloadable)) convert_short16_sat(char16);
12329 short16 const_func __attribute__((overloadable)) convert_short16_rte(uchar16);
12330 short16 const_func __attribute__((overloadable)) convert_short16_sat_rte(uchar16);
12331 short16 const_func __attribute__((overloadable)) convert_short16_rtz(uchar16);
12332 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtz(uchar16);
12333 short16 const_func __attribute__((overloadable)) convert_short16_rtp(uchar16);
12334 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtp(uchar16);
12335 short16 const_func __attribute__((overloadable)) convert_short16_rtn(uchar16);
12336 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtn(uchar16);
12337 short16 const_func __attribute__((overloadable)) convert_short16(uchar16);
12338 short16 const_func __attribute__((overloadable)) convert_short16_sat(uchar16);
12339 short16 const_func __attribute__((overloadable)) convert_short16_rte(short16);
12340 short16 const_func __attribute__((overloadable)) convert_short16_sat_rte(short16);
12341 short16 const_func __attribute__((overloadable)) convert_short16_rtz(short16);
12342 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtz(short16);
12343 short16 const_func __attribute__((overloadable)) convert_short16_rtp(short16);
12344 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtp(short16);
12345 short16 const_func __attribute__((overloadable)) convert_short16_rtn(short16);
12346 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtn(short16);
12347 short16 const_func __attribute__((overloadable)) convert_short16(short16);
12348 short16 const_func __attribute__((overloadable)) convert_short16_sat(short16);
12349 short16 const_func __attribute__((overloadable)) convert_short16_rte(ushort16);
12350 short16 const_func __attribute__((overloadable)) convert_short16_sat_rte(ushort16);
12351 short16 const_func __attribute__((overloadable)) convert_short16_rtz(ushort16);
12352 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtz(ushort16);
12353 short16 const_func __attribute__((overloadable)) convert_short16_rtp(ushort16);
12354 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtp(ushort16);
12355 short16 const_func __attribute__((overloadable)) convert_short16_rtn(ushort16);
12356 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtn(ushort16);
12357 short16 const_func __attribute__((overloadable)) convert_short16(ushort16);
12358 short16 const_func __attribute__((overloadable)) convert_short16_sat(ushort16);
12359 short16 const_func __attribute__((overloadable)) convert_short16_rte(int16);
12360 short16 const_func __attribute__((overloadable)) convert_short16_sat_rte(int16);
12361 short16 const_func __attribute__((overloadable)) convert_short16_rtz(int16);
12362 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtz(int16);
12363 short16 const_func __attribute__((overloadable)) convert_short16_rtp(int16);
12364 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtp(int16);
12365 short16 const_func __attribute__((overloadable)) convert_short16_rtn(int16);
12366 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtn(int16);
```

```
12367 short16 const_func __attribute__((overloadable)) convert_short16(int16);
12368 short16 const_func __attribute__((overloadable)) convert_short16_sat(int16);
12369 short16 const_func __attribute__((overloadable)) convert_short16_rte(uint16);
12370 short16 const_func __attribute__((overloadable)) convert_short16_sat_rte(uint16);
12371 short16 const_func __attribute__((overloadable)) convert_short16_rtz(uint16);
12372 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtz(uint16);
12373 short16 const_func __attribute__((overloadable)) convert_short16_rtp(uint16);
12374 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtp(uint16);
12375 short16 const_func __attribute__((overloadable)) convert_short16_rtn(uint16);
12376 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtn(uint16);
12377 short16 const_func __attribute__((overloadable)) convert_short16(uint16);
12378 short16 const_func __attribute__((overloadable)) convert_short16_sat(uint16);
12379 short16 const_func __attribute__((overloadable)) convert_short16_rte(long16);
12380 short16 const_func __attribute__((overloadable)) convert_short16_sat_rte(long16);
12381 short16 const_func __attribute__((overloadable)) convert_short16_rtz(long16);
12382 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtz(long16);
12383 short16 const_func __attribute__((overloadable)) convert_short16_rtp(long16);
12384 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtp(long16);
12385 short16 const_func __attribute__((overloadable)) convert_short16_rtn(long16);
12386 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtn(long16);
12387 short16 const_func __attribute__((overloadable)) convert_short16(long16);
12388 short16 const_func __attribute__((overloadable)) convert_short16_sat(long16);
12389 short16 const_func __attribute__((overloadable)) convert_short16_rte(ulong16);
12390 short16 const_func __attribute__((overloadable)) convert_short16_sat_rte(ulong16);
12391 short16 const_func __attribute__((overloadable)) convert_short16_rtz(ulong16);
12392 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtz(ulong16);
12393 short16 const_func __attribute__((overloadable)) convert_short16_rtp(ulong16);
12394 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtp(ulong16);
12395 short16 const_func __attribute__((overloadable)) convert_short16_rtn(ulong16);
12396 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtn(ulong16);
12397 short16 const_func __attribute__((overloadable)) convert_short16(ulong16);
12398 short16 const_func __attribute__((overloadable)) convert_short16_sat(ulong16);
12399 short16 const_func __attribute__((overloadable)) convert_short16_rte(float16);
12400 short16 const_func __attribute__((overloadable)) convert_short16_sat_rte(float16);
12401 short16 const_func __attribute__((overloadable)) convert_short16_rtz(float16);
12402 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtz(float16);
12403 short16 const_func __attribute__((overloadable)) convert_short16_rtp(float16);
12404 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtp(float16);
12405 short16 const_func __attribute__((overloadable)) convert_short16_rtn(float16);
12406 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtn(float16);
12407 short16 const_func __attribute__((overloadable)) convert_short16(float16);
12408 short16 const_func __attribute__((overloadable)) convert_short16_sat(float16);
12409 short16 const_func __attribute__((overloadable)) convert_short16_rte(double16);
12410 short16 const_func __attribute__((overloadable)) convert_short16_sat_rte(double16);
12411 short16 const_func __attribute__((overloadable)) convert_short16_rtz(double16);
12412 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtz(double16);
12413 short16 const_func __attribute__((overloadable)) convert_short16_rtp(double16);
12414 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtp(double16);
12415 short16 const_func __attribute__((overloadable)) convert_short16_rtn(double16);
12416 short16 const_func __attribute__((overloadable)) convert_short16_sat_rtn(double16);
12417 short16 const_func __attribute__((overloadable)) convert_short16(double16);
12418 short16 const_func __attribute__((overloadable)) convert_short16_sat(double16);
12419
12420 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rte(char16);
12421 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rte(char16);
12422 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtz(char16);
12423 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtz(char16);
12424 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtp(char16);
12425 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtp(char16);
12426 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtn(char16);
12427 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtn(char16);
12428 ushort16 const_func __attribute__((overloadable)) convert_ushort16(char16);
12429 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat(char16);
12430 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rte(uchar16);
12431 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rte(uchar16);
12432 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtz(uchar16);
12433 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtz(uchar16);
12434 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtp(uchar16);
12435 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtp(uchar16);
12436 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtn(uchar16);
12437 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtn(uchar16);
12438 ushort16 const_func __attribute__((overloadable)) convert_ushort16(uchar16);
12439 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat(uchar16);
12440 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rte(short16);
12441 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rte(short16);
12442 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtz(short16);
12443 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtz(short16);
12444 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtp(short16);
12445 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtp(short16);
12446 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtn(short16);
12447 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtn(short16);
12448 ushort16 const_func __attribute__((overloadable)) convert_ushort16(short16);
12449 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat(short16);
12450 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rte(ushort16);
12451 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rte(ushort16);
12452 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtz(ushort16);
12453 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtz(ushort16);
```

```
12454 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtp(ushort16);
12455 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtp(ushort16);
12456 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtn(ushort16);
12457 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtn(ushort16);
12458 ushort16 const_func __attribute__((overloadable)) convert_ushort16(ushort16);
12459 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat(ushort16);
12460 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rte(int16);
12461 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rte(int16);
12462 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtz(int16);
12463 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtz(int16);
12464 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtp(int16);
12465 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtp(int16);
12466 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtn(int16);
12467 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtn(int16);
12468 ushort16 const_func __attribute__((overloadable)) convert_ushort16(int16);
12469 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat(int16);
12470 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rte(uint16);
12471 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rte(uint16);
12472 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtz(uint16);
12473 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtz(uint16);
12474 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtp(uint16);
12475 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtp(uint16);
12476 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtn(uint16);
12477 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtn(uint16);
12478 ushort16 const_func __attribute__((overloadable)) convert_ushort16(uint16);
12479 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat(uint16);
12480 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rte(long16);
12481 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rte(long16);
12482 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtz(long16);
12483 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtz(long16);
12484 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtp(long16);
12485 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtp(long16);
12486 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtn(long16);
12487 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtn(long16);
12488 ushort16 const_func __attribute__((overloadable)) convert_ushort16(long16);
12489 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat(long16);
12490 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rte(ulong16);
12491 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rte(ulong16);
12492 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtz(ulong16);
12493 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtz(ulong16);
12494 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtp(ulong16);
12495 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtp(ulong16);
12496 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtn(ulong16);
12497 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtn(ulong16);
12498 ushort16 const_func __attribute__((overloadable)) convert_ushort16(ulong16);
12499 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat(ulong16);
12500 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rte(float16);
12501 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rte(float16);
12502 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtz(float16);
12503 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtz(float16);
12504 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtp(float16);
12505 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtp(float16);
12506 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtn(float16);
12507 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtn(float16);
12508 ushort16 const_func __attribute__((overloadable)) convert_ushort16(float16);
12509 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat(float16);
12510 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rte(double16);
12511 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rte(double16);
12512 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtz(double16);
12513 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtz(double16);
12514 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtp(double16);
12515 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtp(double16);
12516 ushort16 const_func __attribute__((overloadable)) convert_ushort16_rtn(double16);
12517 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat_rtn(double16);
12518 ushort16 const_func __attribute__((overloadable)) convert_ushort16(double16);
12519 ushort16 const_func __attribute__((overloadable)) convert_ushort16_sat(double16);
12520
12521 int16 const_func __attribute__((overloadable)) convert_int16_rte(char16);
12522 int16 const_func __attribute__((overloadable)) convert_int16_sat_rte(char16);
12523 int16 const_func __attribute__((overloadable)) convert_int16_rtz(char16);
12524 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtz(char16);
12525 int16 const_func __attribute__((overloadable)) convert_int16_rtp(char16);
12526 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtp(char16);
12527 int16 const_func __attribute__((overloadable)) convert_int16_rtn(char16);
12528 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtn(char16);
12529 int16 const_func __attribute__((overloadable)) convert_int16(char16);
12530 int16 const_func __attribute__((overloadable)) convert_int16_sat(char16);
12531 int16 const_func __attribute__((overloadable)) convert_int16_rte(uchar16);
12532 int16 const_func __attribute__((overloadable)) convert_int16_sat_rte(uchar16);
12533 int16 const_func __attribute__((overloadable)) convert_int16_rtz(uchar16);
12534 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtz(uchar16);
12535 int16 const_func __attribute__((overloadable)) convert_int16_rtp(uchar16);
12536 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtp(uchar16);
12537 int16 const_func __attribute__((overloadable)) convert_int16_rtn(uchar16);
12538 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtn(uchar16);
12539 int16 const_func __attribute__((overloadable)) convert_int16(uchar16);
12540 int16 const_func __attribute__((overloadable)) convert_int16_sat(uchar16);
```

```
12541 int16 const_func __attribute__((overloadable)) convert_int16_rte(short16);
12542 int16 const_func __attribute__((overloadable)) convert_int16_sat_rte(short16);
12543 int16 const_func __attribute__((overloadable)) convert_int16_rtz(short16);
12544 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtz(short16);
12545 int16 const_func __attribute__((overloadable)) convert_int16_rtp(short16);
12546 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtp(short16);
12547 int16 const_func __attribute__((overloadable)) convert_int16_rtn(short16);
12548 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtn(short16);
12549 int16 const_func __attribute__((overloadable)) convert_int16(short16);
12550 int16 const_func __attribute__((overloadable)) convert_int16_sat(short16);
12551 int16 const_func __attribute__((overloadable)) convert_int16_rte(ushort16);
12552 int16 const_func __attribute__((overloadable)) convert_int16_sat_rte(ushort16);
12553 int16 const_func __attribute__((overloadable)) convert_int16_rtz(ushort16);
12554 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtz(ushort16);
12555 int16 const_func __attribute__((overloadable)) convert_int16_rtp(ushort16);
12556 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtp(ushort16);
12557 int16 const_func __attribute__((overloadable)) convert_int16_rtn(ushort16);
12558 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtn(ushort16);
12559 int16 const_func __attribute__((overloadable)) convert_int16(ushort16);
12560 int16 const_func __attribute__((overloadable)) convert_int16_sat(ushort16);
12561 int16 const_func __attribute__((overloadable)) convert_int16_rte(int16);
12562 int16 const_func __attribute__((overloadable)) convert_int16_sat_rte(int16);
12563 int16 const_func __attribute__((overloadable)) convert_int16_rtz(int16);
12564 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtz(int16);
12565 int16 const_func __attribute__((overloadable)) convert_int16_rtp(int16);
12566 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtp(int16);
12567 int16 const_func __attribute__((overloadable)) convert_int16_rtn(int16);
12568 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtn(int16);
12569 int16 const_func __attribute__((overloadable)) convert_int16(int16);
12570 int16 const_func __attribute__((overloadable)) convert_int16_sat(int16);
12571 int16 const_func __attribute__((overloadable)) convert_int16_rte(uint16);
12572 int16 const_func __attribute__((overloadable)) convert_int16_sat_rte(uint16);
12573 int16 const_func __attribute__((overloadable)) convert_int16_rtz(uint16);
12574 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtz(uint16);
12575 int16 const_func __attribute__((overloadable)) convert_int16_rtp(uint16);
12576 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtp(uint16);
12577 int16 const_func __attribute__((overloadable)) convert_int16_rtn(uint16);
12578 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtn(uint16);
12579 int16 const_func __attribute__((overloadable)) convert_int16(uint16);
12580 int16 const_func __attribute__((overloadable)) convert_int16_sat(uint16);
12581 int16 const_func __attribute__((overloadable)) convert_int16_rte(long16);
12582 int16 const_func __attribute__((overloadable)) convert_int16_sat_rte(long16);
12583 int16 const_func __attribute__((overloadable)) convert_int16_rtz(long16);
12584 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtz(long16);
12585 int16 const_func __attribute__((overloadable)) convert_int16_rtp(long16);
12586 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtp(long16);
12587 int16 const_func __attribute__((overloadable)) convert_int16_rtn(long16);
12588 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtn(long16);
12589 int16 const_func __attribute__((overloadable)) convert_int16(long16);
12590 int16 const_func __attribute__((overloadable)) convert_int16_sat(long16);
12591 int16 const_func __attribute__((overloadable)) convert_int16_rte(ulong16);
12592 int16 const_func __attribute__((overloadable)) convert_int16_sat_rte(ulong16);
12593 int16 const_func __attribute__((overloadable)) convert_int16_rtz(ulong16);
12594 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtz(ulong16);
12595 int16 const_func __attribute__((overloadable)) convert_int16_rtp(ulong16);
12596 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtp(ulong16);
12597 int16 const_func __attribute__((overloadable)) convert_int16_rtn(ulong16);
12598 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtn(ulong16);
12599 int16 const_func __attribute__((overloadable)) convert_int16(ulong16);
12600 int16 const_func __attribute__((overloadable)) convert_int16_sat(ulong16);
12601 int16 const_func __attribute__((overloadable)) convert_int16_rte(float16);
12602 int16 const_func __attribute__((overloadable)) convert_int16_sat_rte(float16);
12603 int16 const_func __attribute__((overloadable)) convert_int16_rtz(float16);
12604 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtz(float16);
12605 int16 const_func __attribute__((overloadable)) convert_int16_rtp(float16);
12606 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtp(float16);
12607 int16 const_func __attribute__((overloadable)) convert_int16_rtn(float16);
12608 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtn(float16);
12609 int16 const_func __attribute__((overloadable)) convert_int16(float16);
12610 int16 const_func __attribute__((overloadable)) convert_int16_sat(float16);
12611 int16 const_func __attribute__((overloadable)) convert_int16_rte(double16);
12612 int16 const_func __attribute__((overloadable)) convert_int16_sat_rte(double16);
12613 int16 const_func __attribute__((overloadable)) convert_int16_rtz(double16);
12614 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtz(double16);
12615 int16 const_func __attribute__((overloadable)) convert_int16_rtp(double16);
12616 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtp(double16);
12617 int16 const_func __attribute__((overloadable)) convert_int16_rtn(double16);
12618 int16 const_func __attribute__((overloadable)) convert_int16_sat_rtn(double16);
12619 int16 const_func __attribute__((overloadable)) convert_int16(double16);
12620 int16 const_func __attribute__((overloadable)) convert_int16_sat(double16);
12621
12622 uint16 const_func __attribute__((overloadable)) convert_uint16_rte(char16);
12623 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rte(char16);
12624 uint16 const_func __attribute__((overloadable)) convert_uint16_rtz(char16);
12625 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtz(char16);
12626 uint16 const_func __attribute__((overloadable)) convert_uint16_rtp(char16);
12627 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtp(char16);
```

```
12628 uint16 const_func __attribute__((overloadable)) convert_uint16_rtn(char16);
12629 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtn(char16);
12630 uint16 const_func __attribute__((overloadable)) convert_uint16(char16);
12631 uint16 const_func __attribute__((overloadable)) convert_uint16_sat(char16);
12632 uint16 const_func __attribute__((overloadable)) convert_uint16_rte(uchar16);
12633 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rte(uchar16);
12634 uint16 const_func __attribute__((overloadable)) convert_uint16_rtz(uchar16);
12635 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtz(uchar16);
12636 uint16 const_func __attribute__((overloadable)) convert_uint16_rtp(uchar16);
12637 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtp(uchar16);
12638 uint16 const_func __attribute__((overloadable)) convert_uint16_rtn(uchar16);
12639 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtn(uchar16);
12640 uint16 const_func __attribute__((overloadable)) convert_uint16(uchar16);
12641 uint16 const_func __attribute__((overloadable)) convert_uint16_sat(uchar16);
12642 uint16 const_func __attribute__((overloadable)) convert_uint16_rte(short16);
12643 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rte(short16);
12644 uint16 const_func __attribute__((overloadable)) convert_uint16_rtz(short16);
12645 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtz(short16);
12646 uint16 const_func __attribute__((overloadable)) convert_uint16_rtp(short16);
12647 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtp(short16);
12648 uint16 const_func __attribute__((overloadable)) convert_uint16_rtn(short16);
12649 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtn(short16);
12650 uint16 const_func __attribute__((overloadable)) convert_uint16(short16);
12651 uint16 const_func __attribute__((overloadable)) convert_uint16_sat(short16);
12652 uint16 const_func __attribute__((overloadable)) convert_uint16_rte(ushort16);
12653 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rte(ushort16);
12654 uint16 const_func __attribute__((overloadable)) convert_uint16_rtz(ushort16);
12655 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtz(ushort16);
12656 uint16 const_func __attribute__((overloadable)) convert_uint16_rtp(ushort16);
12657 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtp(ushort16);
12658 uint16 const_func __attribute__((overloadable)) convert_uint16_rtn(ushort16);
12659 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtn(ushort16);
12660 uint16 const_func __attribute__((overloadable)) convert_uint16(ushort16);
12661 uint16 const_func __attribute__((overloadable)) convert_uint16_sat(ushort16);
12662 uint16 const_func __attribute__((overloadable)) convert_uint16_rte(int16);
12663 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rte(int16);
12664 uint16 const_func __attribute__((overloadable)) convert_uint16_rtz(int16);
12665 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtz(int16);
12666 uint16 const_func __attribute__((overloadable)) convert_uint16_rtp(int16);
12667 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtp(int16);
12668 uint16 const_func __attribute__((overloadable)) convert_uint16_rtn(int16);
12669 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtn(int16);
12670 uint16 const_func __attribute__((overloadable)) convert_uint16(int16);
12671 uint16 const_func __attribute__((overloadable)) convert_uint16_sat(int16);
12672 uint16 const_func __attribute__((overloadable)) convert_uint16_rte(uint16);
12673 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rte(uint16);
12674 uint16 const_func __attribute__((overloadable)) convert_uint16_rtz(uint16);
12675 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtz(uint16);
12676 uint16 const_func __attribute__((overloadable)) convert_uint16_rtp(uint16);
12677 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtp(uint16);
12678 uint16 const_func __attribute__((overloadable)) convert_uint16_rtn(uint16);
12679 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtn(uint16);
12680 uint16 const_func __attribute__((overloadable)) convert_uint16(uint16);
12681 uint16 const_func __attribute__((overloadable)) convert_uint16_sat(uint16);
12682 uint16 const_func __attribute__((overloadable)) convert_uint16_rte(long16);
12683 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rte(long16);
12684 uint16 const_func __attribute__((overloadable)) convert_uint16_rtz(long16);
12685 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtz(long16);
12686 uint16 const_func __attribute__((overloadable)) convert_uint16_rtp(long16);
12687 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtp(long16);
12688 uint16 const_func __attribute__((overloadable)) convert_uint16_rtn(long16);
12689 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtn(long16);
12690 uint16 const_func __attribute__((overloadable)) convert_uint16(long16);
12691 uint16 const_func __attribute__((overloadable)) convert_uint16_sat(long16);
12692 uint16 const_func __attribute__((overloadable)) convert_uint16_rte(ulong16);
12693 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rte(ulong16);
12694 uint16 const_func __attribute__((overloadable)) convert_uint16_rtz(ulong16);
12695 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtz(ulong16);
12696 uint16 const_func __attribute__((overloadable)) convert_uint16_rtp(ulong16);
12697 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtp(ulong16);
12698 uint16 const_func __attribute__((overloadable)) convert_uint16_rtn(ulong16);
12699 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtn(ulong16);
12700 uint16 const_func __attribute__((overloadable)) convert_uint16(ulong16);
12701 uint16 const_func __attribute__((overloadable)) convert_uint16_sat(ulong16);
12702 uint16 const_func __attribute__((overloadable)) convert_uint16_rte(float16);
12703 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rte(float16);
12704 uint16 const_func __attribute__((overloadable)) convert_uint16_rtz(float16);
12705 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtz(float16);
12706 uint16 const_func __attribute__((overloadable)) convert_uint16_rtp(float16);
12707 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtp(float16);
12708 uint16 const_func __attribute__((overloadable)) convert_uint16_rtn(float16);
12709 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtn(float16);
12710 uint16 const_func __attribute__((overloadable)) convert_uint16(float16);
12711 uint16 const_func __attribute__((overloadable)) convert_uint16_sat(float16);
12712 uint16 const_func __attribute__((overloadable)) convert_uint16_rte(double16);
12713 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rte(double16);
12714 uint16 const_func __attribute__((overloadable)) convert_uint16_rtz(double16);
```

```
12715 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtz(double16);
12716 uint16 const_func __attribute__((overloadable)) convert_uint16_rtp(double16);
12717 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtp(double16);
12718 uint16 const_func __attribute__((overloadable)) convert_uint16_rtn(double16);
12719 uint16 const_func __attribute__((overloadable)) convert_uint16_sat_rtn(double16);
12720 uint16 const_func __attribute__((overloadable)) convert_uint16(double16);
12721 uint16 const_func __attribute__((overloadable)) convert_uint16_sat(double16);
12722 long16 const_func __attribute__((overloadable)) convert_long16_rte(char16);
12723 long16 const_func __attribute__((overloadable)) convert_long16_sat_rte(char16);
12724 long16 const_func __attribute__((overloadable)) convert_long16_rtz(char16);
12725 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtz(char16);
12726 long16 const_func __attribute__((overloadable)) convert_long16_rtp(char16);
12727 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtp(char16);
12728 long16 const_func __attribute__((overloadable)) convert_long16_rtn(char16);
12729 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtn(char16);
12730 long16 const_func __attribute__((overloadable)) convert_long16(char16);
12731 long16 const_func __attribute__((overloadable)) convert_long16_sat(char16);
12732 long16 const_func __attribute__((overloadable)) convert_long16_rte(uchar16);
12733 long16 const_func __attribute__((overloadable)) convert_long16_sat_rte(uchar16);
12734 long16 const_func __attribute__((overloadable)) convert_long16_rtz(uchar16);
12735 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtz(uchar16);
12736 long16 const_func __attribute__((overloadable)) convert_long16_rtp(uchar16);
12737 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtp(uchar16);
12738 long16 const_func __attribute__((overloadable)) convert_long16_rtn(uchar16);
12739 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtn(uchar16);
12740 long16 const_func __attribute__((overloadable)) convert_long16(uchar16);
12741 long16 const_func __attribute__((overloadable)) convert_long16_sat(uchar16);
12742 long16 const_func __attribute__((overloadable)) convert_long16_rte(short16);
12743 long16 const_func __attribute__((overloadable)) convert_long16_sat_rte(short16);
12744 long16 const_func __attribute__((overloadable)) convert_long16_rtz(short16);
12745 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtz(short16);
12746 long16 const_func __attribute__((overloadable)) convert_long16_rtp(short16);
12747 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtp(short16);
12748 long16 const_func __attribute__((overloadable)) convert_long16_rtn(short16);
12749 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtn(short16);
12750 long16 const_func __attribute__((overloadable)) convert_long16(short16);
12751 long16 const_func __attribute__((overloadable)) convert_long16_sat(short16);
12752 long16 const_func __attribute__((overloadable)) convert_long16_rte(ushort16);
12753 long16 const_func __attribute__((overloadable)) convert_long16_sat_rte(ushort16);
12754 long16 const_func __attribute__((overloadable)) convert_long16_rtz(ushort16);
12755 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtz(ushort16);
12756 long16 const_func __attribute__((overloadable)) convert_long16_rtp(ushort16);
12757 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtp(ushort16);
12758 long16 const_func __attribute__((overloadable)) convert_long16_rtn(ushort16);
12759 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtn(ushort16);
12760 long16 const_func __attribute__((overloadable)) convert_long16(ushort16);
12761 long16 const_func __attribute__((overloadable)) convert_long16_sat(ushort16);
12762 long16 const_func __attribute__((overloadable)) convert_long16_rte(int16);
12763 long16 const_func __attribute__((overloadable)) convert_long16_sat_rte(int16);
12764 long16 const_func __attribute__((overloadable)) convert_long16_rtz(int16);
12765 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtz(int16);
12766 long16 const_func __attribute__((overloadable)) convert_long16_rtp(int16);
12767 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtp(int16);
12768 long16 const_func __attribute__((overloadable)) convert_long16_rtn(int16);
12769 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtn(int16);
12770 long16 const_func __attribute__((overloadable)) convert_long16(int16);
12771 long16 const_func __attribute__((overloadable)) convert_long16_sat(int16);
12772 long16 const_func __attribute__((overloadable)) convert_long16_rte(uint16);
12773 long16 const_func __attribute__((overloadable)) convert_long16_sat_rte(uint16);
12774 long16 const_func __attribute__((overloadable)) convert_long16_rtz(uint16);
12775 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtz(uint16);
12776 long16 const_func __attribute__((overloadable)) convert_long16_rtp(uint16);
12777 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtp(uint16);
12778 long16 const_func __attribute__((overloadable)) convert_long16_rtn(uint16);
12779 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtn(uint16);
12780 long16 const_func __attribute__((overloadable)) convert_long16(uint16);
12781 long16 const_func __attribute__((overloadable)) convert_long16_sat(uint16);
12782 long16 const_func __attribute__((overloadable)) convert_long16_rte(long16);
12783 long16 const_func __attribute__((overloadable)) convert_long16_sat_rte(long16);
12784 long16 const_func __attribute__((overloadable)) convert_long16_rtz(long16);
12785 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtz(long16);
12786 long16 const_func __attribute__((overloadable)) convert_long16_rtp(long16);
12787 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtp(long16);
12788 long16 const_func __attribute__((overloadable)) convert_long16_rtn(long16);
12789 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtn(long16);
12790 long16 const_func __attribute__((overloadable)) convert_long16(long16);
12791 long16 const_func __attribute__((overloadable)) convert_long16_sat(long16);
12792 long16 const_func __attribute__((overloadable)) convert_long16_rte(ulong16);
12793 long16 const_func __attribute__((overloadable)) convert_long16_sat_rte(ulong16);
12794 long16 const_func __attribute__((overloadable)) convert_long16_rtz(ulong16);
12795 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtz(ulong16);
12796 long16 const_func __attribute__((overloadable)) convert_long16_rtp(ulong16);
12797 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtp(ulong16);
12798 long16 const_func __attribute__((overloadable)) convert_long16_rtn(ulong16);
12799 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtn(ulong16);
12800 long16 const_func __attribute__((overloadable)) convert_long16(ulong16);
12801 long16 const_func __attribute__((overloadable)) convert_long16_sat(ulong16);
```

```
12802 long16 const_func __attribute__((overloadable)) convert_long16_rte(float16);
12803 long16 const_func __attribute__((overloadable)) convert_long16_sat_rte(float16);
12804 long16 const_func __attribute__((overloadable)) convert_long16_rtz(float16);
12805 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtz(float16);
12806 long16 const_func __attribute__((overloadable)) convert_long16_rtp(float16);
12807 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtp(float16);
12808 long16 const_func __attribute__((overloadable)) convert_long16_rtn(float16);
12809 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtn(float16);
12810 long16 const_func __attribute__((overloadable)) convert_long16(float16);
12811 long16 const_func __attribute__((overloadable)) convert_long16_sat(float16);
12812 long16 const_func __attribute__((overloadable)) convert_long16_rte(double16);
12813 long16 const_func __attribute__((overloadable)) convert_long16_sat_rte(double16);
12814 long16 const_func __attribute__((overloadable)) convert_long16_rtz(double16);
12815 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtz(double16);
12816 long16 const_func __attribute__((overloadable)) convert_long16_rtp(double16);
12817 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtp(double16);
12818 long16 const_func __attribute__((overloadable)) convert_long16_rtn(double16);
12819 long16 const_func __attribute__((overloadable)) convert_long16_sat_rtn(double16);
12820 long16 const_func __attribute__((overloadable)) convert_long16(double16);
12821 long16 const_func __attribute__((overloadable)) convert_long16_sat(double16);
12822
12823 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rte(char16);
12824 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rte(char16);
12825 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtz(char16);
12826 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtz(char16);
12827 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtp(char16);
12828 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtp(char16);
12829 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtn(char16);
12830 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtn(char16);
12831 ulong16 const_func __attribute__((overloadable)) convert_ulong16(char16);
12832 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat(char16);
12833 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rte(uchar16);
12834 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rte(uchar16);
12835 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtz(uchar16);
12836 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtz(uchar16);
12837 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtp(uchar16);
12838 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtp(uchar16);
12839 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtn(uchar16);
12840 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtn(uchar16);
12841 ulong16 const_func __attribute__((overloadable)) convert_ulong16(uchar16);
12842 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat(uchar16);
12843 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rte(short16);
12844 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rte(short16);
12845 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtz(short16);
12846 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtz(short16);
12847 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtp(short16);
12848 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtp(short16);
12849 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtn(short16);
12850 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtn(short16);
12851 ulong16 const_func __attribute__((overloadable)) convert_ulong16(short16);
12852 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat(short16);
12853 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rte(ushort16);
12854 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rte(ushort16);
12855 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtz(ushort16);
12856 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtz(ushort16);
12857 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtp(ushort16);
12858 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtp(ushort16);
12859 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtn(ushort16);
12860 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtn(ushort16);
12861 ulong16 const_func __attribute__((overloadable)) convert_ulong16(ushort16);
12862 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat(ushort16);
12863 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rte(int16);
12864 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rte(int16);
12865 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtz(int16);
12866 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtz(int16);
12867 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtp(int16);
12868 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtp(int16);
12869 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtn(int16);
12870 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtn(int16);
12871 ulong16 const_func __attribute__((overloadable)) convert_ulong16(int16);
12872 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat(int16);
12873 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rte(uint16);
12874 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rte(uint16);
12875 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtz(uint16);
12876 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtz(uint16);
12877 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtp(uint16);
12878 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtp(uint16);
12879 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtn(uint16);
12880 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtn(uint16);
12881 ulong16 const_func __attribute__((overloadable)) convert_ulong16(uint16);
12882 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat(uint16);
12883 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rte(long16);
12884 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rte(long16);
12885 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtz(long16);
12886 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtz(long16);
12887 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtp(long16);
12888 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtp(long16);
```

```
12889 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtn(long16);
12890 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtn(long16);
12891 ulong16 const_func __attribute__((overloadable)) convert_ulong16(long16);
12892 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat(long16);
12893 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rte(ulong16);
12894 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rte(ulong16);
12895 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtz(ulong16);
12896 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtz(ulong16);
12897 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtp(ulong16);
12898 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtp(ulong16);
12899 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtn(ulong16);
12900 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtn(ulong16);
12901 ulong16 const_func __attribute__((overloadable)) convert_ulong16(ulong16);
12902 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat(ulong16);
12903 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rte(float16);
12904 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rte(float16);
12905 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtz(float16);
12906 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtz(float16);
12907 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtp(float16);
12908 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtp(float16);
12909 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtn(float16);
12910 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtn(float16);
12911 ulong16 const_func __attribute__((overloadable)) convert_ulong16(float16);
12912 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat(float16);
12913 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rte(double16);
12914 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rte(double16);
12915 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtz(double16);
12916 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtz(double16);
12917 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtp(double16);
12918 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtp(double16);
12919 ulong16 const_func __attribute__((overloadable)) convert_ulong16_rtn(double16);
12920 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat_rtn(double16);
12921 ulong16 const_func __attribute__((overloadable)) convert_ulong16(double16);
12922 ulong16 const_func __attribute__((overloadable)) convert_ulong16_sat(double16);
12923
12924 float16 const_func __attribute__((overloadable)) convert_float16_rte(char16);
12925 float16 const_func __attribute__((overloadable)) convert_float16_rtz(char16);
12926 float16 const_func __attribute__((overloadable)) convert_float16_rtp(char16);
12927 float16 const_func __attribute__((overloadable)) convert_float16_rtn(char16);
12928 float16 const_func __attribute__((overloadable)) convert_float16(char16);
12929 float16 const_func __attribute__((overloadable)) convert_float16_rte(uchar16);
12930 float16 const_func __attribute__((overloadable)) convert_float16_rtz(uchar16);
12931 float16 const_func __attribute__((overloadable)) convert_float16_rtp(uchar16);
12932 float16 const_func __attribute__((overloadable)) convert_float16_rtn(uchar16);
12933 float16 const_func __attribute__((overloadable)) convert_float16(uchar16);
12934 float16 const_func __attribute__((overloadable)) convert_float16_rte(short16);
12935 float16 const_func __attribute__((overloadable)) convert_float16_rtz(short16);
12936 float16 const_func __attribute__((overloadable)) convert_float16_rtp(short16);
12937 float16 const_func __attribute__((overloadable)) convert_float16_rtn(short16);
12938 float16 const_func __attribute__((overloadable)) convert_float16(short16);
12939 float16 const_func __attribute__((overloadable)) convert_float16_rte(ushort16);
12940 float16 const_func __attribute__((overloadable)) convert_float16_rtz(ushort16);
12941 float16 const_func __attribute__((overloadable)) convert_float16_rtp(ushort16);
12942 float16 const_func __attribute__((overloadable)) convert_float16_rtn(ushort16);
12943 float16 const_func __attribute__((overloadable)) convert_float16(ushort16);
12944 float16 const_func __attribute__((overloadable)) convert_float16_rte(int16);
12945 float16 const_func __attribute__((overloadable)) convert_float16_rtz(int16);
12946 float16 const_func __attribute__((overloadable)) convert_float16_rtp(int16);
12947 float16 const_func __attribute__((overloadable)) convert_float16_rtn(int16);
12948 float16 const_func __attribute__((overloadable)) convert_float16(int16);
12949 float16 const_func __attribute__((overloadable)) convert_float16_rte(uint16);
12950 float16 const_func __attribute__((overloadable)) convert_float16_rtz(uint16);
12951 float16 const_func __attribute__((overloadable)) convert_float16_rtp(uint16);
12952 float16 const_func __attribute__((overloadable)) convert_float16_rtn(uint16);
12953 float16 const_func __attribute__((overloadable)) convert_float16(uint16);
12954 float16 const_func __attribute__((overloadable)) convert_float16_rte(long16);
12955 float16 const_func __attribute__((overloadable)) convert_float16_rtz(long16);
12956 float16 const_func __attribute__((overloadable)) convert_float16_rtp(long16);
12957 float16 const_func __attribute__((overloadable)) convert_float16_rtn(long16);
12958 float16 const_func __attribute__((overloadable)) convert_float16(long16);
12959 float16 const_func __attribute__((overloadable)) convert_float16_rte(ulong16);
12960 float16 const_func __attribute__((overloadable)) convert_float16_rtz(ulong16);
12961 float16 const_func __attribute__((overloadable)) convert_float16_rtp(ulong16);
12962 float16 const_func __attribute__((overloadable)) convert_float16_rtn(ulong16);
12963 float16 const_func __attribute__((overloadable)) convert_float16(ulong16);
12964 float16 const_func __attribute__((overloadable)) convert_float16_rte(float16);
12965 float16 const_func __attribute__((overloadable)) convert_float16_rtz(float16);
12966 float16 const_func __attribute__((overloadable)) convert_float16_rtp(float16);
12967 float16 const_func __attribute__((overloadable)) convert_float16_rtn(float16);
12968 float16 const_func __attribute__((overloadable)) convert_float16(float16);
12969 float16 const_func __attribute__((overloadable)) convert_float16_rte(double16);
12970 float16 const_func __attribute__((overloadable)) convert_float16_rtz(double16);
12971 float16 const_func __attribute__((overloadable)) convert_float16_rtp(double16);
12972 float16 const_func __attribute__((overloadable)) convert_float16_rtn(double16);
12973 float16 const_func __attribute__((overloadable)) convert_float16(double16);
12974
12975 double16 const_func __attribute__((overloadable)) convert_double16_rte(char16);
```

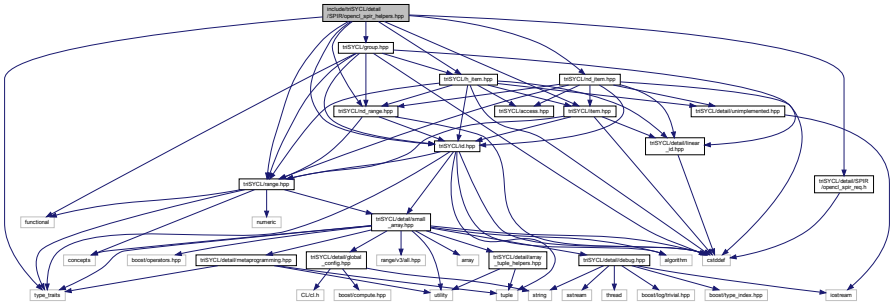
```
12976 double16 const_func __attribute__((overloadable)) convert_double16_rtz(char16);
12977 double16 const_func __attribute__((overloadable)) convert_double16_rtp(char16);
12978 double16 const_func __attribute__((overloadable)) convert_double16_rtn(char16);
12979 double16 const_func __attribute__((overloadable)) convert_double16(char16);
12980 double16 const_func __attribute__((overloadable)) convert_double16_rte(uchar16);
12981 double16 const_func __attribute__((overloadable)) convert_double16_rtz(uchar16);
12982 double16 const_func __attribute__((overloadable)) convert_double16_rtp(uchar16);
12983 double16 const_func __attribute__((overloadable)) convert_double16_rtn(uchar16);
12984 double16 const_func __attribute__((overloadable)) convert_double16(uchar16);
12985 double16 const_func __attribute__((overloadable)) convert_double16_rte(short16);
12986 double16 const_func __attribute__((overloadable)) convert_double16_rtz(short16);
12987 double16 const_func __attribute__((overloadable)) convert_double16_rtp(short16);
12988 double16 const_func __attribute__((overloadable)) convert_double16_rtn(short16);
12989 double16 const_func __attribute__((overloadable)) convert_double16(short16);
12990 double16 const_func __attribute__((overloadable)) convert_double16_rte(ushort16);
12991 double16 const_func __attribute__((overloadable)) convert_double16_rtz(ushort16);
12992 double16 const_func __attribute__((overloadable)) convert_double16_rtp(ushort16);
12993 double16 const_func __attribute__((overloadable)) convert_double16_rtn(ushort16);
12994 double16 const_func __attribute__((overloadable)) convert_double16(ushort16);
12995 double16 const_func __attribute__((overloadable)) convert_double16_rte(int16);
12996 double16 const_func __attribute__((overloadable)) convert_double16_rtz(int16);
12997 double16 const_func __attribute__((overloadable)) convert_double16_rtp(int16);
12998 double16 const_func __attribute__((overloadable)) convert_double16_rtn(int16);
12999 double16 const_func __attribute__((overloadable)) convert_double16(int16);
13000 double16 const_func __attribute__((overloadable)) convert_double16_rte(uint16);
13001 double16 const_func __attribute__((overloadable)) convert_double16_rtz(uint16);
13002 double16 const_func __attribute__((overloadable)) convert_double16_rtp(uint16);
13003 double16 const_func __attribute__((overloadable)) convert_double16_rtn(uint16);
13004 double16 const_func __attribute__((overloadable)) convert_double16(uint16);
13005 double16 const_func __attribute__((overloadable)) convert_double16_rte(long16);
13006 double16 const_func __attribute__((overloadable)) convert_double16_rtz(long16);
13007 double16 const_func __attribute__((overloadable)) convert_double16_rtp(long16);
13008 double16 const_func __attribute__((overloadable)) convert_double16_rtn(long16);
13009 double16 const_func __attribute__((overloadable)) convert_double16(long16);
13010 double16 const_func __attribute__((overloadable)) convert_double16_rte(ulong16);
13011 double16 const_func __attribute__((overloadable)) convert_double16_rtz(ulong16);
13012 double16 const_func __attribute__((overloadable)) convert_double16_rtp(ulong16);
13013 double16 const_func __attribute__((overloadable)) convert_double16_rtn(ulong16);
13014 double16 const_func __attribute__((overloadable)) convert_double16(ulong16);
13015 double16 const_func __attribute__((overloadable)) convert_double16_rte(float16);
13016 double16 const_func __attribute__((overloadable)) convert_double16_rtz(float16);
13017 double16 const_func __attribute__((overloadable)) convert_double16_rtp(float16);
13018 double16 const_func __attribute__((overloadable)) convert_double16_rtn(float16);
13019 double16 const_func __attribute__((overloadable)) convert_double16(float16);
13020 double16 const_func __attribute__((overloadable)) convert_double16_rte(double16);
13021 double16 const_func __attribute__((overloadable)) convert_double16_rtz(double16);
13022 double16 const_func __attribute__((overloadable)) convert_double16_rtp(double16);
13023 double16 const_func __attribute__((overloadable)) convert_double16_rtn(double16);
13024 double16 const_func __attribute__((overloadable)) convert_double16(double16);
13025
13026 #endif
```

13.85 include/triSYCL/detail/SPIR/opencl_spir_helpers.hpp File Reference

Adds helper functions to populate Id's and Range's with OpenCL SPIR intrinsics data: e.g.

```
#include <type_traits>
#include "triSYCL/detail/SPIR/opencl_spir_req.h"
#include "triSYCL/group.hpp"
#include "triSYCL/h_item.hpp"
#include "triSYCL/id.hpp"
#include "triSYCL/item.hpp"
#include "triSYCL/nd_item.hpp"
#include "triSYCL/nd_range.hpp"
#include "triSYCL/range.hpp"
```

Include dependency graph for opencl_spir_helpers.hpp:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.
- namespace trisycl::detail::spir

Functions

- template<typename T , typename F >
  auto trisycl::detail::spir::make_array_with_func (F unary_func)
- template<int Dimensions>
  nd_range< Dimensions > trisycl::detail::spir::make_spir_nd_range ()
- template<int Dimensions>
  h_item< Dimensions > trisycl::detail::spir::make_spir_h_item ()
- template<int Dimensions, typename Index_T >
  auto trisycl::detail::spir::create_parallel_for_arg (Index_T index_or_range)

13.85.1 Detailed Description

Adds helper functions to populate Id's and Range's with OpenCL SPIR intrinsics data: e.g.

get_global_id(0)

Todo implement group and nd_item and to add nd_item to create_parallel_for_arg

andrew point gozillon at yahoo point com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file opencl_spir_helpers.hpp.

13.86 opencl_spir_helpers.hpp

Go to the documentation of this file.

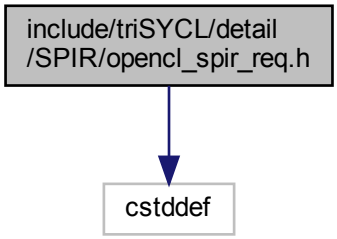
```
00001 #ifndef TRISYCL_SYCL_DETAIL_SPIR_OPENCL_SPIR_HELPERS_HPP
00002 #define TRISYCL_SYCL_DETAIL_SPIR_OPENCL_SPIR_HELPERS_HPP
00003
00004 /** \file
00005 Adds helper functions to populate Id's and Range's with OpenCL SPIR
00006 intrinsics data:  e.g.  get_global_id(0)
00007
00008 \todo implement group and nd_item and to add nd_item to
00009 create_parallel_for_arg
00010
00011 andrew point gozillon at yahoo point com
00012
00013 This file is distributed under the University of Illinois Open Source
00014 License.  See LICENSE.TXT for details.
00015 */
00016 #include <type_traits>
00017
00018 #include "triSYCL/detail/SPIR/opencl_spir_req.h"
00019 #include "triSYCL/group.hpp"
00020 #include "triSYCL/h_item.hpp"
00021 #include "triSYCL/id.hpp"
00022 #include "triSYCL/item.hpp"
00023 #include "triSYCL/nd_item.hpp"
00024 #include "triSYCL/nd_range.hpp"
00025 #include "triSYCL/range.hpp"
00026
00027 /** \addtogroup opencl_spir_helpers
00028 @{
00029 */
00030
00031 namespace trisycl::detail::spir {
00032
00033 // Fills the small_array_sycl of the type passed to it up to the dimensionality
00034 // of the type T (T generally intended to be a sycl id or range, but can be
00035 // anything with a dimensionality variable defined and a subscript operator)
00036 template <typename T,  typename F>
00037 auto make_array_with_func(F unary_func) {
00038   T id_or_range {};
00039
00040   for (size_t i = 0; i < id_or_range.dimensionality; ++i)
00041     id_or_range[i] = unary_func(i);
00042
00043   return id_or_range;
00044 }
00045
00046 template <int Dimensions>
00047 nd_range<Dimensions> make_spir_nd_range() {
00048   return nd_range<Dimensions> {
00049     make_array_with_func<range<Dimensions>»(get_global_size),
00050     make_array_with_func<range<Dimensions>>(get_local_size),
00051     make_array_with_func<id<Dimensions»(get_global_offset)
00052   };
00053 }
00054
00055 // template <int Dimensions>
00056 // nd_item<Dimensions> make_spir_nd_item() {}
00057
00058 template <int Dimensions>
00059 h_item<Dimensions> make_spir_h_item() {
00060   return h_item<Dimensions> {
00061     make_array_with_func<id<Dimensions»(get_global_offset),
00062     make_spir_nd_range<Dimensions>()
00063   };
00064 }
00065
00066 // group
00067 // template <int Dimensions>
00068 // group<Dimensions> make_spir_group() {}
00069
00070 // Uses a passed in index or range to identify the required index or range type
00071 // to generate
00072 template <int Dimensions, typename Index_T>
00073 auto create_parallel_for_arg(Index_T index_or_range) {
00074   if constexpr (std::is_same_v<Index_T, id<Dimensions») {
00075     index_or_range = make_array_with_func<Index_T>(get_global_id);
00076   } else if constexpr (std::is_same_v<Index_T, item<Dimensions») {
00077     // Item's do not contain local information in parallel_for
00078     index_or_range = Index_T {
00079       make_array_with_func<range<Dimensions»(get_global_size),
00080       make_array_with_func<id<Dimensions>>(get_global_id),
00081       make_array_with_func<id<Dimensions»(get_global_offset)
00082     };
```

```
00083   } else {
00084     static_assert(std::is_same_v<Index_T, item<Dimensions» ||
00085                   std::is_same_v<Index_T, id<Dimensions»,
00086                   "undefined Index_T passed to create_parallel_for_arg");
00087   }
00088
00089   return index_or_range;
00090 }
00091
00092 /// @} End the parallelism Doxygen group
00093
00094 } // namespace trisycl::details::spir
00095
00096 /*
00097 # Some Emacs stuff:
00098 ### Local Variables:
00099 ### ispell-local-dictionary:  "american"
00100 ### eval:  (flyspell-prog-mode)
00101 ### End:
00102 */
00103
00104 #endif // TRISYCL_SYCL_DETAIL_SPIR_OPENCL_SPIR_HELPERS_HPP
```

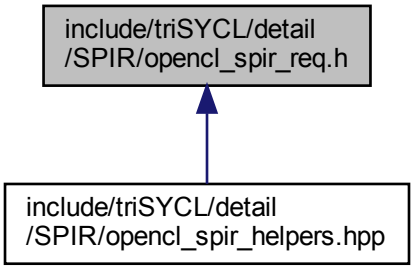
13.87 include/triSYCL/detail/SPIR/opencl_spir_req.h File Reference

`#include <cstddef>`

Include dependency graph for opencl_spir_req.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define __global __attribute__((address_space(1)))

  Added according to pocl/examples/example1-spir64/generate_spir.sh.
- #define __local __attribute__((address_space(2)))
- #define __constant __attribute__((address_space(3)))
- #define __private __attribute__((address_space(4)))
- #define __generic __attribute__((address_space(5)))
- #define const_func __attribute__((const))
- #define readonly __attribute__((pure))

Functions

- std::size_t const_func __attribute__ ((overloadable)) get_local_id(unsigned int dimindx)

  Returns the unique local work-item ID i.e.
- int printf (__constant const char *st,...)

13.87.1 Macro Definition Documentation

13.87.1.1 __constant

```
#define __constant __attribute__((address_space(3)))
```

Definition at line 43 of file opencl_spir_req.h.

13.87.1.2 __generic

```
#define __generic __attribute__((address_space(5)))
```

Definition at line 45 of file opencl_spir_req.h.

13.87.1.3 __global

```
#define __global __attribute__((address_space(1)))
```

Added according to pocl/examples/example1-spir64/generate_spir.sh.

Todo avoid this hack

Definition at line 41 of file opencl_spir_req.h.

13.87.1.4 __local

```
#define __local __attribute__((address_space(2)))
```

Definition at line 42 of file opencl_spir_req.h.

13.87.1.5 __private

```
#define __private __attribute__((address_space(4)))
```

Definition at line 44 of file opencl_spir_req.h.

13.87.1.6 const_func

```
#define const_func __attribute__((const))
```

Definition at line 47 of file opencl_spir_req.h.

13.87.1.7 readonly

```
#define readonly __attribute__((pure))
```

Definition at line 48 of file opencl_spir_req.h.

13.87.2 Function Documentation

13.87.2.1 __attribute__()

```
std::size_t const_func __attribute__ (
             (overloadable)  )
```

Returns the unique local work-item ID i.e.

get_global_offset returns the offset values specified in global_work_offset argument to clEnqueueNDRangeKernel.

get_group_id returns the work-group ID which is a number from 0 .

Returns the number of work-groups that will execute a kernel for dimension identified by dimindx.

Returns the number of local work-items specified in dimension identified by dimindx.

Returns the number of global work-items specified for dimension identified by dimindx.

Returns the unique global work-item ID value for dimension identified by dimindx.

a work-item within a specific work-group for dimension identified by dimindx. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_local_id() returns 0. For clEnqueueTask, this returns 0.

The global work-item ID specifies the work-item ID based on the number of global work-items specified to execute the kernel. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_global_id() returns 0. For clEnqueueTask, this returns 0.

This value is given by the global_work_size argument to clEnqueueNDRangeKernel. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_global_size() returns 1. For clEnqueueTask, this always returns 1.

This value is given by the local_work_size argument to clEnqueueNDRangeKernel if local_work_size is not NULL; otherwise the OpenCL implementation chooses an appropriate local_work_size value which is returned by this function. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_local_size() returns 1. For clEnqueueTask, this always returns 1.

Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_num_groups () returns 1. For clEnqueueTask, this always returns 1.

. get_num_groups(dimindx) – 1. Valid values of dimindx are 0 to get_work_dim() – 1. For other values, get_group↩
_id() returns 0. For clEnqueueTask, this returns 0.

Valid values of dimindx are 0 to get_work_dim() – 1. For other values, get_global_offset() returns 0. For cl↩
EnqueueTask, this returns 0.

Write color value to location specified by coord.xy in the 2D image layer identified by index coord.z in the 2D image array.

Use coord.xy to do an element lookup in the 2D image layer identified by index coord.z in the 2D image array.

Return the number of images in the 1D image array.

Return the number of images in the 2D image array.

Return the 3D image width, height, and depth as an int4 type.

Return the 2D image width and height as an int2 type.

Return the image channel order.

Return the image depth in pixels.

Return the image height in pixels.

Return the image width in pixels.

Write color value to location specified by coordinate (x, y, z) in the 3D image object specified by image.

Use the coordinate (coord.x, coord.y, coord.z) to do an element lookup in the 3D image object specified by image.

Write color value to location specified by coordinate (x, y) in the 2D image object specified by image.

Use the coordinate (x, y) to do an element lookup in the 2D image object specified by image.

The shuffle and shuffle2 built-in functions construct a permutation of elements from one or two input vectors respectively that are of the same type, returning a vector with the same element type as the input and length that is the same as the shuffle mask.

Swaps the old value stored at location p with new value given by val.

unsigned int atomic_sub (volatile __global unsigned int *p, unsigned int val) int atomic_sub (volatile __local int *p, int val) unsigned int atomic_sub (volatile __local unsigned int *p, unsigned int val) Read the 32-bit value (referred to as old) stored at location pointed by p.

Read the 32-bit value (referred to as old) stored at location pointed by p.

Prefetch num_elements * sizeof(gentype) bytes into the global cache.

Wait for events that identify the async_work_group_copy operations to complete.

Perform an async gather of num_elements gentype elements from src to dst.

event_t async_work_group_copy (__global gentype *dst, const __local gentype *src, size_t num_elements, event_t event) Perform an async copy of num_elements gentype elements from src to dst.

Write memory barrier that orders only stores.

Read memory barrier that orders only loads.

Orders loads and stores of a work-item executing a kernel.

For n = 1, 2, 4, 8 and 16 read sizeof (halfn) bytes of data from address (p + (offset * n)).

The floatn value given by data is converted to a halfn value using the appropriate rounding mode.

The float value given by data is first converted to a half value using the appropriate rounding mode.

Read sizeof (halfn) bytes of data from address (p + (offset * n)).

Read sizeof (half) bytes of data from address (p + offset).

Write sizeof (gentypen) bytes given by data to address (p + (offset * n)).

Return sizeof (gentypen) bytes of data read from address (p + (offset * n)).

For each component of a vector type, result[i] = if MSB of c[i] is set ? b[i] : a[i].

Each bit of the result is the corresponding bit of a if the corresponding bit of c is 0.

Returns 1 if the most significant bit in all components of x is set; otherwise returns 0.

Returns 1 if the most significant bit in any component of x is set; otherwise returns 0.

Test for sign bit.

Test if arguments are unordered.

Test if arguments are ordered.

Test for a normal value.

Test for a NaN.

Test for infinity value (+ve or –ve) .

Test for finite value.

Returns the component-wise compare of (x < y) || (x > y) .

Returns the component-wise compare of x <= y.

Returns the component-wise compare of x < y.

Returns the component-wise compare of x >= y.

Returns the component-wise compare of x > y.

Returns the component-wise compare of x != y.

intn isequal (floatn x, floatn y) Returns the component-wise compare of x == y.

Returns the length of vector p computed as: half_sqrt(p.x2 + p.y2 + ...)

Returns fast_length(p0 – p1).

Returns a vector in the same direction as p but with a length of 1.

Return the length of vector p, i.e., sqrt(p.x2 + p.y 2 + ...)

Returns the distance between p0 and p1.

Compute dot product.

Returns the cross product of p0.xyz and p1.xyz.

Returns 1.0 if x > 0, -0.0 if x = -0.0, +0.0 if x = +0.0, or –1.0 if x < 0.

Returns 0.0 if x <= edge0 and 1.0 if x >= edge1 and performs smooth Hermite interpolation between 0 and 1when edge0 < x < edge1.

Returns 0.0 if x < edge, otherwise it returns 1.0.

Converts degrees to radians, i.e.

Returns the linear blend of x & y implemented as: x + (y – x) ∗ a a must be a value in the range 0.0 . . . 1.0.

Converts radians to degrees, i.e.

Returns fmin(fmax(x, minval), maxval).

Multiply two 24-bit integer values x and y.

Multiply two 24-bit integer values x and y and add the 32-bit integer result to the 32-bit integer z.

result[i] = ((long)hi[i] << 32) | lo[i] result[i] = ((ulong)hi[i] << 32) | lo[i]

result[i] = ((int)hi[i] << 16) | lo[i] result[i] = ((uint)hi[i] << 16) | lo[i]

result[i] = ((short)hi[i] << 8) | lo[i] result[i] = ((ushort)hi[i] << 8) | lo[i]

Returns x - y and saturates the result.

For each element in v, the bits are shifted left by the number of bits given by the corresponding element in i (subject to usual shift modulo rules described in section 6.3).

Computes x * y and returns the high half of the product of x and y.

Returns a * b + c and saturates the result.

Returns mul_hi(a, b) + c.

Returns the number of leading 0-bits in x, starting at the most significant bit position.

Returns min(max(x, minval), maxval).

Returns (x + y + 1) >> 1.

Returns (x + y) >> 1.

Returns x + y and saturates the result.

Returns | x – y | without modulo overflow.

Returns | x |.

Compute tangent over an implementation-defined range.

Compute square root over an implementation-defined range.

Compute sine over an implementation-defined range.

Compute inverse square root over an implementationdefined range.

Compute reciprocal over an implementation-defined range.

Compute a base 10 logarithm over an implementationdefined range.

Compute a base 2 logarithm over an implementationdefined range.

Compute natural logarithm over an implementationdefined range.

Compute the base- 10 exponential of x over an implementation-defined range.

Compute the base- 2 exponential of x over an implementation-defined range.

Compute the base- e exponential of x over an implementation-defined range.

Compute x / y over an implementation-defined range.

Compute cosine over an implementation-defined range.

Compute reciprocal.

Compute the base- 10 exponential of x.

Compute the base- 2 exponential of x.

Compute x / y.

Round to integral value using the round to zero rounding mode.

Compute the gamma function.

Compute tan (PI $*$ x).

Compute hyperbolic tangent.

Compute tangent.

Compute square root.

Compute sin (PI $*$ x).

Compute hyperbolic sine.

Compute sine and cosine of x.

Compute sine.

Compute inverse square root.

Return the integral value nearest to x rounding halfway cases away from zero, regardless of the current rounding direction.

Compute x to the power 1/y.

Round to integral value (using round to nearest even rounding mode) in floating-point format.

The remquo function computes the value r such that r = x - n$*$y, where n is the integer nearest the exact value of x/y.

Compute the value r such that r = x - n$*$y, where n is the integer nearest the exact value of x/y.

Compute x to the power y, where x is $>=$ 0.

Compute x to the power y, where y is an integer.

Compute x to the power y.

Computes the next representable single-precision floating-point value following x in the direction of y.

Returns a quiet NaN.

Decompose a floating-point number.

Returns x if $|$ x $|$ $<$ $|$ y $|$, y if $|$ y $|$ $<$ $|$ x $|$, otherwise fmin(x, y).

Returns x if $|$ x $|$ $>$ $|$ y $|$, y if $|$ y $|$ $>$ $|$ x $|$, otherwise fmax(x, y).

mad approximates a $*$ b + c.

Compute the exponent of x, which is the integral part of logr $|$ x $|$.

Compute a base e logarithm of (1.0 + x).

Compute a base 10 logarithm.

Compute a base 2 logarithm.

Compute natural logarithm.

Log gamma function.

Multiply x by 2 to the power n.

Return the exponent as an integer value.

Compute the value of the square root of x^2+ y^2 without undue overflow or underflow.

Extract mantissa and exponent from x.

Returns fmin(x – floor (x), 0x1.fffffep-1f).

Modulus.

Returns y if y $<$ x, otherwise it returns x.

Returns y if x $<$ y, otherwise it returns x.

Returns the correctly rounded floating-point representation of the sum of c with the infinitely precise product of a and b.

Round to integral value using the round to –ve infinity rounding mode.

x - y if x $>$ y, +0 if x is less than or equal to y.

Compute absolute value of a floating-point number.

Compute e^x- 1.0.

Exponential base 10 function.

Exponential base 2 function.

Compute the base- e exponential of x.

Error function encountered in integrating the normal distribution.

Complementary error function.

Compute cos (PI $*$ x).

Compute hyperbolic consine.

Compute cosine.

Returns x with its sign changed to match the sign of y.

Round to integral value using the round to +ve infinity rounding mode.

Compute cube-root.

Compute atan2 (y, x) / PI.

Compute atan (x) / PI.

Hyperbolic arc tangent.

Arc tangent of y / x.

Arc tangent function.

Compute asin (x) / PI.

Inverse hyperbolic sine.

Arc sine function.

Compute acos (x) / PI.

Inverse hyperbolic cosine.

Arc cosine function.

get_global_offset returns the offset values specified in global_work_offset argument to clEnqueueNDRangeKernel.

get_group_id returns the work-group ID which is a number from 0 .

Returns the number of work-groups that will execute a kernel for dimension identified by dimindx.

Returns the unique local work-item ID i.e.

Returns the number of local work-items specified in dimension identified by dimindx.

Returns the unique global work-item ID value for dimension identified by dimindx.

Returns the number of global work-items specified for dimension identified by dimindx.

This is the value given to the work_dim argument specified in clEnqueueNDRangeKernel. For clEnqueueTask, this returns 1.

This value is given by the global_work_size argument to clEnqueueNDRangeKernel. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_global_size() returns 1. For clEnqueueTask, this always returns 1.

The global work-item ID specifies the work-item ID based on the number of global work-items specified to execute the kernel. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_global_id() returns 0. For clEnqueueTask, this returns 0.

This value is given by the local_work_size argument to clEnqueueNDRangeKernel if local_work_size is not NULL; otherwise the OpenCL implementation chooses an appropriate local_work_size value which is returned by this function. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_local_size() returns 1. For clEnqueueTask, this always returns 1.

a work-item within a specific work-group for dimension identified by dimindx. Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_local_id() returns 0. For clEnqueueTask, this returns 0.

Valid values of dimindx are 0 to get_work_dim() – 1. For other values of dimindx, get_num_groups () returns 1. For clEnqueueTask, this always returns 1.

. get_num_groups(dimindx) – 1. Valid values of dimindx are 0 to get_work_dim() – 1. For other values, get_group↩
_id() returns 0. For clEnqueueTask, this returns 0.

Valid values of dimindx are 0 to get_work_dim() – 1. For other values, get_global_offset() returns 0. For cl↩ EnqueueTask, this returns 0.

Rounding of intermediate products shall not occur. Edge case behavior is per the IEEE 754-2008 standard.

If one argument is a NaN, fmax() returns the other argument. If both arguments are NaNs, fmax() returns a NaN.

If one argument is a NaN, fmin() returns the other argument. If both arguments are NaNs, fmin() returns a NaN.

Returns x – y ∗ trunc (x/y).

floor(x) is returned in iptr.

For each component the mantissa returned is a float with magnitude in the interval [1/2, 1) or 0. Each component of x equals mantissa returned ∗ 2^exp.

Returns the natural logarithm of the absolute value of the gamma function. The sign of the gamma function is returned in the signp argument of lgamma_r.

Whether or how the product of a ∗ b is rounded and how supernormal or subnormal intermediate products are handled is not defined. mad is intended to be used where speed is preferred over accuracy.

The modf function breaks the argument x into integral and fractional parts, each of which has the same sign as the argument. It stores the integral part in the object pointed to by iptr.

The nancode may be placed in the significand of the resulting NaN.

Thus, if y is less than x, nextafter() returns the largest representable floating-point number less than x.

If there are two integers closest to x/y, n shall be the even one. If r is zero, it is given the same sign as x.

If there are two integers closest to x/y, n shall be the even one. If r is zero, it is given the same sign as x. This is the same value that is returned by the remainder function. remquo also calculates the lower seven bits of the integral quotient x/y, and gives that value the same sign as x/y. It stores this signed value in the object pointed to by quo.

Refer to section 7.1 for description of rounding modes.

The computed sine is the return value and computed cosine is returned in cosval.

x must be in the range -2^16 ... +2^16.

x must be in the range -2^16 . . . +2^16.

x must be in the range -216 . . . +216.

The maximum error is implementation-defined.

The maximum error is implementation defined.

The maximum error is implementationdefined.

The range of x and y are implementation-defined. The maximum error is implementation-defined.

The intermediate sum does not modulo overflow.

Results are undefined if minval > maxval.

Bits shifted off the left side of the element are shifted back in from the right.

Refer to definition of mul24 to see how the 24-bit integer multiplication is performed.

x and y are 32-bit integers but only the low 24-bits are used to perform the multiplication. mul24 should only be used when values in x and y are in the range [- 2^23, 2^23-1] if x and y are signed integers and in the range [0, 2^24-1] if x and y are unsigned integers. If x and y are not in this range, the multiplication result is implementation-defined.

(180 / PI) ∗ radians.

If x and y are infinite or NaN, the return values are undefined.

If a is not in the range 0.0 ... 1.0, the return values are undefined.

(PI / 180) ∗ degrees.

This is useful in cases where you would want a threshold function with a smooth transition. This is equivalent to: gentype t; t = clamp ((x – edge0) / (edge1 – edge0), 0, 1); return t ∗ t ∗ (3 – 2 ∗ t); Results are undefined if edge0 >= edge1 or if x, edge0 or edge1 is a NaN.

Returns 0.0 if x is a NaN.

The w component of float4 result returned will be 0.0.

This is calculated as length(p0 – p1).

fast_normalize is computed as: p ∗ half_rsqrt (p.x^2 + p.y^2 + ...) The result shall be within 8192 ulps error from the infinitely precise result of if (all(p == 0.0f)) result = p; else result = p / sqrt (p.x^2 + p.y^2 + ...); with the following exceptions: 1) If the sum of squares is greater than FLT_MAX then the value of the floating-point values in the result vector are undefined. 2) If the sum of squares is less than FLT_MIN then the implementation may return back p. 3) If the device is in “denorms are flushed to zero” mode, individual operand elements with magnitude less than sqrt(FLT_MIN) may be flushed to zero before proceeding with the calculation.

isordered() takes arguments x and y, and returns the result isequal(x, x) && isequal(y, y).

isunordered() takes arguments x and y, returning non-zero if x or y is NaN, and zero otherwise.

The scalar version of the function returns a 1 if the sign bit in the float is set else returns 0. The vector version of the function returns the following for each component in floatn: a -1 if the sign bit in the float is set else returns 0.

Otherwise it is the corresponding bit of b.

For a scalar type, result = c ? b : a.

The address computed as (p + (offset ∗ n)) must be 8-bit aligned if gentype is char, uchar; 16-bit aligned if gentype is short, ushort; 32-bit aligned if gentype is int, uint, float; 64-bit aligned if gentype is long, ulong.

The data read is interpreted as a half value. The half value is converted to a float value and the float value is returned. The read address computed as (p + offset) must be 16-bit aligned.

The data read is interpreted as a halfn value. The halfn value read is converted to a floatn value and the floatn value is returned. The read address computed as (p + (offset ∗ n)) must be 16-bit aligned.

The half value is then written to address computed as (p + offset). The address computed as (p + offset) must be 16- bit aligned. vstore_half use the current rounding mode. The default current rounding mode is round to nearest even.

The halfn value is then written to address computed as (p + (offset ∗ n)). The address computed as (p + (offset ∗ n)) must be 16-bit aligned. vstore_halfn uses the current rounding mode. The default current rounding mode is round to nearest even.

The data read is interpreted as a halfn value. The halfn value read is converted to a floatn value and the floatn value is returned. The address computed as (p + (offset ∗ n)) must be aligned to sizeof (halfn) bytes. For n = 3, vloada_half3 reads a half3 from address (p + (offset ∗ 4)) and returns a float3. The address computed as (p + (offset ∗ 4)) must be aligned to sizeof (half) ∗ 4 bytes.

For n = 1, 2, 4, 8 and 16, the halfn value is written to the address computed as (p + (offset

- n)). The address computed as (p + (offset ∗ n)) must be aligned to sizeof (halfn) bytes. For n = 3, the half3 value is written to the address computed as (p + (offset ∗ 4)). The address computed as (p + (offset ∗ 4)) must be aligned to sizeof (half) ∗ 4 bytes. vstorea_halfn uses the current rounding mode. The default current rounding mode is round to nearest even.

  This means that loads and stores preceding the mem_fence will be committed to memory before any loads and stores following the mem_fence. The flags argument specifies the memory address space and can be set to a combination of the following literal values: CLK_LOCAL_MEM_FENCE CLK_GLOBAL_MEM_FENCE.

The flags argument specifies the memory address space and can be set to to a combination of the following literal values: CLK_LOCAL_MEM_FENCE CLK_GLOBAL_MEM_FENCE.

The async copy is performed by all work-items in a workgroup and this built-in function must therefore be encountered by all work-items in a workgroup executing the kernel with the same argument values; otherwise the results are undefined. Returns an event object that can be used by wait_group_events to wait for the async copy to finish. The event argument can also be used to associate the async_work_group_copy with a previous async copy allowing an event to be shared by multiple async copies; otherwise event should be zero. If event argument is non-zero, the event object supplied in event argument will be returned. This function does not perform any implicit synchronization of source data such as using a barrier before performing the copy.

The src_stride is the stride in elements for each gentype element read from src. The dst_stride is the stride in elements for each gentype element written to dst. The async gather is performed by all work-items in a work-group. This built-in function must therefore be encountered by all work-items in a work-group executing the kernel with the same argument values; otherwise the results are undefined. Returns an event object that can be used by wait_group_events to wait for the async copy to finish. The event argument can also be used to associate the async_work_group_strided_copy with a previous async copy allowing an event to be shared by multiple async copies; otherwise event should be zero. If event argument is non-zero, the event object supplied in event argument will be returned. This function does not perform any implicit synchronization of source data such as using a barrier before performing the copy.

The event objects specified in event_list will be released after the wait is performed. This function must be encountered by all workitems in a work-group executing the kernel with the same num_events and event objects specified in event_list; otherwise the results are undefined.

The prefetch instruction is applied to a work-item in a workgroup and does not affect the functional behavior of the kernel.

Compute (old + val) and store result at location pointed by p. The function returns old.

Compute (old - val) and store result at location pointed by p. The function returns old.

Returns old value.

Compute (old + 1) and store result at location pointed by p. The function returns old.

Compute (old - 1) and store result at location pointed by p. The function returns old.

Compute (old == cmp) ? val : old and store result at location pointed by p. The function returns old.

Compute min(old, val) and store minimum value at location pointed by p. The function returns old.

Compute max(old, val) and store maximum value at location pointed by p. The function returns old.

Compute (old & val) and store result at location pointed by p. The function returns old.

Compute (old | val) and store result at location pointed by p. The function returns old.

Compute (old ^ val) and store result at location pointed by p. The function returns old.

The size of each element in the mask must match the size of each element in the result. For shuffle, only the ilogb(2m-1) least significant bits of each mask element are considered. For shuffle2, only the ilogb(2m-1)+1 least significant bits of each mask element are considered. Other bits in the mask shall be ignored. The elements of the input vectors are numbered from left to right across one or both of the vectors. For this purpose, the number of elements in a vector is given by vec_step(gentypem). The shuffle mask operand specifies, for each element of the result vector, which element of the one or two input vectors the result element gets. Examples: uint4 mask = (uint4)(3, 2, 1, 0); float4 a; float4 r = shuffle(a, mask); // r.s0123 = a.wzyx uint8 mask = (uint8)(0, 1, 2, 3, 4, 5, 6, 7); float4 a, b; float8 r = shuffle2(a, b, mask); // r.s0123 = a.xyzw // r.s4567 = b.xyzw uint4 mask; float8 a; float4 b; b = shuffle(a, mask); Examples that are not valid are: uint8 mask; short16 a; short8 b; b = shuffle(a, mask); <- not valid

read_imagei and read_imageui return unnormalized signed integer and unsigned integer values respectively. Each channel will be stored in a 32-bit integer. read_imagei can only be used with image objects created with image_channel_data_type set to one of the following values: CL_SIGNED_INT8, CL_SIGNED_INT16 and CL_SIGNED_INT32. If the image_channel_data_type is not one of the above values, the values returned by read_imagei are undefined. read_imageui can only be used with image objects created with image_channel_data_type set to one of the following values: CL_UNSIGNED_INT8, CL_UNSIGNED_INT16 and CL_UNSIGNED_INT32. If the image_channel_data_type is not one of the above values, the values returned by read_imageui are undefined. The read_image{i|ui} calls support a nearest filter only. The filter_mode specified in sampler must be set to CLK_FILTER_NEAREST; otherwise the values returned are undefined. Furthermore, the read_image{i|ui} calls that take integer coordinates must use a sampler with normalized coordinates set to CLK_NORMALIZED_COORDS_FALSE and addressing mode set to CLK_ADDRESS_CLAMP_TO_EDGE, CLK_ADDRESS_CLAMP or CLK_ADDRESS_NONE; otherwise the values returned are undefined.

Appropriate data format conversion to the specified image format is done before writing the color value. x & y are considered to be unnormalized coordinates and must be in the range 0 ... image width - 1, and 0 ... image height - 1. write_imagef can only be used with image objects created with image_channel_data_type set to one of the pre-defined packed formats or set to CL_SNORM_INT8, CL_UNORM_INT8, CL_SNORM_INT16, CL_UNORM_INT16, CL_HALF_FLOAT or CL_FLOAT. Appropriate data format conversion will be done to convert channel data from a floating-point value to actual data format in which the channels are stored. write_imagei can only be used with image objects created with image_channel_data_type set to one of the following values: CL_SIGNED_INT8, CL_SIGNED_INT16 and CL_SIGNED_INT32. write_imageui can only be used with image objects created with image_channel_data_type set to one of the following values: CL_UNSIGNED_INT8, CL_UNSIGNED_INT16 and CL_UNSIGNED_INT32. The behavior of write_imagef, write_imagei and write_imageui for image objects created with image_channel_data_type values not specified in the description above or with (x, y) coordinate values that are not in the range (0 ... image width - 1, 0 ... image height - 1), respectively, is undefined.

coord.w is ignored. read_imagef returns floating-point values in the range [0.0 ... 1.0] for image objects created with image_channel_data_type set to one of the predefined packed formats or CL_UNORM_INT8, or CL_UNORM_INT16. read_imagef returns floating-point values in the range [-1.0 ... 1.0] for image objects created with image_channel_data_type set to CL_SNORM_INT8, or CL_SNORM_INT16. read_imagef returns floating-point values for image objects created with image_channel_data_type set to CL_HALF_FLOAT or CL_FLOAT. The read_imagef calls that take integer coordinates must use a sampler with filter mode set to CLK_FILTER_NEAREST, normalized coordinates set to CLK_NORMALIZED_COORDS_FALSE and addressing mode set to CLK_ADDRESS_CLAMP_TO_EDGE, CLK_ADDRESS_CLAMP or CLK_ADDRESS_NONE; otherwise the values returned are undefined. Values returned by read_imagef for image objects with image_channel_data_type values not specified in the description are undefined.

coord.w is ignored. read_imagei and read_imageui return unnormalized signed integer and unsigned integer values respectively. Each channel will be stored in a 32-bit integer. read_imagei can only be used with image objects created with image_channel_data_type set to one of the following values: CL_SIGNED_INT8, CL_SIGNED_INT16 and CL_SIGNED_INT32. If the image_channel_data_type is not one of the above values, the values returned by read_imagei are undefined. read_imageui can only be used with image objects created with image_channel_data_type set to one of the following values: CL_UNSIGNED_INT8, CL_UNSIGNED_INT16 and CL_UNSIGNED_INT32. If the image_channel_data_type is not one of the above values, the values returned by read_imageui are undefined. The read_image{i|ui} calls support a nearest filter only. The filter_mode specified in sampler must be set to CLK_FILTER_NEAREST; otherwise the values returned are undefined. Furthermore, the read_image{i|ui} calls that take integer coordinates must use a sampler with normalized coordinates set to CLK_NORMALIZED_COORDS_FALSE and addressing mode set to CLK_ADDRESS_CLAMP_TO_EDGE, CLK_ADDRESS_CLAMP or CLK_ADDRESS_NONE; otherwise the values returned are undefined.

Valid values are: CLK_A CLK_R CLK_Rx CLK_RG CLK_RGx CLK_RA CLK_RGB CLK_RGBx CLK_RGBA CLK↩
_ARGB CLK_BGRA CLK_INTENSITY CLK_LUMINANCE

The width is returned in the x component, and the height in the y component.

The width is returned in the x component, height in the y component, depth in the z component and the w component is 0.

13.87.2.2 printf()

```
int printf (
              __constant const char * st,
               ... )
```

13.88 opencl_spir_req.h

Go to the documentation of this file.

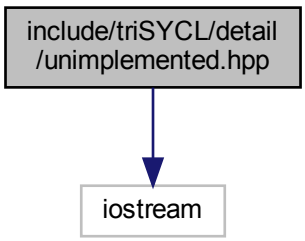
```
00001 #ifndef TRISYCL_SYCL_DETAIL_SPIR_OPENCL_SPIR_REQ_H
00002 #define TRISYCL_SYCL_DETAIL_SPIR_OPENCL_SPIR_REQ_H
00003
00004 /** \file This file is a temporary work around for integrating OpenCL builtins
00005 and intrinsics into triSYCL, it adds only what is needed from opencl_spir.h
00006 for the .cpp to .ll pre-kernel device compilation step.
00007
00008 This avoids the need to modify the compiler to include types not normally
00009 supported without the OpenCL flags active (half, clk_event_t etc.).
00010
00011 The intrinsics and components from this file are based on the opencl_spir.h
00012 file of the Khronos Group SPIR-Tools repository with some minor
00013 modifications:  https://github.com/KhronosGroup/SPIR-Tools
00014
00015 \todo Rather than have this file to work around OpenCL specific types modify
00016 the Clang/LLVM device compiler to include the OpenCL specific types and
00017 related processing when -sycl is used (or -sycl-is-device?)
00018 \todo OR rather than the above use the sycl specfic CL types in here instead
00019 e.g.  cl::sycl::cl_uint in place of unsigned int.
00020
00021 andrew point gozillon at yahoo point com
00022
00023 This file is distributed under the University of Illinois Open Source
00024 License.  See LICENSE.TXT for details.
00025 */
00026
00027 // The compiler doesn't know about size_t outside of the std namespace
00028 // is it better to define size_t as the std implementation or as our
00029 // own using size_t = closest_to_opencl_type?
00030 #include <cstddef>
00031
00032 /// Added according to pocl/examples/example1-spir64/generate_spir.sh
00033 /// \todo avoid this hack
00034 #pragma OPENCL EXTENSION cl_khr_fp64 :  enable
00035
00036 // Address space work around, these index values
00037 // seem to change frequently, so if there is an
00038 // error with address spaces this may be the culprit.
00039 // No address space is also the private/default
00040 // address space in OpenCL
00041 #define __global __attribute__((address_space(1)))
00042 #define __local __attribute__((address_space(2)))
00043 #define __constant __attribute__((address_space(3)))
00044 #define __private __attribute__((address_space(4)))
00045 #define __generic __attribute__((address_space(5)))
00046
00047 #define const_func __attribute__((const))
00048 #define readonly __attribute__((pure))
00049
00050 /**
00051 * Returns the unique local work-item ID i.e.  a work-item
00052 * within a specific work-group for dimension identified by
00053 * dimindx.  Valid values of dimindx are 0 to
00054 * get_work_dim() - 1.  For other values of dimindx,
00055 * get_local_id() returns 0.
00056 * For clEnqueueTask, this returns 0.
```

```
00057 */
00058 std::size_t const_func __attribute__((overloadable))
00059   get_local_id(unsigned int dimindx);
00060
00061 /**
00062 * Returns the unique global work-item ID value for
00063 * dimension identified by dimindx.  The global work-item
00064 * ID specifies the work-item ID based on the number of
00065 * global work-items specified to execute the kernel.  Valid
00066 * values of dimindx are 0 to get_work_dim() - 1.  For
00067 * other values of dimindx, get_global_id() returns 0.
00068 * For clEnqueueTask, this returns 0.
00069 */
00070 std::size_t const_func __attribute__((overloadable))
00071   get_global_id(unsigned int dimindx);
00072
00073 /**
00074 * Returns the number of global work-items specified for
00075 * dimension identified by dimindx.  This value is given by
00076 * the global_work_size argument to
00077 * clEnqueueNDRangeKernel.  Valid values of dimindx
00078 * are 0 to get_work_dim() - 1.  For other values of
00079 * dimindx, get_global_size() returns 1.
00080 * For clEnqueueTask, this always returns 1.
00081 */
00082 std::size_t const_func __attribute__((overloadable))
00083   get_global_size(unsigned int dimindx);
00084
00085 /**
00086 * Returns the number of local work-items specified in
00087 * dimension identified by dimindx.  This value is given by
00088 * the local_work_size argument to
00089 * clEnqueueNDRangeKernel if local_work_size is not
00090 * NULL; otherwise the OpenCL implementation chooses
00091 * an appropriate local_work_size value which is returned
00092 * by this function.  Valid values of dimindx are 0 to
00093 * get_work_dim() - 1.  For other values of dimindx,
00094 * get_local_size() returns 1.
00095 * For clEnqueueTask, this always returns 1.
00096 */
00097 std::size_t const_func __attribute__((overloadable))
00098   get_local_size(unsigned int dimindx);
00099
00100 /**
00101 * Returns the number of work-groups that will execute a
00102 * kernel for dimension identified by dimindx.
00103 * Valid values of dimindx are 0 to get_work_dim() - 1.
00104 * For other values of dimindx, get_num_groups () returns
00105 * 1.
00106 * For clEnqueueTask, this always returns 1.
00107 */
00108 std::size_t const_func __attribute__((overloadable))
00109   get_num_groups(unsigned int dimindx);
00110
00111 /**
00112 * get_group_id returns the work-group ID which is a
00113 * number from 0 ..  get_num_groups(dimindx) - 1.
00114 * Valid values of dimindx are 0 to get_work_dim() - 1.
00115 * For other values, get_group_id() returns 0.
00116 * For clEnqueueTask, this returns 0.
00117 */
00118 std::size_t const_func __attribute__((overloadable))
00119   get_group_id(unsigned int dimindx);
00120
00121 /**
00122 * get_global_offset returns the offset values specified in
00123 * global_work_offset argument to
00124 * clEnqueueNDRangeKernel.
00125 * Valid values of dimindx are 0 to get_work_dim() - 1.
00126 * For other values, get_global_offset() returns 0.
00127 * For clEnqueueTask, this returns 0.
00128 */
00129 std::size_t const_func __attribute__((overloadable))
00130   get_global_offset(unsigned int dimindx);
00131
00132 // Doesn't appear to work on device at the moment
00133 int printf(__constant const char* st, ...);
00134
00135 #endif // TRISYCL_SYCL_DETAIL_SPIR_OPENCL_SPIR_REQ_H
```

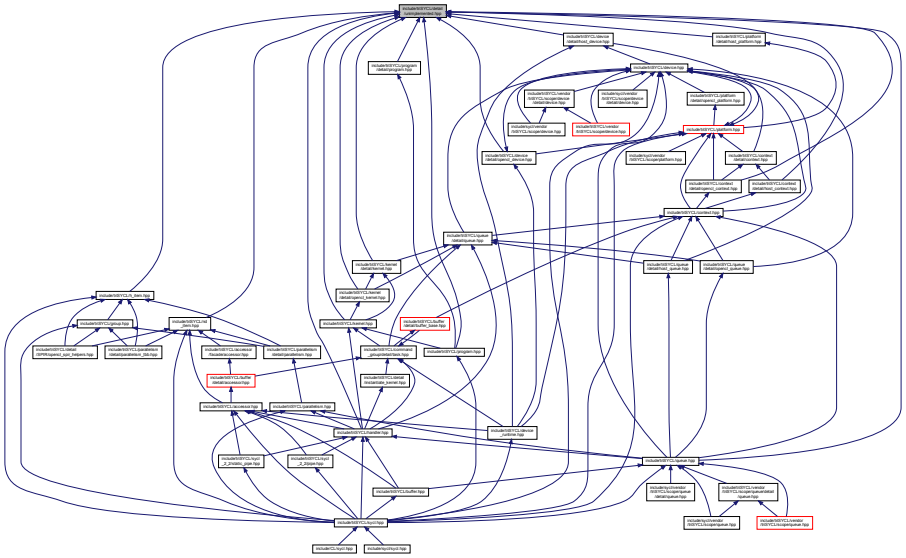
13.89 include/triSYCL/detail/unimplemented.hpp File Reference

```
#include <iostream>
```

Include dependency graph for unimplemented.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Macros

- #define TRISYCL_UNIMPL trisycl::detail::unimplemented(__func__, __FILE__, __LINE__)

Functions

- void trisycl::detail::unimplemented (const char *func, const char *file, int line)

  Display an "unimplemented" message.

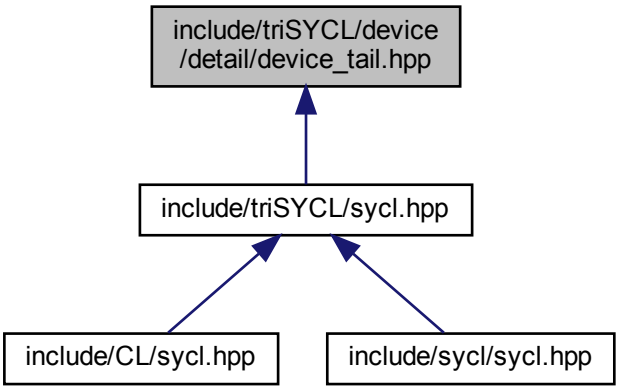
13.90 unimplemented.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_UNIMPLEMENTED_HPP
00002 #define TRISYCL_SYCL_DETAIL_UNIMPLEMENTED_HPP
00003
00004 /** \file Deal with unimplemented features
00005 Ronan at Keryell point FR
00006
00007 This file is distributed under the University of Illinois Open Source
00008 License.  See LICENSE.TXT for details.
00009 */
00010
00011 #include <iostream>
00012
00013 namespace trisycl::detail {
00014
00015 /** \addtogroup helpers Some helpers for the implementation
00016 @{
00017 */
00018
00019 /** Display an "unimplemented" message
00020
00021 Can be changed to call assert(0) or whatever.
00022 */
00023 inline void unimplemented(const char *func, const char *file, int line) {
00024 #ifndef NDEBUG
00025   std::cerr « "Error:  using a non implemented feature :  " « func « std::endl
00026             « "at " « file « ":" « line « std::endl
00027             « "Please contribute to the open source implementation.  :-)"
00028             « std::endl;
00029 #endif
00030 }
00031
00032 #define TRISYCL_UNIMPL trisycl::detail::unimplemented(__func__, __FILE__, __LINE__)
00033 /// @} End the helpers Doxygen group
00034
00035 }
00036
00037 /*
00038 # Some Emacs stuff:
00039 ### Local Variables:
00040 ### ispell-local-dictionary:  "american"
00041 ### eval:  (flyspell-prog-mode)
00042 ### End:
00043 */
00044
00045 #endif // TRISYCL_SYCL_DETAIL_UNIMPLEMENTED_HPP
```

13.91 include/triSYCL/device/detail/device_tail.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

13.92 device_tail.hpp

Go to the documentation of this file.

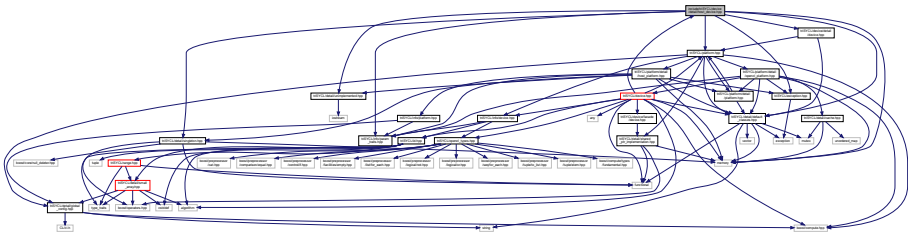
```
00001 #ifndef TRISYCL_SYCL_DEVICE_DETAIL_DEVICE_TAIL_HPP
00002 #define TRISYCL_SYCL_DEVICE_DETAIL_DEVICE_TAIL_HPP
00003
00004 /** \file The ending part of of OpenCL SYCL device
00005
00006 This is here to break a dependence between device and device_selector
00007
00008 Ronan at Keryell point FR
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 namespace trisycl {
00015
00016 /** \addtogroup execution Platforms, contexts, devices and queues
00017 @{
00018 */
00019
00020 /** Return a list of all available devices
00021
00022 Return synchronous errors via SYCL exception classes.
00023 */
00024 vector_class<device>
00025 device::get_devices(info::device_type device_type) {
00026   // Start with the default device
00027   vector_class<device> devices = { {} };
00028
00029 #ifdef TRISYCL_OPENCL
00030   // Then add all the OpenCL devices
00031   for (const auto &d :  boost::compute::system::devices())
```

```
00032     devices.emplace_back(d);
00033 #endif
00034
00035   // The selected devices
00036   vector_class<device> sd;
00037   device_type_selector s { device_type };
00038
00039   // Return the devices with the good criterion according to the selector
00040   std::copy_if(devices.begin(), devices.end(), std::back_inserter(sd),
00041                [&](const device &e) { return s(e) >= 0; });
00042   return sd;
00043 }
00044
00045 /// @} to end the Doxygen group
00046
00047 }
00048
00049 /*
00050 # Some Emacs stuff:
00051 ### Local Variables:
00052 ### ispell-local-dictionary:  "american"
00053 ### eval:  (flyspell-prog-mode)
00054 ### End:
00055 */
00056
00057 #endif // TRISYCL_SYCL_DEVICE_DETAIL_DEVICE_TAIL_HPP
```

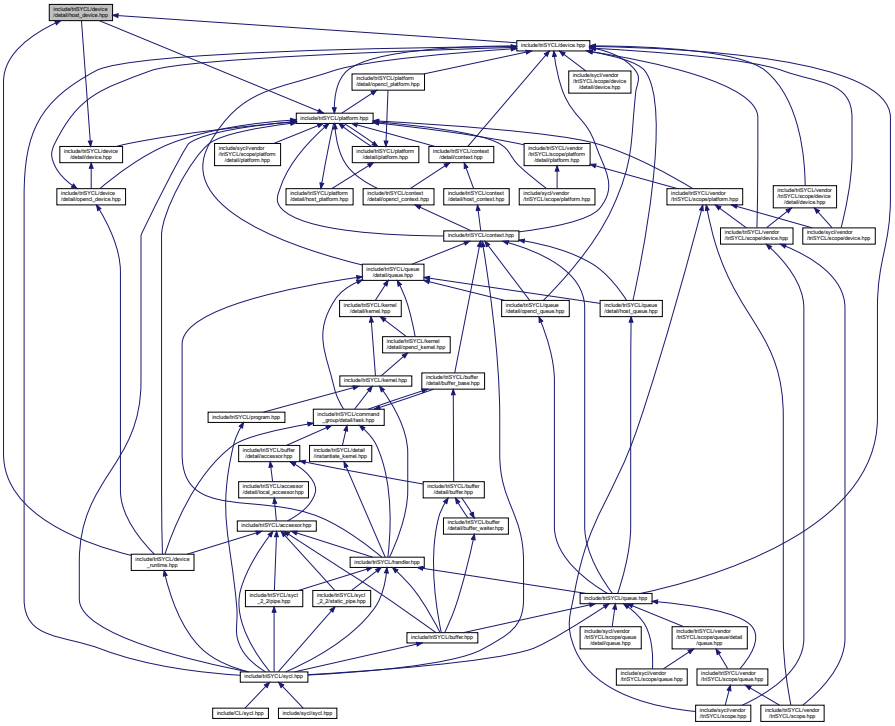
13.93 include/triSYCL/device/detail/host_device.hpp File Reference

```
#include <memory>
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/detail/singleton.hpp"
#include "triSYCL/detail/unimplemented.hpp"
#include "triSYCL/device/detail/device.hpp"
#include "triSYCL/exception.hpp"
#include "triSYCL/info/param_traits.hpp"
#include "triSYCL/platform.hpp"
```

Include dependency graph for host_device.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::host_device

  SYCL host device.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Macros

- #define TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(name, result) case (info::device::name) ↩ : return (result);

  Query the device for OpenCL info::device info.

13.93.1 Macro Definition Documentation

13.93.1.1 TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE

```
#define TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(
              name,
              result )  case (info::device::name) :  return (result);
```

Query the device for OpenCL info::device info.

Return synchronous errors via the SYCL exception class.

Todo To be implemented

Definition at line 97 of file host_device.hpp.

13.94 host_device.hpp

Go to the documentation of this file.

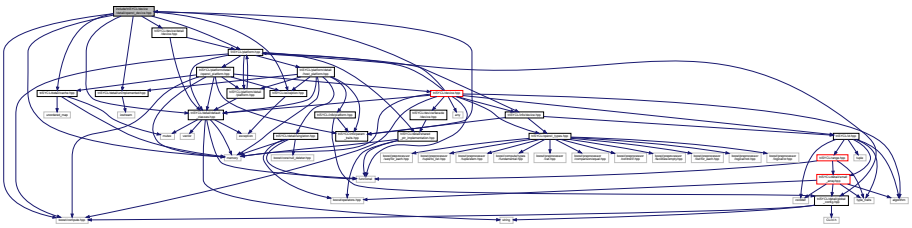
```
00001 #ifndef TRISYCL_SYCL_DEVICE_DETAIL_HOST_DEVICE_HPP
00002 #define TRISYCL_SYCL_DEVICE_DETAIL_HOST_DEVICE_HPP
00003
00004 /** \file The OpenCL SYCL host device implementation
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <memory>
00013
00014 #include "triSYCL/detail/default_classes.hpp"
00015
00016 #include "triSYCL/detail/singleton.hpp"
00017 #include "triSYCL/detail/unimplemented.hpp"
00018 #include "triSYCL/device/detail/device.hpp"
00019 #include "triSYCL/exception.hpp"
00020 #include "triSYCL/info/param_traits.hpp"
00021 #include "triSYCL/platform.hpp"
00022
00023 namespace trisycl::detail {
00024
00025 /** SYCL host device
00026
00027 \todo The implementation is quite minimal for now.  :-)
00028 */
00029 class host_device :  public detail::device,
00030                      public detail::singleton<host_device> {
00031
00032 public:
00033
00034 #ifdef TRISYCL_OPENCL
00035 /** Return the cl_device_id of the underlying OpenCL platform
00036
00037 This throws an error since there is no OpenCL device associated
00038 to the host device.
00039 */
00040   cl_device_id get()const override {
00041     throw non_cl_error("The host device has no OpenCL device");
00042   }
00043
00044
00045 /** Return the underlying Boost.Compute device
00046
00047 This throws an error since there is no OpenCL device associated
00048 to the host device.
00049 */
00050   boost::compute::device &get_boost_compute()override {
00051     throw non_cl_error("The host device has no underlying OpenCL device");
00052   }
00053 #endif
00054
00055
```

```
00056 /// Return true since the device is a SYCL host device
00057   bool is_host()const override {
00058     return true;
00059   }
00060
00061
00062 /// Return false since the host device is not an OpenCL CPU device
00063   bool is_cpu()const override {
00064     return false;
00065   }
00066
00067
00068 /// Return false since the host device is not an OpenCL GPU device
00069   bool is_gpu()const override {
00070     return false;
00071   }
00072
00073
00074 /// Return false since the host device is not an OpenCL accelerator device
00075   bool is_accelerator()const override {
00076     return false;
00077   }
00078
00079
00080 /** Return the platform of device
00081
00082 Return synchronous errors via the SYCL exception class.
00083
00084 \todo To be implemented
00085 */
00086   trisycl::platform get_platform()const override {
00087     TRISYCL_UNIMPL;
00088     return {};
00089   }
00090
00091 /** Query the device for OpenCL info::device info
00092
00093 Return synchronous errors via the SYCL exception class.
00094
00095 \todo To be implemented
00096 */
00097 #define TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(name, result) \
00098 case (info::device::name) :  return (result);
00099   std::any
00100   get_info(info::device param)const override {
00101     switch (param) {
00102     TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(max_work_group_size, static_cast<std::size_t>(8))
00103     TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(max_work_item_sizes, (trisycl::id<3>{ 128, 128, 128 }))
00104     TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(max_compute_units, static_cast<trisycl::cl_uint>(8))
00105     TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(device_type, info::device_type::host)
00106     TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(local_mem_type, info::local_mem_type::global)
00107     TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE(local_mem_size, static_cast<trisycl::cl_ulong>(32768))
00108     default:
00109       return 0;
00110     }
00111   }
00112
00113 /** Specify whether a specific extension is supported on the device
00114
00115 \todo To be implemented
00116 */
00117   bool has_extension(const string_class &extension)const override {
00118     TRISYCL_UNIMPL;
00119     return {};
00120   }
00121
00122
00123 };
00124
00125 /// @} to end the execution Doxygen group
00126
00127 }
00128
00129 /*
00130 # Some Emacs stuff:
00131 ### Local Variables:
00132 ### ispell-local-dictionary:  "american"
00133 ### eval:  (flyspell-prog-mode)
00134 ### End:
00135 */
00136
00137 #endif // TRISYCL_SYCL_DEVICE_DETAIL_HOST_DEVICE_HPP
```

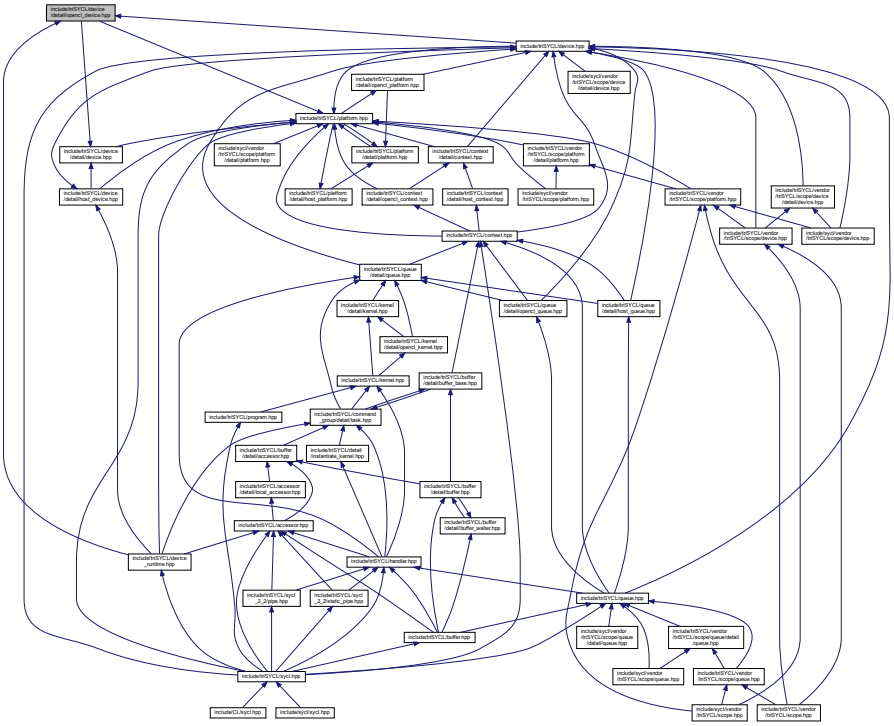
13.95 include/triSYCL/device/detail/opencl_device.hpp File Reference

```
#include <memory>
#include <boost/compute.hpp>
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/detail/cache.hpp"
#include "triSYCL/detail/unimplemented.hpp"
#include "triSYCL/device/detail/device.hpp"
#include "triSYCL/exception.hpp"
#include "triSYCL/info/param_traits.hpp"
#include "triSYCL/platform.hpp"
```

Include dependency graph for opencl_device.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::opencl_device

  SYCL OpenCL device.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.96 opencl_device.hpp

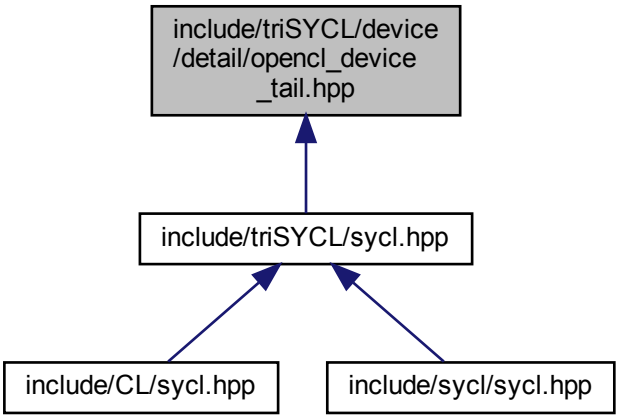
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DEVICE_DETAIL_OPENCL_DEVICE_HPP
00002 #define TRISYCL_SYCL_DEVICE_DETAIL_OPENCL_DEVICE_HPP
00003
00004 /** \file The SYCL OpenCL device implementation
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <memory>
00013
00014 #include <boost/compute.hpp>
00015
00016 #include "triSYCL/detail/default_classes.hpp"
00017
00018 #include "triSYCL/detail/cache.hpp"
00019 #include "triSYCL/detail/unimplemented.hpp"
00020 #include "triSYCL/device/detail/device.hpp"
00021 #include "triSYCL/exception.hpp"
00022 #include "triSYCL/info/param_traits.hpp"
00023 #include "triSYCL/platform.hpp"
00024
00025 namespace trisycl::detail {
00026
00027 /// SYCL OpenCL device
00028 class opencl_device :  public detail::device {
00029
00030 /// Use the Boost Compute abstraction of the OpenCL device
00031   boost::compute::device d;
00032
00033 /** A cache to always return the same alive device for a given
00034 OpenCL device
00035
00036 C++11 guaranties the static construction is thread-safe
00037 */
00038   static detail::cache<cl_device_id, detail::opencl_device> cache;
00039
00040 public:
00041
00042 /// Return the cl_device_id of the underlying OpenCL device
00043   cl_device_id get()const override {
00044     return d.id();
00045   }
00046
00047
00048 /// Return the underlying Boost.Compute device
00049   boost::compute::device &get_boost_compute()override {
00050     return d;
00051   }
00052
00053
00054 /// Return false since an OpenCL device is not the SYCL host device
00055   bool is_host()const override {
00056     return false;
00057   }
00058
00059
00060 /// Test if the OpenCL is a CPU device
00061   bool is_cpu()const override {
00062     // Even in Boost.Compute the type is a bit-field, so use & instead of ==
00063     return d.type() & boost::compute::device::cpu;
00064   }
00065
00066
00067 /// Test if the OpenCL is a GPU device
00068   bool is_gpu()const override {
```

```
00069     // Even in Boost.Compute the type is a bit-field, so use & instead of ==
00070     return d.type() & boost::compute::device::gpu;
00071   }
00072
00073
00074 /// Test if the OpenCL is an accelerator device
00075   bool is_accelerator()const override {
00076     // Even in Boost.Compute the type is a bit-field, so use & instead of ==
00077     return d.type() & boost::compute::device::accelerator;
00078   }
00079
00080
00081 /** Return the platform of device
00082
00083 Return synchronous errors via the SYCL exception class.
00084 */
00085   trisycl::platform get_platform()const override {
00086     return d.platform();
00087   }
00088
00089 /** Query the device for OpenCL info::device info
00090
00091 Return synchronous errors via the SYCL exception class.
00092 */
00093   inline std::any get_info(info::device param) const override;
00094
00095 /** Specify whether a specific extension is supported on the device.
00096 */
00097   bool has_extension(const string_class &extension)const override {
00098     return d.supports_extension(extension);
00099   }
00100
00101
00102 ///// Get a singleton instance of the opencl_device
00103   static std::shared_ptr<opencl_device>
00104   instance(const boost::compute::device &d) {
00105     return cache.get_or_register(d.id(),
00106                                  [&] { return new opencl_device { d }; });
00107   }
00108
00109 private:
00110
00111   template <info::device param>
00112   inline std::any get_info() const;
00113 /// Only the instance factory can build it
00114   opencl_device(const boost::compute::device &d) :  d { d } {}
00115
00116 public:
00117
00118 /// Unregister from the cache on destruction
00119   ~opencl_device()override {
00120     cache.remove(d.id());
00121   }
00122
00123 };
00124
00125 /* Allocate the cache here but since this is a pure-header library,
00126 use a weak symbol so that only one remains when SYCL headers are
00127 used in different compilation units of a program
00128 */
00129 TRISYCL_WEAK_ATTRIB_PREFIX
00130 detail::cache<cl_device_id, detail::opencl_device> opencl_device::cache
00131 TRISYCL_WEAK_ATTRIB_SUFFIX;
00132
00133 }
00134
00135 /*
00136 # Some Emacs stuff:
00137 ### Local Variables:
00138 ### ispell-local-dictionary:  "american"
00139 ### eval:  (flyspell-prog-mode)
00140 ### End:
00141 */
00142
00143 #endif // TRISYCL_SYCL_DEVICE_DETAIL_OPENCL_DEVICE_HPP
```

13.97 include/triSYCL/device/detail/opencl_device_tail.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Macros

- #define TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(name, result) case info::device::name : return result;
- #define TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(name, field) case info::device::name : return d.get_info<CL_DEVICE_##field>();

13.97.1 Macro Definition Documentation

13.97.1.1 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE

```
#define TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(
              name,
              field )  case info::device::name :  return d.get_info<CL_DEVICE_##field>();
```

Definition at line 9 of file opencl_device_tail.hpp.

13.97.1.2 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE

```
#define TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(
              name,
              result )  case info::device::name :  return result;
```

Definition at line 6 of file opencl_device_tail.hpp.

13.98 opencl_device_tail.hpp

Go to the documentation of this file.

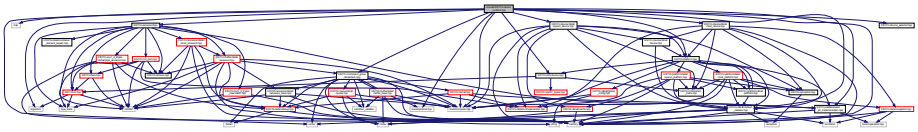
```
00001 #ifndef TRISYCL_SYCL_DEVICE_DETAIL_OPENCL_DEVICE_TAIL_HPP
00002 #define TRISYCL_SYCL_DEVICE_DETAIL_OPENCL_DEVICE_TAIL_HPP
00003
00004 namespace trisycl::detail {
00005
00006 #define TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(name, result) \
00007 case info::device::name :  return result;
00008
00009 #define TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(name, field)         \
00010 case info::device::name :  return d.get_info<CL_DEVICE_##field>();
00011
00012 inline std::any opencl_device::get_info(info::device param)const {
00013   switch (param) {
00014 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(
00015   device_type,
00016   d.get_info<CL_DEVICE_TYPE>() == CL_DEVICE_TYPE_CPU
00017   ?  info::device_type::cpu :  info::device_type::gpu);
00018 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(vendor_id, VENDOR_ID)
00019 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_compute_units, MAX_COMPUTE_UNITS )
00020 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_work_item_dimensions,
00021                                        MAX_WORK_ITEM_DIMENSIONS)
00022 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(
00023   max_work_item_sizes,
00024   (::trisycl::id<3> { d.get_info<CL_DEVICE_MAX_WORK_ITEM_SIZES>()[0]
00025                     , d.get_info<CL_DEVICE_MAX_WORK_ITEM_SIZES>()[1]
00026                     , d.get_info<CL_DEVICE_MAX_WORK_ITEM_SIZES>()[2] }))
00027 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_work_group_size, MAX_WORK_GROUP_SIZE)
00028
00029 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_vector_width_char,
00030                                        PREFERRED_VECTOR_WIDTH_CHAR)
00031 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_vector_width_short,
00032                                        PREFERRED_VECTOR_WIDTH_SHORT)
00033 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_vector_width_int,
00034                                        PREFERRED_VECTOR_WIDTH_INT)
00035 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_vector_width_long,
00036                                        PREFERRED_VECTOR_WIDTH_LONG)
00037 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_vector_width_float,
00038                                        PREFERRED_VECTOR_WIDTH_FLOAT)
00039 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_vector_width_double,
00040                                        PREFERRED_VECTOR_WIDTH_DOUBLE)
00041 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(preferred_vector_width_half,
00042                                     ::trisycl::cl_uint { 0 })
00043
00044 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(native_vector_width_char,
00045                                        NATIVE_VECTOR_WIDTH_CHAR)
00046 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(native_vector_width_short,
00047                                        NATIVE_VECTOR_WIDTH_SHORT)
00048 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(native_vector_width_int,
00049                                        NATIVE_VECTOR_WIDTH_INT)
00050 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(native_vector_width_long,
00051                                        NATIVE_VECTOR_WIDTH_LONG)
00052 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(native_vector_width_float,
00053                                        NATIVE_VECTOR_WIDTH_FLOAT)
00054 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(native_vector_width_double,
00055                                        NATIVE_VECTOR_WIDTH_DOUBLE)
00056 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(native_vector_width_half,
00057                                     ::trisycl::cl_uint { 0 })
00058
00059 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_clock_frequency, MAX_CLOCK_FREQUENCY)
00060 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(address_bits, ADDRESS_BITS)
00061 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_mem_alloc_size, MAX_MEM_ALLOC_SIZE)
00062 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(image_support, IMAGE_SUPPORT )
00063 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_read_image_args, MAX_READ_IMAGE_ARGS)
00064 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_write_image_args,
00065                                        MAX_WRITE_IMAGE_ARGS)
00066 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(image2d_max_width, IMAGE2D_MAX_WIDTH)
```

```
00067 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(image2d_max_height, IMAGE2D_MAX_HEIGHT)
00068 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(image3d_max_width, IMAGE3D_MAX_WIDTH)
00069 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(image3d_max_height, IMAGE3D_MAX_HEIGHT)
00070 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(image3d_max_depth, IMAGE3D_MAX_DEPTH)
00071 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(image_max_buffer_size, size_t { 0 })
00072 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(image_max_array_size, size_t { 0 })
00073 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_samplers, MAX_SAMPLERS)
00074 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(max_parameter_size, MAX_PARAMETER_SIZE)
00075 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(mem_base_addr_align, MEM_BASE_ADDR_ALIGN)
00076 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(half_fp_config,
00077                                     vector_class<info::fp_config>{} )
00078 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(single_fp_config,
00079                                     vector_class<info::fp_config>{} )
00080 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(double_fp_config,
00081                                     vector_class<info::fp_config>{} )
00082 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(global_mem_cache_type,
00083                                     info::global_mem_cache_type::none )
00084 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(
00085   global_mem_cache_line_size,
00086   ::trisycl::cl_uint { static_cast<::trisycl::cl_uint>
00087       (d.get_info<CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE>()) })
00088 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(global_mem_cache_size,
00089                                        GLOBAL_MEM_CACHE_SIZE)
00090 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(global_mem_size, GLOBAL_MEM_SIZE)
00091 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(max_constant_buffer_size,
00092                                     ::trisycl::cl_ulong { 0 } )
00093 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(max_constant_args,
00094                                     ::trisycl::cl_uint { 0 } )
00095 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(local_mem_type, info::local_mem_type::local)
00096 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(local_mem_size, LOCAL_MEM_SIZE)
00097 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(error_correction_support,
00098                                        ERROR_CORRECTION_SUPPORT)
00099 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(host_unified_memory,
00100                                        HOST_UNIFIED_MEMORY)
00101 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(profiling_timer_resolution,
00102                                        PROFILING_TIMER_RESOLUTION)
00103 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(is_endian_little, ENDIAN_LITTLE)
00104 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(is_available, AVAILABLE)
00105 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(is_compiler_available,
00106                                        COMPILER_AVAILABLE)
00107 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(is_linker_available, false)
00108 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(execution_capabilities,
00109                                     vector_class<info::execution_capability>{} )
00110 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(queue_profiling, false)
00111 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(built_in_kernels,
00112                                     vector_class<string_class>{} )
00113 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(platform, ::trisycl::platform{} )
00114 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(name, NAME)
00115 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(vendor, VENDOR)
00116 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(driver_version,
00117                                     d.get_info<CL_DRIVER_VERSION>())
00118 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(profile, PROFILE)
00119 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(version, VERSION)
00120 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(opencl_c_version, OPENCL_C_VERSION )
00121 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(extensions, vector_class<string_class>{} )
00122 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(printf_buffer_size, PRINTF_BUFFER_SIZE)
00123 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(preferred_interop_user_sync,
00124                                        PREFERRED_INTEROP_USER_SYNC)
00125 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(parent_device, ::trisycl::device{})
00126 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(partition_max_sub_devices,
00127                                     ::trisycl::cl_uint { 0 })
00128 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(partition_properties,
00129                                     vector_class<info::partition_property>{})
00130 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(
00131   partition_affinity_domains,
00132   vector_class<info::partition_affinity_domain>{})
00133 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(partition_type_property,
00134                                     info::partition_property::no_partition)
00135 TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE(
00136   partition_type_affinity_domain,
00137   info::partition_affinity_domain::not_applicable)
00138 TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE(reference_count, REFERENCE_COUNT)
00139   default:
00140     return 0;
00141   }
00142 }
00143
00144 }
00145
00146 #endif
```

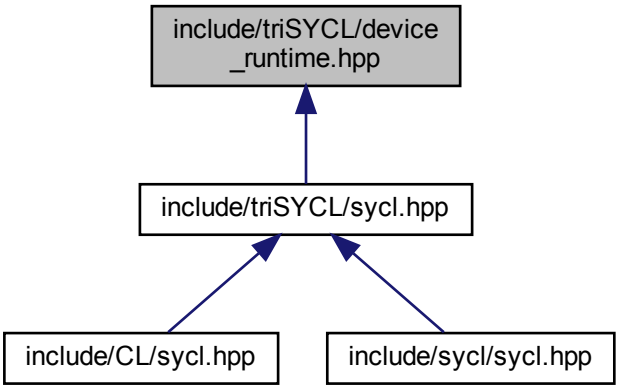
13.99 include/triSYCL/device_runtime.hpp File Reference

```
#include <algorithm>
#include <map>
#include <memory>
#include <vector>
#include <boost/compute.hpp>
#include <boost/optional.hpp>
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/accessor.hpp"
#include "triSYCL/command_group/detail/task.hpp"
#include "triSYCL/detail/shared_ptr_implementation.hpp"
#include "triSYCL/device/detail/host_device.hpp"
#include "triSYCL/device/detail/opencl_device.hpp"
#include "triSYCL/info/device.hpp"
#include "triSYCL/device_selector.hpp"
#include "triSYCL/platform.hpp"
```

Include dependency graph for device_runtime.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::drt::accessor< Accessor >

  SYCL accessor seen from a device perspective. *More...*
- struct trisycl::drt::code::program

  A binary program containing some kernels.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::drt

  The device-side runtime.
- namespace trisycl::drt::code

  The binary code of the kernels.

Functions

- TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX trisycl::drt::serialize_arg (detail::task &task, std::size_t index, void ∗arg, std::size_t arg_size)

  Set an argument value of the kernel.
- TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX trisycl::drt::serialize_accessor_arg (detail::task &task, std::size_t index, void ∗arg, std::size_t arg_size)

  Set an accessor argument of the kernel.
- TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX trisycl::drt::set_kernel (detail↩ ::task &task, const char ∗kernel_name, const char ∗kernel_short_name)

  Set the kernel for a later invocation.

Variables

- TRISYCL_WEAK_ATTRIB_PREFIX std::map< std::string, std::vector< unsigned char > > TRISYCL_WEAK_ATTRIB_SUFFIX trisycl::drt::kernel_IR

  The code of the kernels arranged by name.

13.100 device_runtime.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DEVICE_RUNTIME_HPP
00002 #define TRISYCL_SYCL_DEVICE_RUNTIME_HPP
00003
00004 /** \file The OpenCL SYCL device-side runtime
00005
00006 This code is expected to be called once some Clang & LLVM passes
00007 of the device compiler has been applied.
00008
00009 \todo Refactor to be more modular and OpenCL-independent so it can
00010 more easily be retargeted to some other runtime API.
00011
00012 Ronan at Keryell point FR
00013
00014 This file is distributed under the University of Illinois Open Source
00015 License.  See LICENSE.TXT for details.
00016 */
00017
00018 #include <algorithm>
00019 #include <map>
00020 #include <memory>
00021 #include <vector>
00022
00023 #ifdef TRISYCL_OPENCL
00024 #include <boost/compute.hpp>
00025 #endif
00026 #include <boost/optional.hpp>
00027
00028 #include "triSYCL/detail/default_classes.hpp"
00029
00030 #include "triSYCL/accessor.hpp"
00031 #include "triSYCL/command_group/detail/task.hpp"
00032 #include "triSYCL/detail/shared_ptr_implementation.hpp"
```

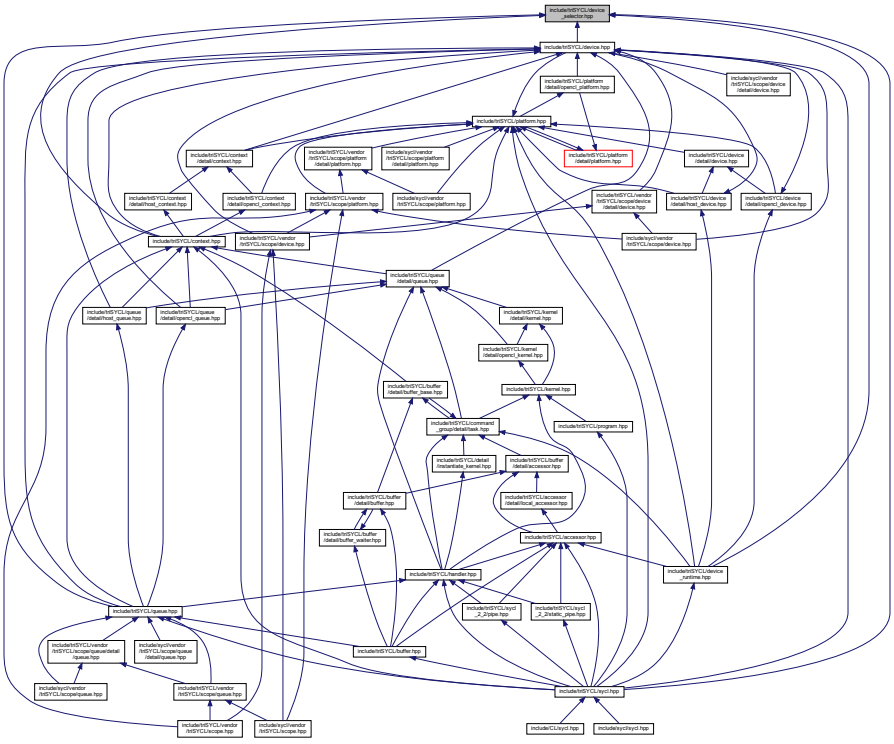
```
00033 #include "triSYCL/device/detail/host_device.hpp"
00034 #ifdef TRISYCL_OPENCL
00035 #include "triSYCL/device/detail/opencl_device.hpp"
00036 #endif
00037 #include "triSYCL/info/device.hpp"
00038 #include "triSYCL/device_selector.hpp"
00039 #include "triSYCL/platform.hpp"
00040
00041 /// The device-side runtime
00042 namespace trisycl::drt {
00043 /** \addtogroup device_runtime Device-side runtime implementation
00044 @{
00045 */
00046
00047 /// SYCL accessor seen from a device perspective
00048 template <typename Accessor>
00049 class accessor :
00050     public detail::container_element_aspect<typename Accessor::value_type> {
00051
00052 /** The pointer to the data
00053
00054 For now none of the following works yet for various reasons:
00055 \code
00056 trisycl::global_ptr<typename Accessor::value_type> buffer;
00057 trisycl::global<typename Accessor::value_type *> buffer;
00058 __global typename Accessor::pointer buffer;
00059 \endcode
00060
00061 So use a more trivial approach
00062 */
00063   TRISYCL_GLOBAL_AS typename Accessor::value_type *buffer;
00064
00065 /** The size of the accessor
00066
00067 The compiler should optimize it out if it is unused */
00068   std::size_t size;
00069
00070 public:
00071
00072 /// Capture the minimal accessor features
00073   accessor(Accessor &a) :
00074     /* Initialize the buffer with actually a \c cl_mem since it will be
00075 overridden with the kernel argument by the serializer at
00076 run-time, but at least from the kernel point of view it has to
00077 look like an \c Accessor::value_type * for the kernel
00078 outlining */
00079     buffer {
00080       reinterpret_cast<TRISYCL_GLOBAL_AS typename Accessor::value_type *>
00081         (a.implementation->get_order())
00082         },
00083     size { a.get_count() }
00084   {
00085 #ifndef TRISYCL_DEVICE
00086     /* Register the buffer address to be updated with the final
00087 version before the kernel set arg */
00088     TRISYCL_DUMP_T("drt::accessor(Accessor &a) :  &a = "
00089                    « (void *) &a « '\n'
00090                    « "\t&buffer = " « (void *) &buffer
00091                    « ", Order = " « a.implementation->get_order());
00092 #endif
00093   }
00094
00095
00096 /// use the accessor with integers à la []
00097   auto &operator[](std::size_t index) const noexcept {
00098     return buffer[index];
00099   }
00100
00101 };
00102
00103
00104 /** Set an argument value of the kernel
00105
00106 \param[in] task is the implementation detail of a triSYCL kernel
00107
00108 \param[in] index is the order of the argument to set (0 is the first one)
00109
00110 \param[in] arg points to the argument value
00111
00112 \param[in] arg_size is the size of the argument
00113 */
00114 TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX
00115 serialize_arg(detail::task &task,
00116               std::size_t index,
00117               void *arg,
00118               std::size_t arg_size) {
00119   TRISYCL_DUMP_T("serialize_arg index = " « index
```

```
00120                 « ", size = " « arg_size « ", arg = " « arg);
00121   task.set_arg(index, arg_size, arg);
00122 }
00123
00124
00125 /** Set an accessor argument of the kernel
00126
00127 \param[in] task is the implementation detail of a triSYCL kernel
00128
00129 \param[in] index is the order of the argument to set (0 is the first one)
00130
00131 \param[in] arg points to the argument value
00132
00133 \param[in] arg_size is the size of the argument
00134 */
00135 TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX
00136 serialize_accessor_arg(detail::task &task,
00137                        std::size_t index,
00138                        void *arg,
00139                        std::size_t arg_size) {
00140   TRISYCL_DUMP_T("serialize_accessor_arg index = " « index
00141                  « ", size = " « arg_size « ", arg = " « arg);
00142 #ifdef TRISYCL_OPENCL
00143   /* The address of the argument has been actually initialized by the
00144 accessor constructor to the accessor order to relate an argument
00145 to its accessor */
00146   task.set_arg(index,
00147                task.get_compute_buffer(reinterpret_cast<std::size_t>(arg)));
00148 #endif
00149 }
00150
00151
00152 /// The binary code of the kernels
00153 namespace code {
00154
00155 /// A binary program containing some kernels
00156   struct program {
00157 /// The size of a binary program
00158     std::size_t binary_size;
00159 /// The bytes of program.  Use this type for \c boost::compute
00160     unsigned const char *binary;
00161
00162 /** The current program
00163
00164 \todo Deal with several programs
00165 */
00166     static TRISYCL_WEAK_ATTRIB_PREFIX boost::optional<program> TRISYCL_WEAK_ATTRIB_SUFFIX
00167     p;
00168
00169     program(std::size_t binary_size, const char *binary)
00170       :  binary_size { binary_size }
00171       , binary { reinterpret_cast<unsigned const char *>(binary) } {
00172         p = *this;
00173         TRISYCL_DUMP_T("Create program with binary size = 0x" « binary_size);
00174       }
00175   };
00176
00177   TRISYCL_WEAK_ATTRIB_PREFIX boost::optional<program>
00178   TRISYCL_WEAK_ATTRIB_SUFFIX program::p;
00179
00180 }
00181
00182 /** The code of the kernels arranged by name
00183
00184 \todo use std::byte when moving to C++17
00185 */
00186 TRISYCL_WEAK_ATTRIB_PREFIX std::map<std::string, std::vector<unsigned char»
00187 TRISYCL_WEAK_ATTRIB_SUFFIX kernel_IR;
00188
00189
00190 /** Set the kernel for a later invocation
00191
00192 \param[in] task is the implementation detail of a triSYCL kernel
00193
00194 \param[in] kernel_name is set by the device compiler to identify
00195 the kernel to call
00196 */
00197 TRISYCL_WEAK_ATTRIB_PREFIX void TRISYCL_WEAK_ATTRIB_SUFFIX
00198 set_kernel(detail::task &task,
00199            const char *kernel_name,
00200            const char *kernel_short_name) {
00201   TRISYCL_DUMP_T("set_kernel setting up " « kernel_name
00202                  « "\n\taka " « kernel_short_name);
00203   // \todo Add kernel caching per device
00204   // auto binary = kernel_IR.find(kernel_name);
00205   // if (binary == kernel_IR.end()) {
00206 #ifdef TRISYCL_OPENCL
```

```
00207   auto context = task.get_queue()->get_boost_compute().get_context();
00208   // Construct an OpenCL program from the precompiled kernel file
00209   auto program = boost::compute::program::create_with_binary
00210     (code::program::p->binary,
00211      code::program::p->binary_size,
00212      context);
00213   TRISYCL_DUMP_T("...on device with name "
00214                  << task.get_queue()->get_boost_compute().get_device().name());
00215   // Build the OpenCL program
00216   program.build();
00217
00218   // Build a SYCL kernel from the OpenCL kernel
00219   trisycl::kernel k { boost::compute::kernel { program, kernel_short_name } };
00220
00221   task.set_kernel(k.implementation);
00222 #endif
00223 }
00224
00225 /// @} to end the Doxygen group
00226
00227 }
00228
00229 /*
00230 # Some Emacs stuff:
00231 ### Local Variables:
00232 ### ispell-local-dictionary:  "american"
00233 ### eval:  (flyspell-prog-mode)
00234 ### End:
00235 */
00236
00237 #endif // TRISYCL_SYCL_DEVICE_RUNTIME_HPP
```

13.101 include/triSYCL/device_selector.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::device_selector

  The SYCL heuristics to select a device. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

13.102 device_selector.hpp

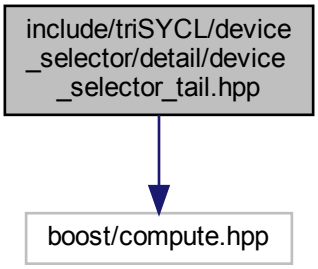
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DEVICE_SELECTOR_HPP
00002 #define TRISYCL_SYCL_DEVICE_SELECTOR_HPP
00003
00004 /** \file The OpenCL SYCL device_selector
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 namespace trisycl {
00013
00014 /** \addtogroup execution Platforms, contexts, devices and queues
00015 @{
00016 */
00017
00018 /** The SYCL heuristics to select a device
00019
00020 The device with the highest score is selected
00021 */
00022 class device_selector {
00023
00024   device select_device(vector_class<platform> platforms) const;
00025 public:
00026
00027 /** Returns a selected device using the functor operator defined in
00028 sub-classes operator()(const device &dev)
00029
00030 \todo Remove this from specification
00031 */
00032   device select_device() const;
00033
00034 /**  This pure virtual operator allows the customization of device
00035 selection.
00036
00037 It defines the behavior of the device_selector functor called by
00038 the SYCL runtime on device selection.  It returns a "score" for each
00039 device in the system and the highest rated device will be used
00040 by the SYCL runtime.
00041 */
00042   virtual int operator()(const device &dev) const = 0;
00043
00044
00045 /// Virtual destructor so the final destructor can be called if any
00046   virtual ~device_selector() {}
00047
00048 };
00049
00050 /// @} to end the execution Doxygen group
00051
00052 }
00053
00054 /*
00055 # Some Emacs stuff:
00056 ### Local Variables:
00057 ### ispell-local-dictionary:  "american"
00058 ### eval:  (flyspell-prog-mode)
00059 ### End:
00060 */
00061
00062 #endif // TRISYCL_SYCL_DEVICE_SELECTOR_HPP
```

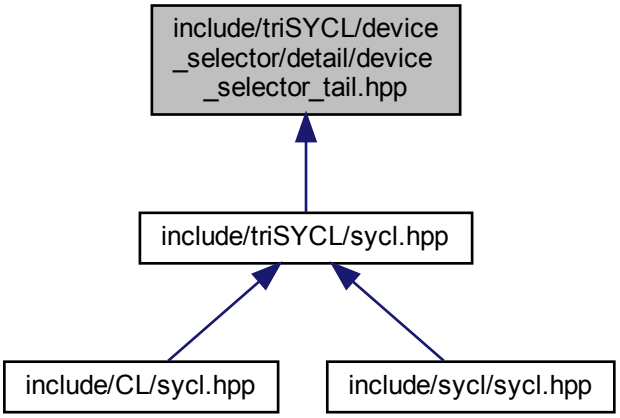
13.103 include/triSYCL/device_selector/detail/device_selector_tail.hpp File Reference

```
#include <boost/compute.hpp>
```

Include dependency graph for device_selector_tail.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::device_type_selector

  A device selector by device_type. More...
- class trisycl::device_typename_selector< DeviceType >

  Select a device by template device_type parameter. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

Typedefs

- using trisycl::default_selector = device_typename_selector< info::device_type::automatic >

  Devices selected by heuristics of the system.
- using trisycl::gpu_selector = device_typename_selector< info::device_type::gpu >

  Select devices according to device type info::device::device_type::gpu from all the available OpenCL devices.
- using trisycl::cpu_selector = device_typename_selector< info::device_type::cpu >

  Select devices according to device type info::device::device_type::cpu from all the available devices and heuristics.
- using trisycl::host_selector = device_typename_selector< info::device_type::host >

  Selects the SYCL host CPU device that does not require an OpenCL runtime.

13.104 device_selector_tail.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DEVICE_SELECTOR_DETAIL_DEVICE_SELECTOR_TAIL_HPP
00002 #define TRISYCL_SYCL_DEVICE_SELECTOR_DETAIL_DEVICE_SELECTOR_TAIL_HPP
00003
00004 /** \file The ending part of of the OpenCL SYCL device_selector
00005
00006 This is here to break a dependence between device and device_selector
00007
00008 \todo Implement lacking SYCL 2.2 selectors
00009
00010 Ronan at Keryell point FR
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #ifdef TRISYCL_OPENCL
00017 #include <boost/compute.hpp>
00018 #endif
00019
00020 namespace trisycl {
00021
00022 /** \addtogroup execution Platforms, contexts, devices and queues
00023 @{
00024 */
00025 inline device device_selector::select_device(vector_class<platform> platforms)const {
00026   int best_id = -1;
00027   int best_score = -1;
00028   device best_device;
00029
00030   for (auto &platform :  platforms) {
00031     for (auto &dev :  platform.get_devices()) {
00032       int score = operator()(dev);
00033       if (score > best_score) {
00034         best_id = 1;
00035         best_device = dev;
00036         best_score = score;
00037       }
00038     }
00039   }
00040   if (best_id == -1)
00041     throw ::trisycl::runtime_error("Could not find device");
00042
00043   return best_device;
00044 }
00045
00046 inline device device_selector::select_device()const  {
00047   return select_device(platform::get_platforms());
00048 }
00049
00050 /** A device selector by device_type
00051
```

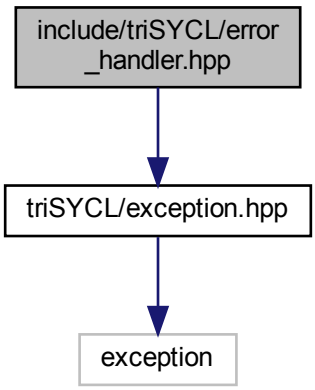
```
00052 \todo To be added to the specification
00053 */
00054 class device_type_selector :  public device_selector {
00055
00056 private:
00057
00058 /// The device_type to select
00059   info::device_type device_type;
00060
00061 /** Cache the default device to select with the default device
00062 selector.
00063
00064 This is the host device at construction time and remains as is
00065 if there is no openCL device */
00066   device default_device;
00067
00068 public:
00069
00070   device_type_selector(info::device_type device_type)
00071     :  device_type { device_type } {
00072     // The default device selection heuristic
00073 #ifdef TRISYCL_OPENCL
00074     if (device_type == info::device_type::automatic) {
00075       // Ask Boost.Compute for the default OpenCL device
00076       try {
00077         default_device = boost::compute::system::default_device();
00078       }
00079       catch (...)  {
00080         /* If there is no OpenCL device, just keep the
00081 default-constructed device, which is the host device */
00082       }
00083     }
00084 #endif
00085   }
00086
00087
00088   // To select only the requested device_type
00089   int operator()(const device &dev)const override {
00090     if (device_type == info::device_type::all)
00091       // All devices fit all
00092       return 1;
00093
00094     if (device_type == info::device_type::automatic)
00095       // Only select the default device
00096       return dev == default_device ?  1 :  -1;
00097
00098     if (device_type == info::device_type::opencl)
00099       // For now, any non host device is an OpenCL device
00100       return dev.is_host() ?  -1 :  1;
00101
00102     return dev.type() == device_type ?  1 :  -1;
00103   }
00104
00105 };
00106
00107
00108 /** Select a device by template device_type parameter
00109
00110 \todo To be added to the specification
00111 */
00112 template <info::device_type DeviceType>
00113 class device_typename_selector :  public device_type_selector {
00114
00115 public:
00116
00117   device_typename_selector() :  device_type_selector { DeviceType } {}
00118
00119 };
00120
00121
00122 /** Devices selected by heuristics of the system
00123
00124 If no OpenCL device is found then it defaults to the SYCL host device.
00125
00126 To influence the default device selection, use the Boost.Compute
00127 environment variables:
00128
00129 - \c BOOST_COMPUTE_DEFAULT_DEVICE
00130
00131 - \c BOOST_COMPUTE_DEFAULT_DEVICE_TYPE
00132
00133 - \c BOOST_COMPUTE_DEFAULT_PLATFORM
00134
00135 - \c BOOST_COMPUTE_DEFAULT_VENDOR
00136 */
00137 using default_selector = device_typename_selector<info::device_type::automatic>;
00138
```

```
00139
00140 /** Select devices according to device type info::device::device_type::gpu
00141 from all the available OpenCL devices.
00142
00143 If no OpenCL GPU device is found the selector fails.
00144
00145 Select the best GPU, if any.
00146 */
00147 using gpu_selector = device_typename_selector<info::device_type::gpu>;
00148
00149
00150 /** Select devices according to device type info::device::device_type::cpu
00151 from all the available devices and heuristics
00152
00153 If no OpenCL CPU device is found the selector fails.
00154 */
00155 using cpu_selector = device_typename_selector<info::device_type::cpu>;
00156
00157
00158 /** Selects the SYCL host CPU device that does not require an OpenCL
00159 runtime
00160 */
00161 using host_selector = device_typename_selector<info::device_type::host>;
00162
00163 /// @} to end the execution Doxygen group
00164
00165 }
00166
00167 /*
00168 # Some Emacs stuff:
00169 ### Local Variables:
00170 ### ispell-local-dictionary:  "american"
00171 ### eval:  (flyspell-prog-mode)
00172 ### End:
00173 */
00174
00175 #endif // TRISYCL_SYCL_DEVICE_SELECTOR_DETAIL_DEVICE_SELECTOR_TAIL_HPP
```

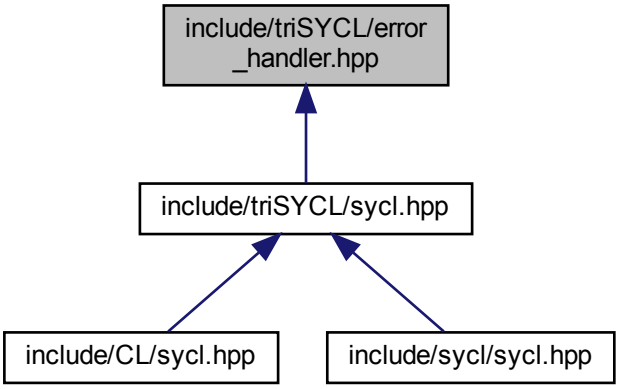
13.105 include/triSYCL/error_handler.hpp File Reference

`#include "triSYCL/exception.hpp"`

Include dependency graph for error_handler.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::error_handler

  User supplied error handler to call a user-provided function when an error happens from a SYCL object that was constructed with this error handler. More...
- struct trisycl::trisycl::default_error_handler

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::trisycl

13.106 error_handler.hpp

Go to the documentation of this file.

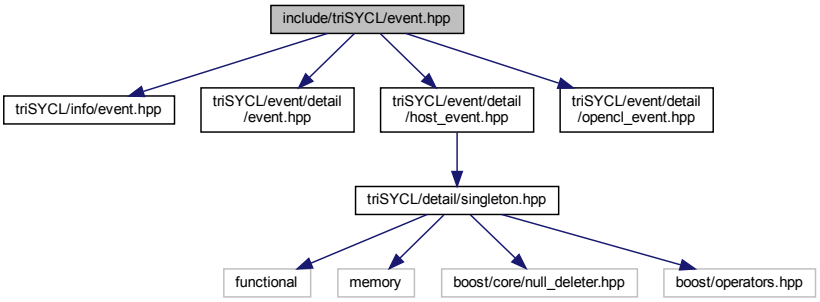
```
00001 #ifndef TRISYCL_SYCL_ERROR_HANDLER_HPP
00002 #define TRISYCL_SYCL_ERROR_HANDLER_HPP
00003
00004 /** \file The OpenCL SYCL error_handler
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include "triSYCL/exception.hpp"
00013
00014 namespace trisycl {
00015
00016 /** \addtogroup error_handling Error handling
00017 @{
00018 */
00019
00020 /// \todo Refactor when updating to latest specification
00021 namespace trisycl {
```

```
00022   // Create a default error handler to be used when nothing is specified
00023   struct default_error_handler;
00024 }
00025
00026
00027 /** User supplied error handler to call a user-provided function when an
00028 error happens from a SYCL object that was constructed with this error
00029 handler
00030 */
00031 struct error_handler {
00032 /** The method to define to be called in the case of an error
00033
00034 \todo Add "virtual void" to the specification
00035 */
00036   virtual void report_error(exception &error) = 0;
00037
00038 /** Add a default_handler to be used by default
00039
00040 \todo add this concept to the specification?
00041 */
00042   static trisycl::default_error_handler default_handler;
00043
00044   // Virtual to call the real destructor
00045   virtual ~error_handler() = 0;
00046 };
00047
00048
00049 namespace trisycl {
00050
00051   struct default_error_handler :  error_handler {
00052
00053     void report_error(exception &)override {
00054     }
00055   };
00056 }
00057
00058   // \todo finish initialization
00059   //error_handler::default_handler = nullptr;
00060
00061
00062 /// @} End the error_handling Doxygen group
00063
00064 }
00065
00066 /*
00067 # Some Emacs stuff:
00068 ### Local Variables:
00069 ### ispell-local-dictionary:  "american"
00070 ### eval:  (flyspell-prog-mode)
00071 ### End:
00072 */
00073
00074 #endif // TRISYCL_SYCL_ERROR_HANDLER_HPP
```

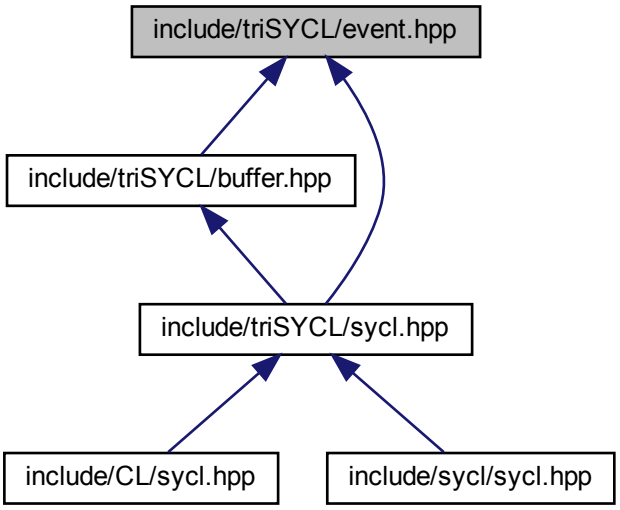
13.107 include/triSYCL/event.hpp File Reference

```
#include "triSYCL/info/event.hpp"
#include "triSYCL/event/detail/event.hpp"
#include "triSYCL/event/detail/host_event.hpp"
#include "triSYCL/event/detail/opencl_event.hpp"
```

Include dependency graph for event.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::event

  SYCL event.
- struct std::hash< trisycl::event >

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace std

Functions

- template<> auto trisycl::event::get_info< info::event::command_execution_status > () const
- template<> auto trisycl::event::get_info< info::event::reference_count > () const
- template<> auto trisycl::event::get_profiling_info< info::event_profiling::command_submit > () const
- template<> auto trisycl::event::get_profiling_info< info::event_profiling::command_start > () const
- template<> auto trisycl::event::get_profiling_info< info::event_profiling::command_end > () const

13.108 event.hpp

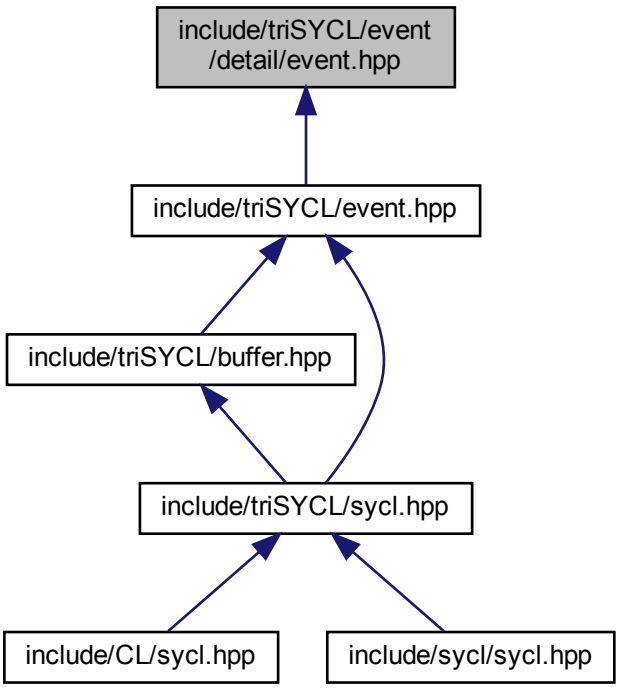
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_EVENT_HPP
00002 #define TRISYCL_SYCL_EVENT_HPP
00003
00004 /** \file The event class
00005
00006 Ronan at keryell dot FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011 #include "triSYCL/info/event.hpp"
00012 #include "triSYCL/event/detail/event.hpp"
00013 #include "triSYCL/event/detail/host_event.hpp"
00014 #ifdef TRISYCL_OPENCL
00015 #include "triSYCL/event/detail/opencl_event.hpp"
00016 #endif
00017
00018 namespace trisycl {
00019
00020 /// SYCL event.
00021 class event :  public detail::shared_ptr_implementation<event, detail::event> {
00022
00023   // The type encapsulating the implementation
00024   using implementation_t = typename event::shared_ptr_implementation;
00025
00026   friend implementation_t;
00027 public:
00028
00029   event() :  implementation_t { detail::host_event::instance() } {}
00030
00031 #ifdef TRISYCL_OPENCL
00032 /** Construct an event class using the clEvent from OpenCL.
00033
00034 This is a triSYCL extension for boost::compute interoperation.
00035 */
00036   event(cl_event clEvent, const context& syclContext)
00037     :  event { boost::compute::event { clEvent } } {}
00038
00039 /** Construct an event class using the boost::compute::event.
00040
00041 This is a triSYCL extension for boost::compute interoperation.
00042 */
00043   event(const boost::compute::event &e)
00044     :  implementation_t { detail::opencl_event::instance(e) } {}
00045 #endif
00046
00047 /// Return true if the event is a SYCL host event
00048   bool is_host()const { return implementation->is_host(); }
00049
00050 #ifdef TRISYCL_OPENCL
00051 /// Return the cl_event from the underlying OpenCL event.
00052   cl_event get() {
00053     return implementation->get();
00054   }
00055
00056 /** Return the underlying boost compute event
00057
00058 This is a triSYCL extension
00059 */
00060   boost::compute::event get_boost_compute()const {
00061     return implementation->get_boost_compute();
00062   }
00063 #endif
00064
00065   vector_class<event> get_wait_list() {
00066     TRISYCL_UNIMPL;
00067     return {};
```

```
00068   }
00069
00070 /** Wait for the event and the command associated with it to complete.
00071
00072 */
00073   void wait() {
00074     implementation->wait();
00075   }
00076
00077   static void wait(const vector_class<event> &eventList) {
00078     TRISYCL_UNIMPL;
00079   }
00080
00081   void wait_and_throw() {
00082     TRISYCL_UNIMPL;
00083   }
00084
00085   static void wait_and_throw(const vector_class<event> &eventList) {
00086     TRISYCL_UNIMPL;
00087   }
00088
00089 /// Query the event for information
00090   template <info::event param>
00091   inline auto get_info() const;
00092
00093 /// Query the event for event profiling
00094   template <info::event_profiling param>
00095   inline auto get_profiling_info() const;
00096
00097 };
00098
00099 template<>
00100 inline auto event::get_info<info::event::command_execution_status>()const {
00101   return implementation->get_command_execution_status();
00102 }
00103
00104 template<>
00105 inline auto event::get_info<info::event::reference_count>()const {
00106   return implementation->get_reference_count();
00107 }
00108
00109 template<>
00110 inline auto event::get_profiling_info<info::event_profiling::command_submit>()const {
00111   return implementation->get_profiling_info(info::event_profiling::command_submit);
00112 }
00113
00114 template<>
00115 inline auto event::get_profiling_info<info::event_profiling::command_start>()const {
00116   return implementation->get_profiling_info(info::event_profiling::command_start);
00117 }
00118
00119 template<>
00120 inline auto event::get_profiling_info<info::event_profiling::command_end>()const {
00121   return implementation->get_profiling_info(info::event_profiling::command_end);
00122 }
00123
00124 }
00125
00126
00127 namespace std {
00128
00129 template <> struct hash<trisycl::event> {
00130
00131   auto operator()(const trisycl::event &e)const {
00132     // Forward the hashing to the implementation
00133     return e.hash();
00134   }
00135
00136 };
00137
00138 }
00139
00140 /*
00141 # Some Emacs stuff:
00142 ### Local Variables:
00143 ### ispell-local-dictionary:  "american"
00144 ### eval:  (flyspell-prog-mode)
00145 ### End:
00146 */
00147
00148 #endif // TRISYCL_SYCL_EVENT_HPP
```

13.109 include/triSYCL/event/detail/event.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::detail::event

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.110 event.hpp

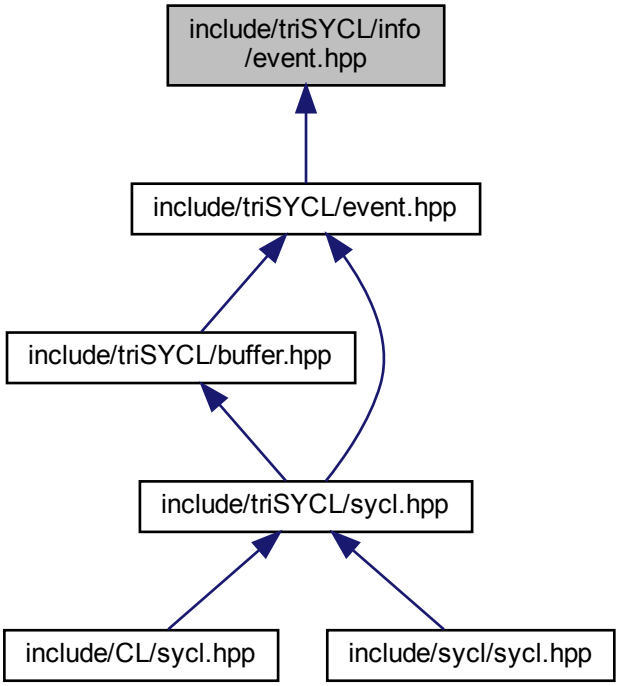
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_EVENT_DETAIL_EVENT_HPP
00002 #define TRISYCL_SYCL_EVENT_DETAIL_EVENT_HPP
00003
00004 /** \file The OpenCL SYCL abstract event
00005
00006 Dave Airlie
```

```
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 namespace trisycl::detail {
00013
00014 struct event :  detail::debug<detail::event> {
00015
00016 public:
00017 #ifdef TRISYCL_OPENCL
00018 /// Return the cl_event of the underlying OpenCL event
00019   virtual cl_event get() const = 0;
00020
00021 /// Return the underlying Boost.Compute event, if any
00022   virtual const boost::compute::event &get_boost_compute() const = 0;
00023 #endif
00024
00025   virtual bool is_host() const = 0;
00026
00027   virtual info::event_command_status get_command_execution_status() const = 0;
00028
00029   virtual cl_ulong get_profiling_info(info::event_profiling param) const = 0;
00030
00031   virtual cl_uint get_reference_count() const = 0;
00032
00033   virtual void wait() const = 0;
00034
00035   virtual ~event() {}
00036 };
00037
00038 }
00039
00040 #endif
```

13.111 include/triSYCL/info/event.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::info

Enumerations

- enum class trisycl::info::event : unsigned int { trisycl::info::command_execution_status , trisycl::info::reference_count }
- enum class trisycl::info::event_command_status : unsigned int { trisycl::info::TRISYCL_OPENCL_ONLY =(= CL_SUBMITTED) , trisycl::info::TRISYCL_OPENCL_ONLY =(= CL_RUNNING) , trisycl::info::TRISYCL_OPENCL_ONLY =(= CL_COMPLETE) }
- enum class trisycl::info::event_profiling : unsigned int { trisycl::info::TRISYCL_OPENCL_ONLY =(= CL↩_PROFILING_COMMAND_SUBMIT) , trisycl::info::TRISYCL_OPENCL_ONLY =(= CL_PROFILING_↩COMMAND_START) , trisycl::info::TRISYCL_OPENCL_ONLY =(= CL_PROFILING_COMMAND_END) }

Functions

- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::event_profiling::command_submit, trisycl::cl_ulong)
- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::event_profiling::command_start, trisycl::cl_ulong)
- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::event_profiling::command_end, trisycl::cl_ulong)

13.112 event.hpp

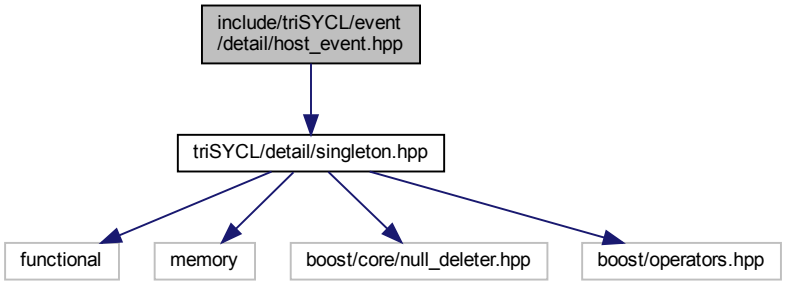
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_INFO_EVENT_HPP
00002 #define TRISYCL_SYCL_INFO_EVENT_HPP
00003
00004 namespace trisycl::info {
00005
00006 enum class event :  unsigned int {
00007   command_execution_status,
00008   reference_count,
00009 };
00010
00011 enum class event_command_status :  unsigned int {
00012    submitted TRISYCL_OPENCL_ONLY(= CL_SUBMITTED),
00013    running TRISYCL_OPENCL_ONLY(= CL_RUNNING),
00014    complete TRISYCL_OPENCL_ONLY(= CL_COMPLETE)
00015 };
00016
00017 enum class event_profiling :  unsigned int {
00018   command_submit TRISYCL_OPENCL_ONLY(= CL_PROFILING_COMMAND_SUBMIT),
00019   command_start TRISYCL_OPENCL_ONLY(= CL_PROFILING_COMMAND_START),
00020   command_end TRISYCL_OPENCL_ONLY(= CL_PROFILING_COMMAND_END),
00021 };
00022
00023 TRISYCL_INFO_PARAM_TRAITS(info::event::command_execution_status, event_command_status)
00024 TRISYCL_INFO_PARAM_TRAITS(info::event::reference_count, trisycl::cl_uint)
00025
00026 TRISYCL_INFO_PARAM_TRAITS(info::event_profiling::command_submit, trisycl::cl_ulong);
00027 TRISYCL_INFO_PARAM_TRAITS(info::event_profiling::command_start, trisycl::cl_ulong);
00028 TRISYCL_INFO_PARAM_TRAITS(info::event_profiling::command_end, trisycl::cl_ulong);
00029
00030 }
00031 #endif /* TRISYCL_SYCL_INFO_EVENT_HPP */
```

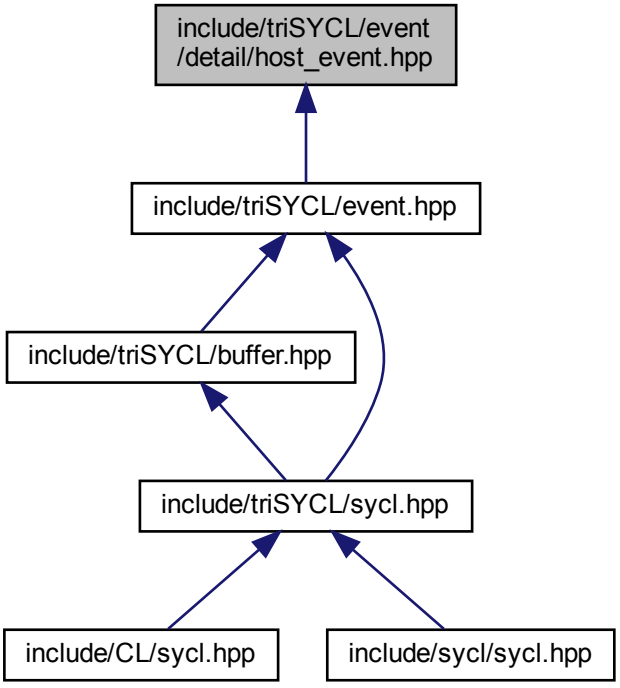
13.113 include/triSYCL/event/detail/host_event.hpp File Reference

```
#include "triSYCL/detail/singleton.hpp"
```

Include dependency graph for host_event.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::host_event

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

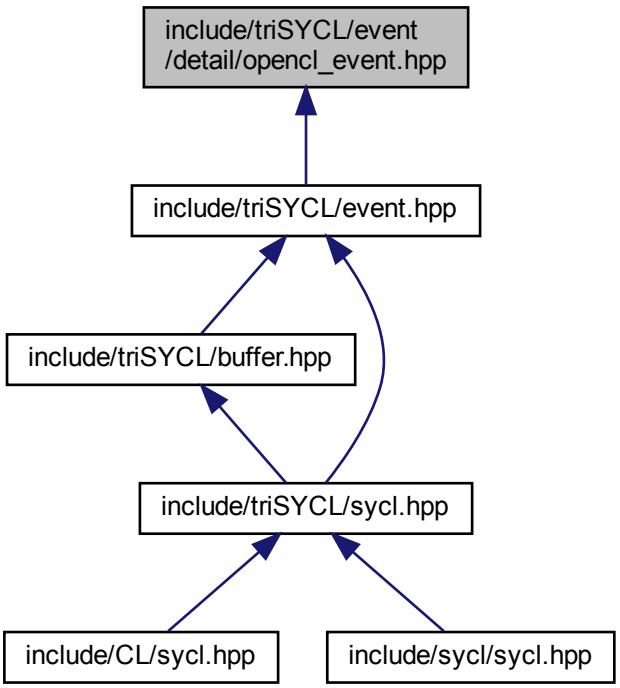
13.114 host_event.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_EVENT_DETAIL_HOST_EVENT_HPP
00002 #define TRISYCL_SYCL_EVENT_DETAIL_HOST_EVENT_HPP
00003
00004 /** \file The OpenCL triSYCL host event implementation
00005
00006 Dave Airlie
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include "triSYCL/detail/singleton.hpp"
00013
00014 namespace trisycl::detail {
00015
00016 class host_event :  public detail::event,
00017                     public detail::singleton<host_event> {
00018
00019 public:
00020 #ifdef TRISYCL_OPENCL
00021   cl_event get()const override {
00022     throw non_cl_error("The host event has no OpenCL event");
00023   }
00024
00025   boost::compute::event &get_boost_compute()const override {
00026     throw
00027       non_cl_error("The host device has no underlying Boost Compute event");
00028   }
00029 #endif
00030
00031   bool is_host()const override {
00032     return true;
00033   }
00034
00035   cl_uint get_reference_count()const override {
00036     return 0;
00037   }
00038
00039   info::event_command_status get_command_execution_status()const override {
00040     return info::event_command_status::submitted;
00041   }
00042
00043   cl_ulong get_profiling_info(info::event_profiling param)const override {
00044     return 0;
00045   }
00046
00047   void wait()const override {
00048   }
00049 };
00050
00051 }
00052
00053 #endif // TRISYCL_SYCL_EVENT_DETAIL_HOST_EVENT_HPP
```

13.115 include/triSYCL/event/detail/opencl_event.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::opencl_event

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.116 opencl_event.hpp

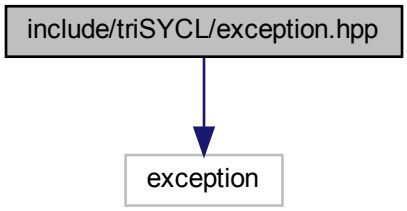
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_EVENT_DETAIL_OPENCL_EVENT_HPP
00002 #define TRISYCL_SYCL_EVENT_DETAIL_OPENCL_EVENT_HPP
00003
00004 /** \file The triSYCL OpenCL event implementation
00005
00006 Dave Airlie
```

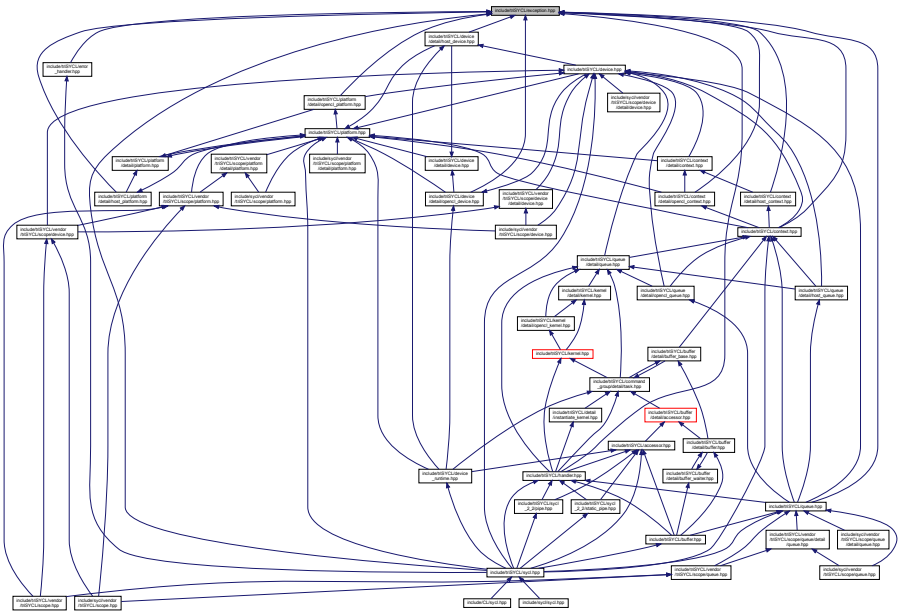
```
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012
00013 namespace trisycl::detail {
00014
00015 class opencl_event :  public detail::event {
00016   boost::compute::event e;
00017
00018 /** A cache to always return the same alive event for a given OpenCL
00019 event
00020
00021 C++11 guaranties the static construction is thread-safe
00022 */
00023   static detail::cache<cl_event, detail::opencl_event> cache;
00024 public:
00025   cl_event get()const override {
00026     return e.get();
00027   }
00028
00029   const boost::compute::event &get_boost_compute()const override {
00030     return e;
00031   }
00032
00033   bool is_host()const override {
00034     return false;
00035   }
00036
00037   info::event_command_status get_command_execution_status()const override {
00038     return static_cast<info::event_command_status>(e.status());
00039   }
00040
00041   cl_ulong get_profiling_info(info::event_profiling param)const override {
00042     return e.get_profiling_info<cl_ulong>(static_cast<cl_profiling_info>(param));
00043   }
00044
00045   trisycl::cl_uint get_reference_count()const override {
00046     return e.get_info<CL_EVENT_REFERENCE_COUNT>();
00047   }
00048
00049   void wait()const override {
00050     e.wait();
00051   }
00052
00053 /// Get a singleton instance of the \c opencl_event
00054   static std::shared_ptr<opencl_event>
00055   instance(const boost::compute::event &e) {
00056     return cache.get_or_register(e.get(),
00057                                  [&] { return new opencl_event { e }; });
00058   }
00059 private:
00060 /// Only the instance factory can build it
00061   opencl_event(const boost::compute::event &e) :
00062     e { e } {}
00063
00064 public:
00065   ~opencl_event()override {
00066     cache.remove(e.get());
00067   }
00068 };
00069
00070 }
00071
00072 #endif // TRISYCL_SYCL_EVENT_DETAIL_OPENCL_EVENT_HPP
```

13.117 include/triSYCL/exception.hpp File Reference

`#include <exception>`
Include dependency graph for exception.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::exception_list

  Exception list to store several exceptions. More...
- class trisycl::exception

  Encapsulate a SYCL error information. More...
- class trisycl::cl_exception

  Returns the OpenCL error code encapsulated in the exception. More...
- struct trisycl::async_exception

  An error stored in an exception_list for asynchronous errors. More...

- class trisycl::runtime_error
- class trisycl::kernel_error

  Error that occurred before or while enqueuing the SYCL kernel. More...
- class trisycl::accessor_error

  Error regarding the trisycl::accessor objects defined. More...
- class trisycl::nd_range_error

  Error regarding the trisycl::nd_range specified for the SYCL kernel. More...
- class trisycl::event_error

  Error regarding associated trisycl::event objects. More...
- class trisycl::invalid_parameter_error

  Error regarding parameters to the SYCL kernel, it may apply to any captured parameters to the kernel lambda. More...
- class trisycl::device_error

  The SYCL device will trigger this exception on error. More...
- class trisycl::compile_program_error

  Error while compiling the SYCL kernel to a SYCL device. More...
- class trisycl::link_program_error

  Error while linking the SYCL kernel to a SYCL device. More...
- class trisycl::invalid_object_error

  Error regarding any memory objects being used inside the kernel. More...
- class trisycl::memory_allocation_error

  Error on memory allocation on the SYCL device for a SYCL kernel. More...
- class trisycl::pipe_error

  A failing pipe error will trigger this exception on error. More...
- class trisycl::platform_error

  The SYCL platform will trigger this exception on error. More...
- class trisycl::profiling_error

  The SYCL runtime will trigger this error if there is an error when profiling info is enabled. More...
- class trisycl::feature_not_supported

  Exception thrown when an optional feature or extension is used in a kernel but its not available on the device the SYCL kernel is being enqueued on. More...
- class trisycl::non_cl_error

  Exception for an OpenCL operation requested in a non OpenCL area. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

Typedefs

- using trisycl::exception_ptr = std::exception_ptr

  A shared pointer to an exception as in C++ specification.
- using trisycl::async_handler = function_class< void, exception_list >

13.118 exception.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_EXCEPTION_HPP
00002 #define TRISYCL_SYCL_EXCEPTION_HPP
00003
00004 /** \file The OpenCL SYCL exception
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <exception>
00013
00014 namespace trisycl {
00015
00016 /** \addtogroup error_handling Error handling
00017 @{
00018 */
00019
00020
00021 /** A shared pointer to an exception as in C++ specification
00022
00023 \todo Do we need this instead of reusing directly the one from C++11?
00024 */
00025 using exception_ptr = std::exception_ptr;
00026
00027
00028 /** Exception list to store several exceptions
00029
00030 \todo Do we need to define it in SYCL or can we rely on plain C++17 one?
00031 */
00032 struct exception_list :  std::vector<exception_ptr> {
00033   using std::vector<exception_ptr>::vector;
00034 };
00035
00036 using async_handler = function_class<void, exception_list>;
00037
00038
00039 /// Encapsulate a SYCL error information
00040 class exception {
00041
00042 /// The error message to return
00043   string_class message;
00044
00045 public:
00046
00047 /// Construct an exception with a message for internal use
00048   exception(const string_class &message) :  message { message } {}
00049
00050 /// Returns a descriptive string for the error, if available
00051   string_class what()const {
00052     return message;
00053   }
00054
00055
00056 /** Returns the context that caused the error
00057
00058 Returns nullptr if not a buffer error.
00059
00060 \todo Cannot return nullptr.  Use optional?  Use a specific exception type?
00061 */
00062   //context get_context()
00063
00064 };
00065
00066
00067 /// Returns the OpenCL error code encapsulated in the exception
00068 class cl_exception :  public exception {
00069
00070 #ifdef TRISYCL_OPENCL
00071 /// The OpenCL error code to return
00072
00073   cl_int cl_code;
00074
00075 public:
00076
00077 /**  Construct an exception with a message and OpenCL error code for
00078 internal use */
00079   cl_exception(const string_class &message, cl_int cl_code)
00080     :  exception { message }, cl_code { cl_code } {}
00081
00082   // thrown as a result of an OpenCL API error code
```

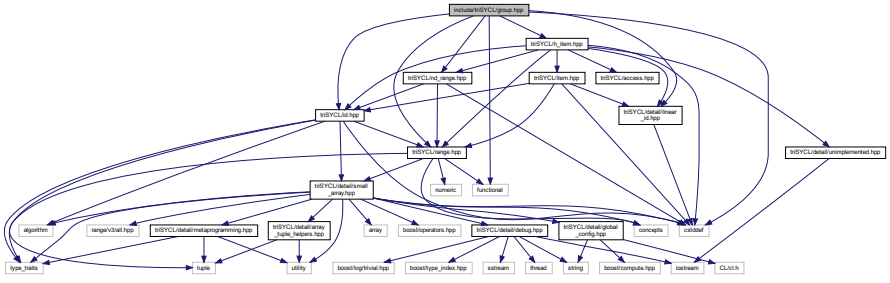
```
00083   cl_int get_cl_code()const {
00084     return cl_code;
00085   }
00086 #endif
00087
00088 };
00089
00090
00091 /// An error stored in an exception_list for asynchronous errors
00092 struct async_exception :  exception {
00093   using exception::exception;
00094 };
00095
00096
00097 class runtime_error :  public exception {
00098   using exception::exception;
00099 };
00100
00101
00102 /// Error that occurred before or while enqueuing the SYCL kernel
00103 class kernel_error :  public runtime_error {
00104   using runtime_error::runtime_error;
00105 };
00106
00107
00108 /// Error regarding the trisycl::accessor objects defined
00109 class accessor_error :  public runtime_error {
00110   using runtime_error::runtime_error;
00111 };
00112
00113
00114 /// Error regarding the trisycl::nd_range specified for the SYCL kernel
00115 class nd_range_error :  public runtime_error {
00116   using runtime_error::runtime_error;
00117 };
00118
00119
00120 /// Error regarding associated trisycl::event objects
00121 class event_error :  public runtime_error {
00122   using runtime_error::runtime_error;
00123 };
00124
00125
00126 /** Error regarding parameters to the SYCL kernel, it may apply to any
00127 captured parameters to the kernel lambda
00128 */
00129 class invalid_parameter_error :  public runtime_error {
00130   using runtime_error::runtime_error;
00131 };
00132
00133
00134 /// The SYCL device will trigger this exception on error
00135 class device_error :  public exception {
00136   using exception::exception;
00137 };
00138
00139
00140 /// Error while compiling the SYCL kernel to a SYCL device
00141 class compile_program_error :  public device_error {
00142   using device_error::device_error;
00143 };
00144
00145
00146 /// Error while linking the SYCL kernel to a SYCL device
00147 class link_program_error :  public device_error {
00148   using device_error::device_error;
00149 };
00150
00151
00152 /// Error regarding any memory objects being used inside the kernel
00153 class invalid_object_error :  public device_error {
00154   using device_error::device_error;
00155 };
00156
00157
00158 /// Error on memory allocation on the SYCL device for a SYCL kernel
00159 class memory_allocation_error :  public device_error {
00160   using device_error::device_error;
00161 };
00162
00163
00164 /// A failing pipe error will trigger this exception on error
00165 class pipe_error :  public runtime_error {
00166   using runtime_error::runtime_error;
00167 };
00168
00169
```

```
00170 /// The SYCL platform will trigger this exception on error
00171 class platform_error :  public device_error {
00172   using device_error::device_error;
00173 };
00174
00175
00176 /** The SYCL runtime will trigger this error if there is an error when
00177 profiling info is enabled
00178 */
00179 class profiling_error :  public device_error {
00180   using device_error::device_error;
00181 };
00182
00183
00184 /** Exception thrown when an optional feature or extension is used in
00185 a kernel but its not available on the device the SYCL kernel is
00186 being enqueued on
00187 */
00188 class feature_not_supported :  public device_error {
00189   using device_error::device_error;
00190 };
00191
00192
00193 /** Exception for an OpenCL operation requested in a non OpenCL area
00194
00195 \todo Add to the specification
00196
00197 \todo Clean implementation
00198
00199 \todo Exceptions are named error in C++
00200 */
00201 class non_cl_error :  public runtime_error {
00202   using runtime_error::runtime_error;
00203 };
00204
00205
00206 /// @} End the error_handling Doxygen group
00207
00208 }
00209
00210 /*
00211 # Some Emacs stuff:
00212 ### Local Variables:
00213 ### ispell-local-dictionary:  "american"
00214 ### eval:  (flyspell-prog-mode)
00215 ### End:
00216 */
00217
00218 #endif // TRISYCL_SYCL_EXCEPTION_HPP
```

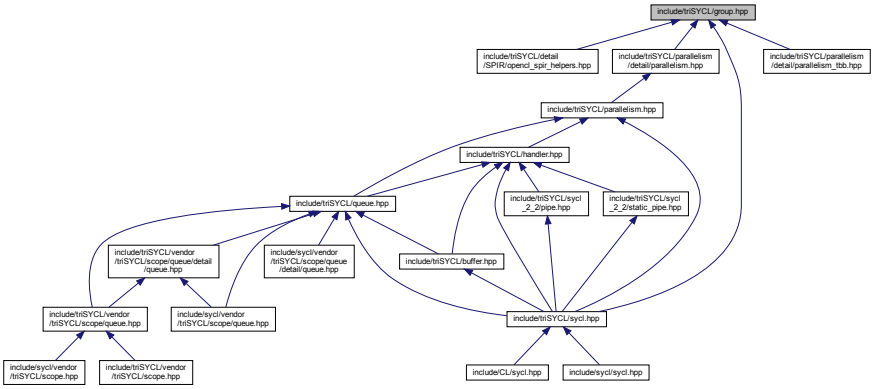
13.119 include/triSYCL/group.hpp File Reference

```
#include <cstddef>
#include <functional>
#include "triSYCL/detail/linear_id.hpp"
#include "triSYCL/h_item.hpp"
#include "triSYCL/id.hpp"
#include "triSYCL/nd_range.hpp"
#include "triSYCL/range.hpp"
```

Include dependency graph for group.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::group< Dimensions >

  A group index used in a parallel_for_workitem to specify a work_group. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Functions

- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for_workitem_in_group (const group< Dimensions > &g, ParallelForFunctor f)

  Implement the loop on the work-items inside a work-group.

13.120 group.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_GROUP_HPP
00002 #define TRISYCL_SYCL_GROUP_HPP
00003
00004 /** \file The OpenCL SYCL group<>
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <cstddef>
00013 #include <functional>
00014
00015 #include "triSYCL/detail/linear_id.hpp"
00016 #include "triSYCL/h_item.hpp"
00017 #include "triSYCL/id.hpp"
00018 #include "triSYCL/nd_range.hpp"
```

```
00019 #include "triSYCL/range.hpp"
00020
00021 namespace trisycl {
00022
00023 template <int Dimensions = 1>
00024 struct group;
00025
00026 namespace detail {
00027
00028 template <int Dimensions = 1, typename ParallelForFunctor>
00029 void parallel_for_workitem_in_group(const group<Dimensions> &g,
00030                                     ParallelForFunctor f);
00031
00032 }
00033
00034 /** \addtogroup parallelism Expressing parallelism through kernels
00035 @{
00036 */
00037
00038 /** A group index used in a parallel_for_workitem to specify a work_group
00039 */
00040 template <int Dimensions>
00041 struct group :  boost::equality_comparable<group<Dimensions>> {
00042 /// The number of dimensions of the group
00043   static auto constexpr rank() { return Dimensions; }
00044
00045 private:
00046
00047 /// The coordinate of the group item
00048   id<Dimensions> group_id;
00049
00050 /// Keep a reference on the nd_range to serve potential query on it
00051   nd_range<Dimensions> ndr;
00052
00053 public:
00054
00055 /** Create a group from an nd_range<> with a 0 id<>
00056
00057 \todo This should be private since it is only used by the triSYCL
00058 implementation
00059 */
00060   group(const nd_range<Dimensions> &ndr) :  ndr { ndr } {}
00061
00062
00063 /** Create a group from an id and a nd_range<>
00064
00065 \todo This should be private somehow, but it is used by the
00066 validation infrastructure
00067 */
00068   group(const id<Dimensions> &i, const nd_range<Dimensions> &ndr ) :
00069     group_id { i }, ndr { ndr } {}
00070
00071
00072 /** To be able to copy and assign group, use default constructors too
00073
00074 \todo Make most of them protected, reserved to implementation
00075 */
00076   group() = default;
00077
00078
00079 /** Return an id representing the index of the group within the nd_range
00080 for every dimension
00081 */
00082   id<Dimensions> get_id()const { return group_id; }
00083
00084
00085 /// Return the index of the group in the given dimension
00086   size_t get_id(int dimension)const { return get_id()[dimension]; }
00087
00088
00089 /** Return the index of the group in the given dimension within the
00090 nd_range<>
00091
00092 \todo In this implementation it is not const because the group<> is
00093 written in the parallel_for iterators.  To fix according to the
00094 specification
00095 */
00096   auto &operator[](int dimension) {
00097     return group_id[dimension];
00098   }
00099
00100
00101 /** Return a range<> representing the dimensions of the current
00102 group
00103
00104 This local range may have been provided by the programmer, or chosen
00105 by the runtime.
```

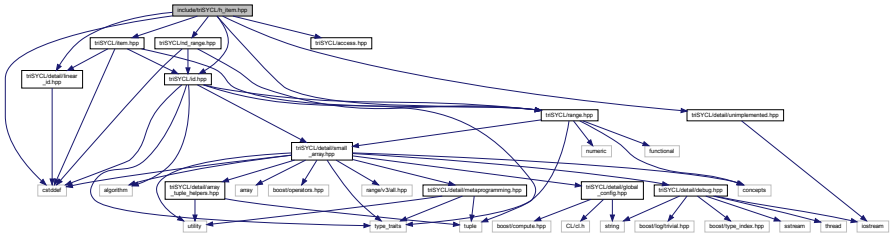
```
00106
00107 \todo Fix this comment and the specification
00108 */
00109   range<Dimensions> get_group_range()const {
00110     return get_nd_range().get_group_range();
00111   }
00112
00113
00114 /// Return element dimension from the con stituent group range
00115   size_t get_group_range(int dimension)const {
00116     return get_group_range()[dimension];
00117   }
00118
00119
00120 /// Get the local range for this work_group
00121   range<Dimensions> get_global_range()const {
00122     return get_nd_range().get_global_range();
00123   }
00124
00125
00126 /// Return element dimension from the constituent global range
00127   size_t get_global_range(int dimension)const {
00128     return get_global_range()[dimension];
00129   }
00130
00131
00132 /** Get the local range for this work_group
00133
00134 \todo Add to the specification
00135 */
00136   range<Dimensions> get_local_range()const {
00137     return get_nd_range().get_local_range();
00138   }
00139
00140
00141 /** Return element dimension from the constituent local range
00142
00143 \todo Add to the specification
00144 */
00145   size_t get_local_range(int dimension)const {
00146     return get_local_range()[dimension];
00147   }
00148
00149
00150 /**  Get the offset of the NDRange
00151
00152 \todo Add to the specification
00153 */
00154   id<Dimensions> get_offset()const { return get_nd_range().get_offset(); }
00155
00156
00157 /**  Get the offset of the NDRange
00158
00159 \todo Add to the specification
00160 */
00161   size_t get_offset(int dimension)const { return get_offset()[dimension]; }
00162
00163
00164 /// \todo Also provide this access to the current nd_range
00165   nd_range<Dimensions> get_nd_range()const { return ndr; }
00166
00167
00168 /** Get a linearized version of the group ID
00169
00170 */
00171   size_t get_linear_id()const {
00172     return detail::linear_id(get_group_range(), get_id());
00173   }
00174
00175
00176 /** Loop on the work-items inside a work-group
00177 */
00178   void parallel_for_work_item(std::function<void(h_item<rank()>)> f)
00179 const {
00180     detail::parallel_for_workitem_in_group(*this, f);
00181   }
00182
00183   /* Comparison operators for group object.
00184 */
00185   bool operator==(const group &groupB)const {
00186     return (group_id == groupB.group_id &&
00187             ndr == groupB.ndr);
00188   }
00189 };
00190
00191 /// @} End the parallelism Doxygen group
00192
```

```
00193 }
00194
00195 /*
00196 # Some Emacs stuff:
00197 ### Local Variables:
00198 ### ispell-local-dictionary:  "american"
00199 ### eval:  (flyspell-prog-mode)
00200 ### End:
00201 */
00202
00203 #endif // TRISYCL_SYCL_GROUP_HPP
```

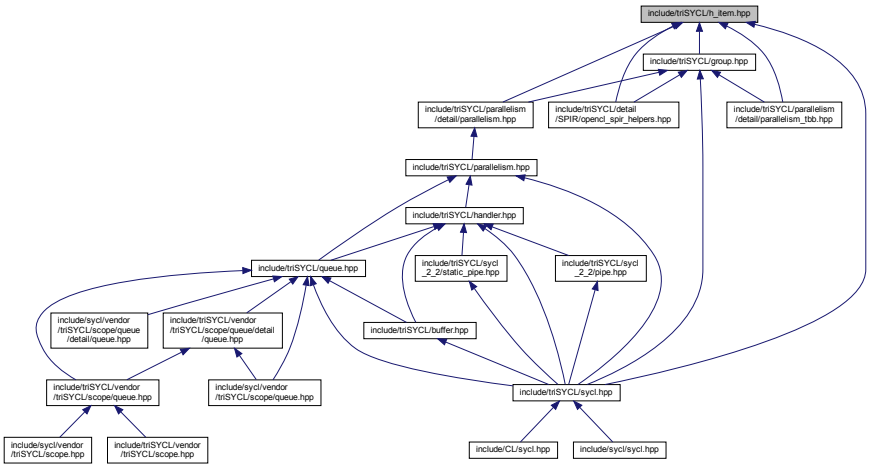
13.121 include/triSYCL/h_item.hpp File Reference

```
#include <cstddef>
#include "triSYCL/access.hpp"
#include "triSYCL/detail/linear_id.hpp"
#include "triSYCL/detail/unimplemented.hpp"
#include "triSYCL/id.hpp"
#include "triSYCL/item.hpp"
#include "triSYCL/nd_range.hpp"
#include "triSYCL/range.hpp"
```

Include dependency graph for h_item.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::h_item< Dimensions >

  h_item<int dimensions> identifies an instance of a group::parallel_for_work_item function object executing at each point in a local range<int dimensions> passed to a parallel_for_work_item call or to the corresponding parallel_for_work_group call if no range is passed to the parallel_for_work_item call. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

13.122 h_item.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_H_ITEM_HPP
00002 #define TRISYCL_SYCL_H_ITEM_HPP
00003
00004 /** \file The OpenCL SYCL h_item<>
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <cstddef>
00013
00014 #include "triSYCL/access.hpp"
00015 #include "triSYCL/detail/linear_id.hpp"
00016 #include "triSYCL/detail/unimplemented.hpp"
00017 #include "triSYCL/id.hpp"
00018 #include "triSYCL/item.hpp"
00019 #include "triSYCL/nd_range.hpp"
00020 #include "triSYCL/range.hpp"
00021
00022 namespace trisycl {
00023
00024 /** \addtogroup parallelism Expressing parallelism through kernels
00025 @{
00026 */
00027
00028 /** \c h_item<int dimensions> identifies an instance of a \c
00029 group::parallel_for_work_item function object executing at each
00030 point in a local \c range<int dimensions> passed to a \c
00031 parallel_for_work_item call or to the corresponding \c
00032 parallel_for_work_group call if no range is passed to the \c
00033 parallel_for_work_item call.  It encapsulates enough information to
00034 identify the work-item's local and global items according to the
00035 information given to \c parallel_for_work_group (physical ids) as
00036 well as the work-item's logical local items in the flexible
00037 range.  All returned items objects are offset-less.  Instances of
00038 the \c h_item<int dimensions> class are not user-constructible and
00039 are passed by the runtime to each instance of the function object.
00040 */
00041 template <int Dimensions = 1>
00042 struct h_item :  boost::equality_comparable<h_item<Dimensions>> {
00043 /// The number of dimensions of the h_item
00044   static auto constexpr rank() { return Dimensions; }
00045
00046 private:
00047
00048   id<Dimensions> global_index;
00049   /* This is a cached value since it can be computed from global_index and
00050 ND_range */
00051   id<Dimensions> local_index;
00052   nd_range<Dimensions> ND_range;
00053
00054 public:
00055
00056 /** Create an empty nd_item<> from an nd_range<>
00057
```

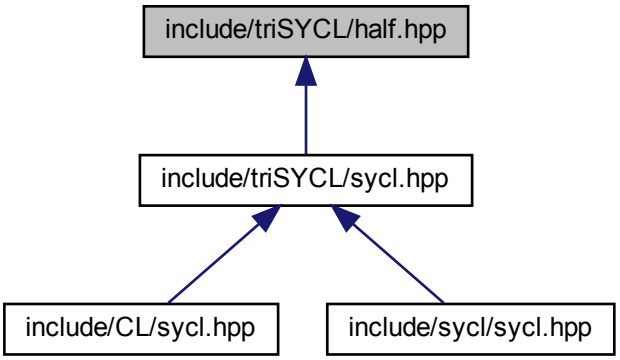
```
00058 \todo This is for the triSYCL implementation which is expected to
00059 call set_global() and set_local() later.  This should be hidden to
00060 the user.
00061 */
00062   h_item(nd_range<Dimensions> ndr) :  ND_range { ndr } {}
00063
00064
00065 /** Create a full nd_item
00066
00067 \todo This is for validation purpose.  Hide this to the programmer
00068 somehow
00069 */
00070   h_item(id<Dimensions> global_index,
00071          nd_range<Dimensions> ndr) :
00072     global_index { global_index },
00073     // Compute the local index using the offset and the group size
00074     local_index
00075       { (global_index - ndr.get_offset())%id<Dimensions> { ndr.get_local() } },
00076     ND_range { ndr }
00077   {}
00078
00079
00080 /** To be able to copy and assign nd_item, use default constructors too
00081
00082 \todo Make most of them protected, reserved to implementation
00083 */
00084   h_item() = default;
00085
00086
00087 /** Return the constituent global id representing the work-item's
00088 position in the global iteration space
00089 */
00090   id<Dimensions> get_global_id()const { return global_index; }
00091
00092
00093 /** Return the constituent element of the global id representing the
00094 work-item's position in the global iteration space in the given
00095 dimension
00096 */
00097   size_t get_global_id(int dimension)const {
00098     return get_global_id()[dimension];
00099   }
00100
00101
00102 /** Return the flattened id of the current work-item after subtracting
00103 the offset
00104 */
00105   size_t get_global_linear_id()const {
00106     return detail::linear_id(get_global_range(), get_global_id(), get_offset());
00107   }
00108
00109
00110 /** Return the constituent local id representing the work-item's
00111 position within the current work-group
00112 */
00113   id<Dimensions> get_local_id()const { return local_index; }
00114
00115
00116 /** Return the constituent element of the local id representing the
00117 work-item's position within the current work-group in the given
00118 dimension
00119 */
00120   size_t get_local_id(int dimension)const { return get_local_id()[dimension]; }
00121
00122
00123 /** Return the flattened id of the current work-item within the current
00124 work-group
00125 */
00126   size_t get_local_linear_id()const {
00127     return detail::linear_id(get_local_range(), get_local_id());
00128   }
00129
00130
00131 /** Return the constituent group group representing the work-group's
00132 position within the overall nd_range
00133 */
00134   id<Dimensions> get_group()const {
00135     /* Convert get_local_range() to an id<> to remove ambiguity into using
00136 implicit conversion either from range<> to id<> or the opposite */
00137     return get_global_id()/id<Dimensions> { get_local_range() };
00138   }
00139
00140
00141 /** Return the constituent element of the group id representing the
00142 work-group's position within the overall nd_range in the given
00143 dimension.
00144 */
```

```
00145   size_t get_group(int dimension)const {
00146     return get_group()[dimension];
00147   }
00148
00149
00150 /// Return the flattened id of the current work-group
00151   size_t get_group_linear_id()const {
00152     return detail::linear_id(get_group_range(), get_group());
00153   }
00154
00155
00156 /// Return the number of groups in the nd_range
00157   id<Dimensions> get_group_range()const {
00158     return get_nd_range().get_group_range();
00159   }
00160
00161 /// Return the number of groups for dimension in the nd_range
00162   size_t get_group_range(int dimension)const {
00163      return get_group_range()[dimension];
00164   }
00165
00166
00167 /// Return a range<> representing the dimensions of the nd_range<>
00168   range<Dimensions> get_global_range()const {
00169     return get_nd_range().get_global_range();
00170   }
00171
00172
00173 /// Return a range<> representing the dimensions of the current work-group
00174   range<Dimensions> get_local_range()const {
00175     return get_nd_range().get_local_range();
00176   }
00177
00178
00179 /** Return an id<> representing the n-dimensional offset provided to the
00180 constructor of the nd_range<> and that is added by the runtime to the
00181 global-ID of each work-item
00182 */
00183   id<Dimensions> get_offset()const { return get_nd_range().get_offset(); }
00184
00185
00186 /// Return the nd_range<> of the current execution
00187   nd_range<Dimensions> get_nd_range()const { return ND_range; }
00188
00189
00190 /** Allows projection down to an item
00191
00192 \todo Add to the specification
00193 */
00194   item<Dimensions> get_item()const {
00195     return { get_global_range(), get_global_id(), get_offset() };
00196   }
00197
00198
00199 /** Execute a barrier with memory ordering on the local address space,
00200 global address space or both based on the value of flag
00201
00202 The current work-item will wait at the barrier until all work-items
00203 in the current work-group have reached the barrier.
00204
00205 In addition, the barrier performs a fence operation ensuring that all
00206 memory accesses in the specified address space issued before the
00207 barrier complete before those issued after the barrier
00208 */
00209   void barrier(access::fence_space flag =
00210                access::fence_space::global_and_local)const {
00211 #if defined(_OPENMP) && !defined(TRISYCL_NO_BARRIER)
00212     /* Use OpenMP barrier in the implementation with 1 OpenMP thread per
00213 work-item of the work-group */
00214 #pragma omp barrier
00215 #else
00216     // \todo To be implemented efficiently otherwise
00217     TRISYCL_UNIMPL;
00218 #endif
00219   }
00220
00221
00222   // For the triSYCL implementation, need to set the local index
00223   void set_local(id<Dimensions> Index) { local_index = Index; }
00224
00225
00226   // For the triSYCL implementation, need to set the global index
00227   void set_global(id<Dimensions> Index) { global_index = Index; }
00228
00229 /// comparison operators
00230   bool operator==(const h_item<Dimensions> &itemB)const {
00231     return (ND_range == itemB.ND_range &&
```

```
00232             global_index == itemB.global_index);
00233   }
00234 };
00235
00236 /// @} End the parallelism Doxygen group
00237
00238 }
00239
00240 /*
00241 # Some Emacs stuff:
00242 ### Local Variables:
00243 ### ispell-local-dictionary:  "american"
00244 ### eval:  (flyspell-prog-mode)
00245 ### End:
00246 */
00247
00248 #endif // TRISYCL_SYCL_H_ITEM_HPP
```

13.123 include/triSYCL/half.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::half

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

13.124 half.hpp

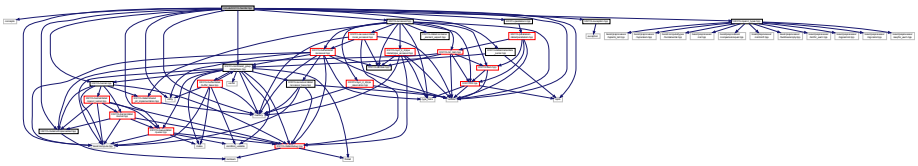
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_HALF_HPP
00002 #define TRISYCL_SYCL_HALF_HPP
00003
00004 /** \file SYCL Half support - dummy
00005 This file is distributed under the University of Illinois Open Source
00006 License.  See LICENSE.TXT for details.
00007 */
00008
00009 /* Dummy half implementation - this is enough to get the CTS to build.  */
00010 namespace trisycl {
00011
00012 class half {
00013 public:
00014   half(int) {};
00015
00016   bool operator>(const half &h1) { return false; };
00017 };
00018
00019 }
00020
00021 #endif
```

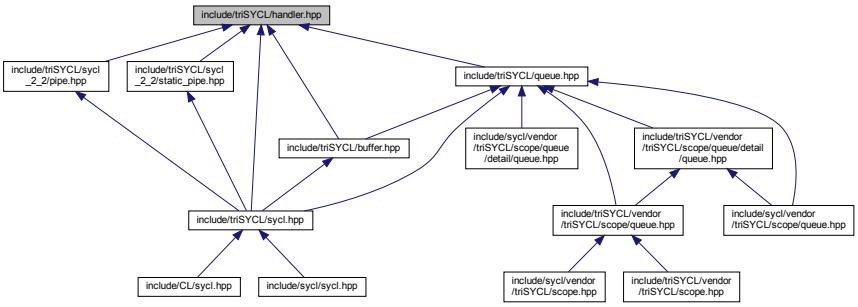
13.125 include/triSYCL/handler.hpp File Reference

```
#include <concepts>
#include <cstddef>
#include <memory>
#include <tuple>
#include <utility>
#include <boost/compute.hpp>
#include "triSYCL/accessor.hpp"
#include "triSYCL/command_group/detail/task.hpp"
#include "triSYCL/detail/instantiate_kernel.hpp"
#include "triSYCL/detail/unimplemented.hpp"
#include "triSYCL/exception.hpp"
#include "triSYCL/kernel.hpp"
#include "triSYCL/opencl_types.hpp"
#include "triSYCL/parallelism.hpp"
#include "triSYCL/queue/detail/queue.hpp"
```

Include dependency graph for handler.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::handler

  Command group handler class. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Functions

- static std::shared_ptr< detail::task > trisycl::detail::add_buffer_to_task (handler *command_group_handler, std::shared_ptr< detail::buffer_base > b, bool is_write_mode)

  Register a buffer as used by a task.

13.126 handler.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_HANDLER_HPP
00002 #define TRISYCL_SYCL_HANDLER_HPP
00003
00004 /** \file The OpenCL SYCL command group handler
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <concepts>
00013 #include <cstddef>
00014 #include <memory>
00015 #include <tuple>
00016 #include <utility>
00017
00018 #ifdef TRISYCL_OPENCL
00019 #include <boost/compute.hpp>
00020 #endif
00021
```

```
00022 #include "triSYCL/accessor.hpp"
00023 #include "triSYCL/command_group/detail/task.hpp"
00024 #include "triSYCL/detail/instantiate_kernel.hpp"
00025 #include "triSYCL/detail/unimplemented.hpp"
00026 #include "triSYCL/exception.hpp"
00027 #include "triSYCL/kernel.hpp"
00028 #include "triSYCL/opencl_types.hpp"
00029 #include "triSYCL/parallelism.hpp"
00030 #include "triSYCL/queue/detail/queue.hpp"
00031
00032 namespace trisycl {
00033
00034 /** \addtogroup execution Platforms, contexts, devices and queues
00035 @{
00036 */
00037
00038 /** Command group handler class
00039
00040 A command group handler object can only be constructed by the SYCL runtime.
00041
00042 All of the accessors defined in the command group scope take as a
00043 parameter an instance of the command group handler and all the kernel
00044 invocation functions are methods of this class.
00045 */
00046 class handler :  public detail::debug<handler> {
00047
00048 public:
00049
00050 /** Attach the task and accessors to it.
00051 */
00052   std::shared_ptr<detail::task> task;
00053
00054
00055   /* Create a command group handler from the queue detail
00056
00057 The queue detail is used to track kernel completion.
00058
00059 Note that this is an implementation dependent constructor.  Normal
00060 users cannot construct handler from scratch.
00061
00062 \todo Make this constructor private
00063 */
00064   handler(const std::shared_ptr<detail::queue> &q) {
00065     // Create a new task for this command_group
00066     task = std::make_shared<detail::task>(q);
00067   }
00068
00069
00070 #ifdef TRISYCL_OPENCL
00071 /** Set accessor kernel arg for an OpenCL kernel which is used through the
00072 SYCL/OpenCL interop interface
00073
00074 The index value specifies which parameter of the OpenCL kernel is
00075 being set and the accessor object, which OpenCL buffer or image is
00076 going to be given as kernel argument.
00077
00078 \todo Update the specification to use a ref && to the accessor instead?
00079
00080 \todo It is not that clean to have set_arg() associated to a
00081 command handler.  Rethink the specification?
00082
00083 \todo It seems more logical to have these methods on kernel instead
00084 */
00085   template <typename DataType,
00086             int Dimensions,
00087             access::mode Mode,
00088             access::target Target = access::target::global_buffer>
00089   void set_arg(int arg_index,
00090                accessor<DataType, Dimensions, Mode, Target> && acc_obj) {
00091     /* Think about setting the kernel argument before actually calling
00092 the kernel.
00093
00094 Explicitly capture task by copy instead of having this captured
00095 by reference and task by reference by side effect */
00096     task->add_prelude([=, task = task] {
00097         task->set_arg(arg_index, acc_obj.implementation->get_cl_buffer());
00098       });
00099   }
00100
00101
00102 /** Set kernel args for an OpenCL kernel which is used through the
00103 SYCL/OpenCL interoperability interface with a wrapper type
00104 */
00105   template <typename T, typename = std::enable_if_t<is_wrapper<T>::value> >
00106   void set_arg(int arg_index, T && scalar_value) {
00107     /* Explicitly capture task by copy instead of having this captured
00108 by reference and task by reference by side effect */
```

```
00109     task->add_prelude([=, task = task] {
00110         task->set_arg(arg_index, scalar_value.unwrap());
00111       });
00112   }
00113
00114
00115 /** Set kernel args for an OpenCL kernel which is used through the
00116 SYCL/OpenCL interoperability interface without a wrapper type
00117 */
00118   template <typename T>
00119   std::enable_if_t<!is_wrapper<T>::value>
00120   set_arg(int arg_index, T && scalar_value) {
00121     /* Explicitly capture task by copy instead of having this captured
00122 by reference and task by reference by side effect */
00123     task->add_prelude([=, task = task] {
00124         task->set_arg(arg_index, scalar_value);
00125       });
00126   }
00127
00128 private:
00129
00130 /// Helper to individually call set_arg() for each argument
00131   template <std::size_t...  Is, typename...  Ts>
00132   void dispatch_set_arg(std::index_sequence<Is...>, Ts&&...  args) {
00133     // Use an intermediate tuple to ease individual argument access
00134     auto &&t = std::make_tuple(std::forward<Ts>(args)...);
00135     // Dispatch individual set_arg() for each argument
00136     auto just_to_evaluate = {
00137       0 /*< At least 1 element to deal with empty set_args() */,
00138       ( set_arg(Is, std::forward<Ts>(std::get<Is>(t))), 0)...
00139     };
00140     // Remove the warning about unused variable
00141     static_cast<void>(just_to_evaluate);
00142   }
00143
00144 public:
00145
00146 /** Set all kernel args for an OpenCL kernel which is used through the
00147 SYCL/OpenCL interop interface
00148
00149 \todo Update the specification to add this function according to
00150 https://cvs.khronos.org/bugzilla/show_bug.cgi?id=15978 proposal
00151 */
00152   template <typename...  Ts>
00153   void set_args(Ts &&...  args) {
00154     /* Construct a set of increasing argument index to be able to call
00155 the real set_arg */
00156     dispatch_set_arg(std::index_sequence_for<Ts...>{},
00157                      std::forward<Ts>(args)...);
00158   }
00159 #endif
00160
00161 private:
00162
00163 /** Schedule the kernel
00164
00165 Add a traced version of the kernel in host mode or add the
00166 kernel in an instantiating function for later extraction by the
00167 compiler
00168 */
00169   template <typename KernelName,
00170             typename Kernel>
00171   void schedule_kernel(Kernel k) {
00172     /* Explicitly capture task by copy instead of having this captured
00173 by reference and task by reference by side effect */
00174     task->schedule(detail::trace_kernel<KernelName>([=, t = task] () mutable {
00175           if (t->owner_queue->is_host())
00176             k();
00177           else {
00178             TRISYCL_DUMP_T("schedule_kernel &k = " « (void *) &k);
00179
00180             // Structure the kernel code so it can be outlined and called
00181             detail::launch_device_kernel<KernelName>(*t, k);
00182             // \todo for now only deal with 1 physical work-item only
00183             t->get_kernel().single_task(t, t->get_queue());
00184           }
00185         }));
00186   }
00187
00188 /** Schedule a parallel for kernel
00189
00190 \todo Add host fall-back execution for parallel_for_kernel
00191
00192 Add a traced version of the kernel in host mode or add the
00193 kernel in an instantiating function for later extraction by the
00194 compiler and then execute it using parallel_for
00195 */
```

```
00196   template <typename KernelName,
00197             typename Kernel,
00198             int N>
00199   void schedule_parallel_for_kernel(Kernel k, const range<N> &num_work_items) {
00200     task->schedule(detail::trace_kernel<KernelName>([=, t = task] () mutable {
00201           // if (t->owner_queue->is_host())
00202              // k();
00203           // else {
00204             TRISYCL_DUMP_T("schedule_parallel_for_kernel &k = " « (void *) &k);
00205
00206             // Structure the kernel code so it can be outlined and called
00207             detail::launch_device_kernel<KernelName>(*t, k);
00208
00209             t->get_kernel().parallel_for(t, t->get_queue(), num_work_items);
00210       // }
00211     }));
00212   }
00213
00214 public:
00215
00216 /** Kernel invocation method of a kernel defined as a lambda or
00217 functor.  If it is a lambda function or the functor type is globally
00218 visible there is no need for the developer to provide a kernel name type
00219 (typename KernelName) for it, as described in 3.5.3
00220
00221 SYCL single_task launches a computation without parallelism at
00222 launch time.
00223
00224 \param F specify the kernel to be launched as a single_task
00225
00226 \param KernelName is a class type that defines the name to be used for
00227 the underlying kernel
00228 */
00229   template <typename KernelName = std::nullptr_t,
00230             typename ParallelForFunctor>
00231   void single_task(ParallelForFunctor &&f) {
00232     TRISYCL_DUMP_T("single_task &f = " « (void *) &f);
00233
00234     schedule_kernel<KernelName>(std::forward<ParallelForFunctor>(f));
00235   }
00236
00237
00238 /** SYCL parallel_for launches a data parallel computation with
00239 parallelism specified at launch time by a range<>
00240
00241 Kernel invocation method of a kernel defined as a lambda or functor,
00242 for the specified range and given an id or item for indexing in the
00243 indexing space defined by range.
00244
00245 If it is a lambda function or the if the functor type is globally
00246 visible there is no need for the developer to provide a kernel name
00247 type (typename KernelName) for it.
00248
00249 \param global_size is the full size of the range<>
00250
00251 \tparam Dims dimensionality of the iteration space
00252
00253 \tparam KernelName is a class type that defines the name to be used
00254 for the underlying kernel
00255
00256 \param f is the kernel functor to execute
00257
00258 \tparam ParallelForFunctor is the type of the kernel functor
00259 */
00260   template <typename KernelName = std::nullptr_t, int Dims,
00261             typename ParallelForFunctor>
00262   // Do not land here if we are using the sycl::kernel API
00263   requires (!std::derived_from<ParallelForFunctor, kernel>)
00264   void parallel_for(const range<Dims>& global_size, ParallelForFunctor f) {
00265     if constexpr (detail::use_native_work_item) {
00266       // Use a normal parallel for
00267       schedule_parallel_for_kernel<KernelName>(
00268           [=] { detail::parallel_for(global_size, f); }, global_size);
00269     } else
00270       // Launch a single-task kernel containing the loop nests
00271       schedule_kernel<KernelName>(
00272           [=] { detail::parallel_for(global_size, f); });
00273   }
00274
00275 /** SYCL parallel_for launches a data parallel computation with
00276 parallelism specified at launch time with a range defined with a
00277 { dim1, dim2, dim3...  } syntax
00278
00279 Kernel invocation method of a kernel defined as a lambda or functor,
00280 for the specified range and given an id or item for indexing in the
00281 indexing space defined by range.
00282
```

```
00283 If it is a lambda function or the if the functor type is globally
00284 visible there is no need for the developer to provide a kernel name
00285 type (typename KernelName) for it.
00286
00287 \param global_size is the full size of the range<> defined as a
00288 C array to allow type deduction from a list-initialization
00289 with { args...  }
00290
00291 \tparam Dims dimensionality of the iteration space
00292
00293 \tparam KernelName is a class type that defines the name to be used
00294 for the underlying kernel
00295
00296 \param f is the kernel functor to execute
00297
00298 \tparam ParallelForFunctor is the type of the kernel functor
00299 */
00300   template <typename KernelName = std::nullptr_t, int Dims,
00301             typename ParallelForFunctor>
00302   void parallel_for(const std::size_t (&global_size)[Dims],
00303                     ParallelForFunctor f) {
00304     parallel_for<KernelName>(range<Dims> { global_size }, f);
00305   }
00306
00307 /** SYCL parallel_for launches a data parallel computation with
00308 parallelism specified at launch time with a range defined with a
00309 single dimension as an integer
00310
00311 Kernel invocation method of a kernel defined as a lambda or functor,
00312 for the specified range and given an id or item for indexing in the
00313 indexing space defined by range.
00314
00315 If it is a lambda function or the if the functor type is globally
00316 visible there is no need for the developer to provide a kernel name
00317 type (typename KernelName) for it.
00318
00319 \param global_size is the full size of the range<1> defined as an
00320 integer
00321
00322 \tparam KernelName is a class type that defines the name to be used
00323 for the underlying kernel
00324
00325 \param f is the kernel functor to execute
00326
00327 \tparam ParallelForFunctor is the type of the kernel functor
00328 */
00329   template <typename KernelName = std::nullptr_t,
00330             typename ParallelForFunctor>
00331   // Do not land here if we are using the sycl::kernel API
00332   requires (!std::derived_from<ParallelForFunctor, kernel>)
00333   void parallel_for(std::size_t global_size,
00334                     ParallelForFunctor f) {
00335     parallel_for<KernelName>(range { global_size }, f);
00336   }
00337
00338 /** Kernel invocation method of a kernel defined as a lambda or functor,
00339 for the specified range and offset and given an id or item for
00340 indexing in the indexing space defined by range
00341
00342 If it is a lambda function or the if the functor type is globally
00343 visible there is no need for the developer to provide a kernel name
00344 type (typename KernelName) for it, as described in detail in 3.5.3
00345
00346 \param global_size is the global size of the range<>
00347
00348 \param offset is the offset to be add to the id<> during iteration
00349
00350 \param f is the kernel functor to execute
00351
00352 \param ParallelForFunctor is the kernel functor type
00353
00354 \param KernelName is a class type that defines the name to be used for
00355 the underlying kernel
00356 */
00357   template <typename KernelName = std::nullptr_t, int Dims,
00358             typename ParallelForFunctor>
00359   void parallel_for(range<Dims> global_size, id<Dims> offset,
00360                     ParallelForFunctor f) {
00361     schedule_kernel<KernelName>(
00362         [=] { detail::parallel_for_global_offset(global_size, offset, f); });
00363   }
00364
00365 /** Kernel invocation method of a kernel defined as a lambda or functor,
00366 for the specified nd_range and given an nd_item for indexing in the
00367 indexing space defined by the nd_range
00368
00369 If it is a lambda function or the if the functor type is globally
```

```
00370 visible there is no need for the developer to provide a kernel name
00371 type (typename KernelName) for it, as described in detail in 3.5.3
00372
00373 \param r defines the iteration space with the work-group layout and
00374 offset
00375
00376 \param Dimensions dimensionality of the iteration space
00377
00378 \param f is the kernel functor to execute
00379
00380 \param ParallelForFunctor is the kernel functor type
00381
00382 \param KernelName is a class type that defines the name to be used for
00383 the underlying kernel
00384 */
00385   template <typename KernelName = std::nullptr_t,
00386             int Dimensions,
00387             typename ParallelForFunctor>
00388   void parallel_for(nd_range<Dimensions> r,
00389                     ParallelForFunctor f) {
00390     schedule_kernel<KernelName>([=] { detail::parallel_for(r, f); });
00391   }
00392
00393
00394 /** Hierarchical kernel invocation method of a kernel defined as a
00395 lambda encoding the body of each work-group to launch
00396
00397 May contain multiple kernel built-in parallel_for_work_item
00398 functions representing the execution on each work-item.
00399
00400 Launch num_work_groups work-groups of runtime-defined
00401 size.  Described in detail in 3.5.3.
00402
00403 \param r defines the iteration space with the work-group layout and
00404 offset
00405
00406 \param Dimensions dimensionality of the iteration space
00407
00408 \param f is the kernel functor to execute
00409
00410 \param ParallelForFunctor is the kernel functor type
00411
00412 \param KernelName is a class type that defines the name to be used for
00413 the underlying kernel
00414 */
00415   template <typename KernelName = std::nullptr_t,
00416             int Dimensions = 1,
00417             typename ParallelForFunctor>
00418   void parallel_for_work_group(nd_range<Dimensions> r,
00419                                ParallelForFunctor f) {
00420     schedule_kernel<KernelName>([=] {
00421         detail::parallel_for_workgroup(r, f);
00422       });
00423   }
00424
00425
00426 /** Hierarchical kernel invocation method of a kernel defined as a
00427 lambda encoding the body of each work-group to launch
00428
00429 May contain multiple kernel built-in parallel_for_work_item
00430 functions representing the execution on each work-item.
00431
00432 Launch num_work_groups work-groups of runtime-defined
00433 size.  Described in detail in 3.5.3.
00434
00435 \param r defines the iteration space with the work-group layout and
00436 offset
00437
00438 \param Dimensions dimensionality of the iteration space
00439
00440 \param f is the kernel functor to execute
00441
00442 \param ParallelForFunctor is the kernel functor type
00443
00444 \param KernelName is a class type that defines the name to be used for
00445 the underlying kernel
00446 */
00447   template <typename KernelName = std::nullptr_t,
00448             int Dimensions = 1,
00449             typename ParallelForFunctor>
00450   void parallel_for_work_group(range<Dimensions> r1, range<Dimensions> r2,
00451                                ParallelForFunctor f) {
00452     parallel_for_work_group(nd_range<Dimensions> { r1, r2 }, f);
00453   }
00454
00455
00456 /** Kernel invocation method of a kernel defined as pointer to a kernel
```

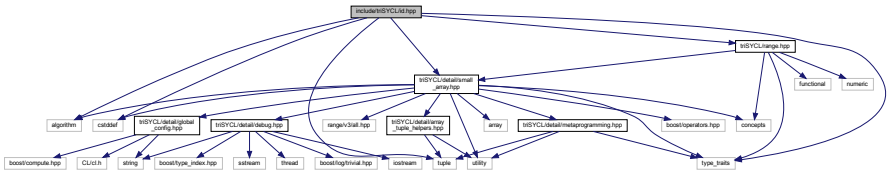
```
00457 object, described in detail in 3.5.3
00458
00459 \todo Add in the spec a version taking a kernel and a functor,
00460 to have host fall-back
00461 */
00462   void single_task(kernel sycl_kernel) {
00463     /* For now just use the usual host task system to schedule
00464 manually the OpenCL kernels instead of using OpenCL event-based
00465 scheduling
00466
00467 \todo Move the tracing inside the kernel implementation
00468
00469 \todo Simplify this 2 step ugly interface
00470 */
00471     task->set_kernel(sycl_kernel.implementation);
00472     task->schedule(detail::trace_kernel<kernel>([=, t = task] {
00473           sycl_kernel.implementation->single_task(t, t->get_queue());
00474         }));
00475   }
00476
00477
00478 /** Kernel invocation method of a kernel defined as a kernel object,
00479 for the specified range and given an id or item for indexing in
00480 the indexing space defined by range, described in detail in
00481 5.4.
00482
00483 \todo Add in the spec a version taking a kernel and a functor,
00484 to have host fall-back
00485 */
00486   template <int Dimensions = 1, typename SYCLrange>
00487     requires(std::derived_from<SYCLrange, range<Dimensions>> ||
00488              std::integral<SYCLrange>)
00489   void parallel_for(SYCLrange num_work_items, kernel sycl_kernel) {
00490     /* For now just use the usual host task system to schedule
00491 manually the OpenCL kernels instead of using OpenCL event-based
00492 scheduling
00493
00494 todo Move the tracing inside the kernel implementation
00495
00496 todo Simplify this 2 step ugly interface
00497 */
00498     task->set_kernel(sycl_kernel.implementation);
00499     /* Use an intermediate variable to capture task by copy because
00500 otherwise "this" is captured by reference and havoc with task
00501 just accessing the dead "this".  Nasty bug to find...  */
00502     task->schedule(detail::trace_kernel<kernel>([=, t = task] {
00503       sycl_kernel.implementation->parallel_for(t, t->get_queue(),
00504                                                detail::rangeify(num_work_items));
00505     }));
00506   }
00507
00508 /** Kernel invocation method of a kernel defined as pointer to a kernel
00509 object, for the specified nd_range and given an nd_item for indexing
00510 in the indexing space defined by the nd_range, described in detail
00511 in 3.5.3
00512
00513 \todo Add in the spec a version taking a kernel and a functor,
00514 to have host fall-back
00515
00516 \todo To be implemented
00517 */
00518   template <int Dimensions = 1>
00519   void parallel_for(nd_range<Dimensions>, kernel syclKernel) {
00520     TRISYCL_UNIMPL;
00521   }
00522
00523 };
00524
00525 namespace detail {
00526
00527 /** Register a buffer as used by a task
00528
00529 This is a proxy function to avoid complicated type recursion.
00530 */
00531 static std::shared_ptr<detail::task>
00532 add_buffer_to_task(handler *command_group_handler,
00533                    std::shared_ptr<detail::buffer_base> b,
00534                    bool is_write_mode) {
00535   command_group_handler->task->add_buffer(b, is_write_mode);
00536   return command_group_handler->task;
00537 }
00538
00539 }
00540
00541 /// @} End the execution Doxygen group
00542
00543 }
```

```
00544
00545 /*
00546 # Some Emacs stuff:
00547 ### Local Variables:
00548 ### ispell-local-dictionary:  "american"
00549 ### eval:  (flyspell-prog-mode)
00550 ### End:
00551 */
00552
00553 #endif // TRISYCL_SYCL_HANDLER_HPP
```

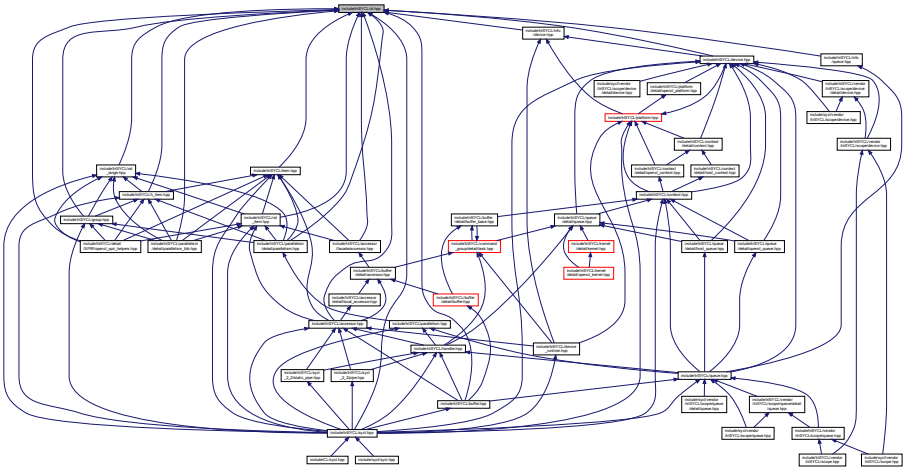
13.127 include/triSYCL/id.hpp File Reference

```
#include <algorithm>
#include <cstddef>
#include <tuple>
#include <type_traits>
#include "triSYCL/detail/small_array.hpp"
#include "triSYCL/range.hpp"
```

Include dependency graph for id.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::id< Dimensions >

  Define a multi-dimensional index, used for example to locate a work item. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

Functions

- template<typename... BasicType>
  trisycl::id (BasicType... Args) -> id< sizeof...(Args)>

  User-defined deduction guides to deduce the rank of an id from the number of arguments.

13.128 id.hpp

Go to the documentation of this file.

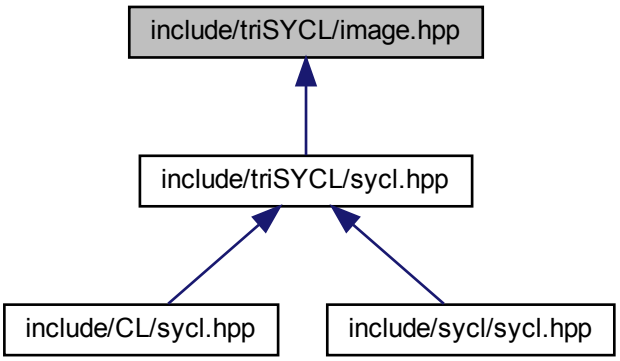
```
00001 #ifndef TRISYCL_SYCL_ID_HPP
00002 #define TRISYCL_SYCL_ID_HPP
00003
00004 /** \file The OpenCL SYCL id<>
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <algorithm>
00013 #include <cstddef>
00014 #include <tuple>
00015 #include <type_traits>
00016
00017 #include "triSYCL/detail/small_array.hpp"
00018 #include "triSYCL/range.hpp"
00019
00020 namespace trisycl {
00021
00022 template <int Dimensions, bool with_offset> class item;
00023
00024 /** \addtogroup parallelism Expressing parallelism through kernels
00025 @{
00026 */
00027
00028 /** Define a multi-dimensional index, used for example to locate a work
00029 item
00030 */
00031 template <int Dimensions = 1>
00032 class id :  public detail::small_array_sycl<
00033               std::size_t,
00034               id<Dimensions>,
00035               Dimensions > {
00036
00037 public:
00038 /// The number of dimensions of the id
00039   static auto constexpr rank() { return Dimensions; }
00040
00041   // Inherit from all the constructors
00042   using detail::small_array_sycl<std::size_t,
00043                                  id<Dimensions>,
00044                                  Dimensions>::small_array_sycl;
00045
00046 /// Construct an id from a range
00047   id(const range<Dimensions>& range_size)
00048 /** Use the fact we have a constructor of a small_array from a another
00049 kind of small_array
00050 */
00051       :  detail::small_array_sycl<std::size_t, id<Dimensions>, Dimensions> {
00052         range_size
00053       } {}
00054
00055 /// Construct an id from an item global_id
00056   id(const item<Dimensions, true> &rhs)
00057     :  detail::small_array_sycl<std::size_t, id<Dimensions>, Dimensions>
00058       { rhs.get_id() }
00059   {}
00060
```

```
00061 /// Default constructor must 0 all elements
00062   id() :  detail::small_array_sycl<std::size_t, id<Dimensions>, Dimensions> { 0 }
00063   {}
00064 };
00065
00066 /**  User-defined deduction guides to deduce the rank of an id from the number
00067 of arguments
00068
00069 For example id { 4, 5, 7 } will define an id<3> { 4, 5, 7 }
00070 */
00071 template <typename...  BasicType> id(BasicType...  Args) -> id<sizeof...(Args)>;
00072
00073 /// @} End the parallelism Doxygen group
00074
00075 }
00076
00077 /*
00078 # Some Emacs stuff:
00079 ### Local Variables:
00080 ### ispell-local-dictionary:  "american"
00081 ### eval:  (flyspell-prog-mode)
00082 ### End:
00083 */
00084
00085 #endif // TRISYCL_SYCL_ID_HPP
```

13.129 include/triSYCL/image.hpp File Reference

OpenCL SYCL image class.

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

13.129.1 Detailed Description

OpenCL SYCL image class.

Ronan at Keryell point FR

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file image.hpp.

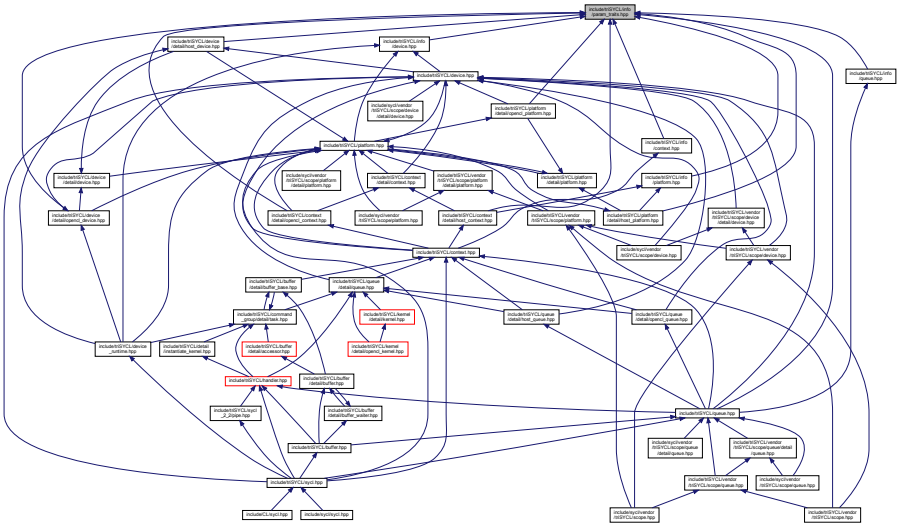
13.130 image.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_IMAGE_HPP
00002 #define TRISYCL_SYCL_IMAGE_HPP
00003
00004 /** \file
00005
00006 OpenCL SYCL image class
00007
00008 Ronan at Keryell point FR
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 namespace trisycl {
00015
00016 /** \addtogroup data
00017
00018 @{
00019 */
00020
00021 /// \todo implement image
00022 template <int Dimensions> struct image;
00023
00024
00025 /// @} End the data Doxygen group
00026
00027
00028 }
00029
00030 /*
00031 # Some Emacs stuff:
00032 ### Local Variables:
00033 ### ispell-local-dictionary:  "american"
00034 ### eval:  (flyspell-prog-mode)
00035 ### End:
00036 */
00037
00038 #endif // TRISYCL_SYCL_IMAGE_HPP
```

13.131 include/triSYCL/info/param_traits.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::info::param_traits< T, Param >

  Implement a meta-function from (T, value) to T' to express the return type value of an OpenCL function of kind (T, value)

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::info

Macros

- #define TRISYCL_INFO_PARAM_TRAITS_ANY_T(T, RETURN_TYPE)

  To declare a param_traits returning RETURN_TYPE for function of any T.
- #define TRISYCL_INFO_PARAM_TRAITS(VALUE, RETURN_TYPE)

  To declare a param_traits returning RETURN_TYPE for function taking a VALUE of type T.

13.131.1 Macro Definition Documentation

13.131.1.1 TRISYCL_INFO_PARAM_TRAITS

```
#define TRISYCL_INFO_PARAM_TRAITS(
              VALUE,
              RETURN_TYPE )
```

Value:

```
  template <>                                                   \
  struct param_traits<decltype(VALUE), VALUE> {                 \
    using return_type = RETURN_TYPE;                            \
  };
```

To declare a param_traits returning RETURN_TYPE for function taking a VALUE of type T.

Definition at line 34 of file param_traits.hpp.

13.131.1.2 TRISYCL_INFO_PARAM_TRAITS_ANY_T

```
#define TRISYCL_INFO_PARAM_TRAITS_ANY_T(
              T,
              RETURN_TYPE )
```

Value:

```
  template <T Param>                                            \
  struct param_traits<T, Param> {                               \
    using return_type = RETURN_TYPE;                            \
  };
```

To declare a param_traits returning RETURN_TYPE for function of any T.

Definition at line 24 of file param_traits.hpp.

13.132 param_traits.hpp

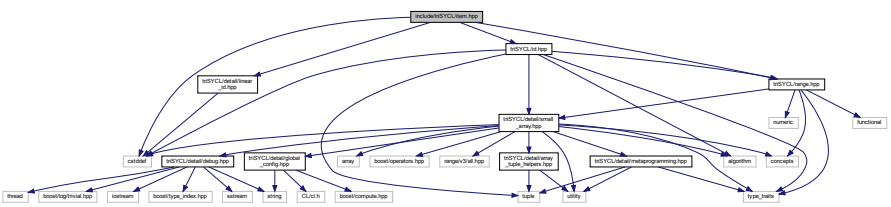
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_INFO_PARAM_TRAITS_HPP
00002 #define TRISYCL_SYCL_INFO_PARAM_TRAITS_HPP
00003
00004 /** \file The OpenCL SYCL param_traits
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 namespace trisycl::info {
00013
00014 /** Implement a meta-function from (T, value) to T' to express the return type
00015 value of an OpenCL function of kind (T, value)
00016 */
00017 template <typename T, T Param>
00018 struct param_traits {
00019   // By default no return type
00020 };
00021
00022
00023 /// To declare a param_traits returning RETURN_TYPE for function of any T
00024 #define TRISYCL_INFO_PARAM_TRAITS_ANY_T(T, RETURN_TYPE)      \
00025 template <T Param>                                            \
00026 struct param_traits<T, Param> {                               \
00027 using return_type = RETURN_TYPE;                            \
00028 };
00029
00030
00031 /** To declare a param_traits returning RETURN_TYPE for function taking a
00032 VALUE of type T
00033 */
00034 #define TRISYCL_INFO_PARAM_TRAITS(VALUE, RETURN_TYPE)        \
00035 template <>                                                   \
00036 struct param_traits<decltype(VALUE), VALUE> {                 \
00037 using return_type = RETURN_TYPE;                            \
00038 };
00039
00040 }
00041
00042 /*
00043 # Some Emacs stuff:
00044 ### Local Variables:
00045 ### ispell-local-dictionary:  "american"
00046 ### eval:  (flyspell-prog-mode)
00047 ### End:
00048 */
00049
00050 #endif // TRISYCL_SYCL_INFO_PARAM_TRAITS_HPP
```

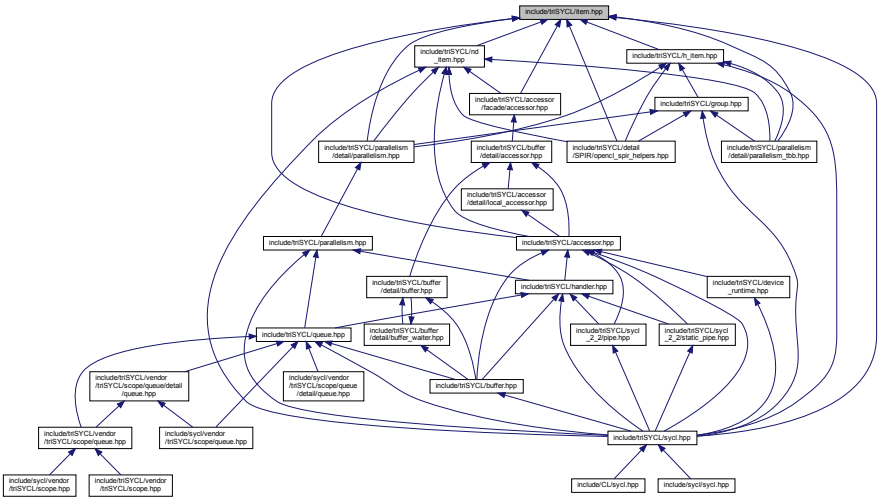
13.133 include/triSYCL/item.hpp File Reference

```
#include <cstddef>
#include "triSYCL/detail/linear_id.hpp"
#include "triSYCL/id.hpp"
#include "triSYCL/range.hpp"
```

Include dependency graph for item.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::item< Dimensions, with_offset >

  A SYCL item stores information on a work-item with some more context such as the definition range and offset. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

13.134 item.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_ITEM_HPP
00002 #define TRISYCL_SYCL_ITEM_HPP
00003
00004 /** \file The OpenCL SYCL item<>
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <cstddef>
00013
00014 #include "triSYCL/detail/linear_id.hpp"
00015 #include "triSYCL/id.hpp"
00016 #include "triSYCL/range.hpp"
00017
00018 namespace trisycl {
00019
00020 /** \addtogroup parallelism Expressing parallelism through kernels
00021 @{
00022 */
00023
00024 /** A SYCL item stores information on a work-item with some more context
00025 such as the definition range and offset.
00026 */
```

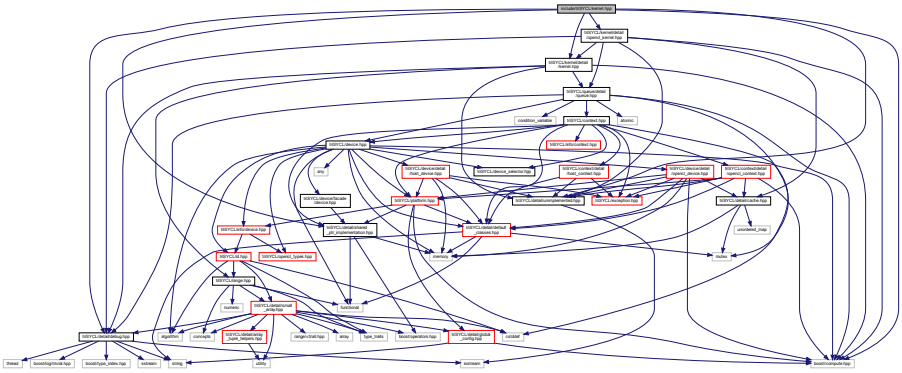
```
00027 template <int Dimensions = 1, bool with_offset = true>
00028 class item :  boost::equality_comparable<item<Dimensions>> {
00029
00030 public:
00031 /// The number of dimensions of the item
00032   static auto constexpr rank() { return Dimensions; }
00033
00034 private:
00035
00036   range<Dimensions> global_range;
00037   id<Dimensions> global_index;
00038   id<Dimensions> offset;
00039
00040 public:
00041
00042 /** Create an item from a local size and an optional offset
00043
00044 This constructor is used by the triSYCL implementation and the
00045 non-regression testing.
00046 */
00047   item(range<Dimensions> global_size,
00048        id<Dimensions> global_index,
00049        id<Dimensions> offset = {}) :
00050     global_range { global_size },
00051     global_index { global_index },
00052     offset { offset }
00053   {}
00054
00055
00056 /** To be able to copy and assign item, use default constructors too
00057
00058 \todo Make most of them protected, reserved to implementation
00059 */
00060   item() = default;
00061
00062
00063 /** Return the constituent local or global id<> representing the
00064 work-item's position in the iteration space
00065 */
00066   id<Dimensions> get_id()const { return global_index; }
00067
00068
00069 /** Return the requested dimension of the constituent id<> representing
00070 the work-item's position in the iteration space
00071 */
00072   size_t get_id(int dimension)const { return get_id()[dimension]; }
00073
00074
00075 /** Return the constituent id<> l-value representing the work-item's
00076 position in the iteration space in the given dimension
00077 */
00078   auto &operator[](int dimension) { return global_index[dimension]; }
00079
00080
00081 /** Returns a range<> representing the dimensions of the range of
00082 possible values of the item
00083 */
00084   range<Dimensions> get_range()const { return global_range; }
00085
00086 /// Return the same value as get_range().get(dimension)
00087   size_t get_range(int dimension)const { return get_range().get(dimension); }
00088
00089 /** Returns an id<> representing the n-dimensional offset provided to
00090 the parallel_for and that is added by the runtime to the global-ID
00091 of each work-item, if this item represents a global range
00092
00093 For an item representing a local range of where no offset was passed
00094 this will always return an id of all 0 values.
00095 */
00096   id<Dimensions> get_offset()const {
00097     static_assert(with_offset == true, "get_offset only callable with has_offset = true");
00098     return offset;
00099   }
00100
00101
00102 /** Return the linearized ID in the item's range
00103
00104 Computed as the flatted ID after the offset is subtracted.
00105 */
00106   size_t get_linear_id()const {
00107     return detail::linear_id(get_range(), get_id(), get_offset());
00108   }
00109
00110
00111 /** For the implementation, need to set the global index
00112
00113 \todo Move to private and add friends
```

```
00114 */
00115   void set(id<Dimensions> Index) { global_index = Index; }
00116
00117 /** Returns an item with same dimensions but offset set to 0 */
00118   operator item<Dimensions, true> ()const {
00119     static_assert(with_offset == false, "get_offset only callable with has_offset = true");
00120     return { this->get_range(), this->get_id(), 0 };
00121   }
00122
00123 /// Display the value for debugging and validation purpose
00124   void display()const {
00125     global_range.display();
00126     global_index.display();
00127     offset.display();
00128   }
00129
00130 /// Comparison operators for item
00131   bool operator==(const item<Dimensions> &itemB)const {
00132     return (global_range == itemB.global_range &&
00133             global_index == itemB.global_index &&
00134             offset == itemB.offset);
00135   }
00136 };
00137
00138 /// @} End the parallelism Doxygen group
00139
00140 }
00141
00142 /*
00143 # Some Emacs stuff:
00144 ### Local Variables:
00145 ### ispell-local-dictionary:  "american"
00146 ### eval:  (flyspell-prog-mode)
00147 ### End:
00148 */
00149
00150 #endif // TRISYCL_SYCL_ITEM_HPP
```

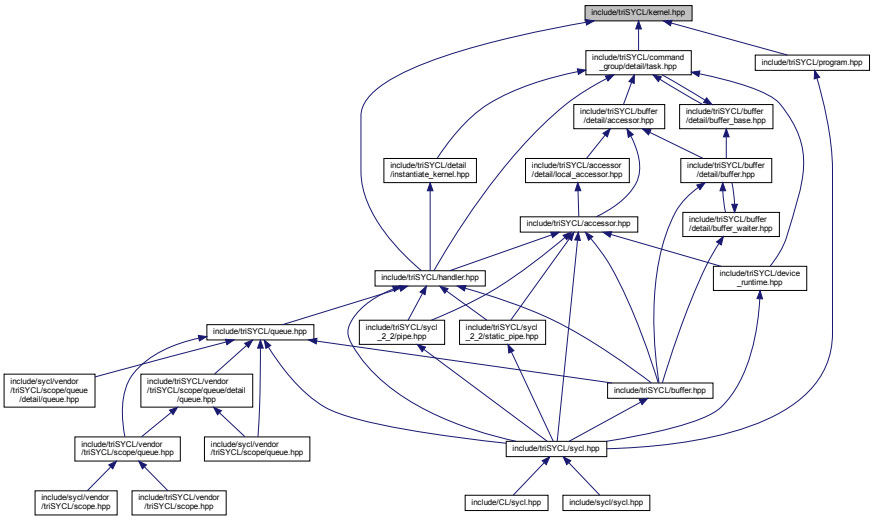
13.135 include/triSYCL/kernel.hpp File Reference

```
#include <boost/compute.hpp>
#include "triSYCL/detail/debug.hpp"
#include "triSYCL/detail/shared_ptr_implementation.hpp"
#include "triSYCL/detail/unimplemented.hpp"
#include "triSYCL/kernel/detail/kernel.hpp"
#include "triSYCL/kernel/detail/opencl_kernel.hpp"
```

Include dependency graph for kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::kernel

  SYCL kernel. More...
- struct std::hash< trisycl::kernel >

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace std

13.136 kernel.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_KERNEL_HPP
00002 #define TRISYCL_SYCL_KERNEL_HPP
00003
00004 /** \file The OpenCL SYCL kernel
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #ifdef TRISYCL_OPENCL
00013 #include <boost/compute.hpp>
00014 #endif
00015
00016 #include "triSYCL/detail/debug.hpp"
00017 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00018 #include "triSYCL/detail/unimplemented.hpp"
00019 //#include "triSYCL/info/kernel.hpp"
00020 #include "triSYCL/kernel/detail/kernel.hpp"
00021 #ifdef TRISYCL_OPENCL
00022 #include "triSYCL/kernel/detail/opencl_kernel.hpp"
00023 #endif
```

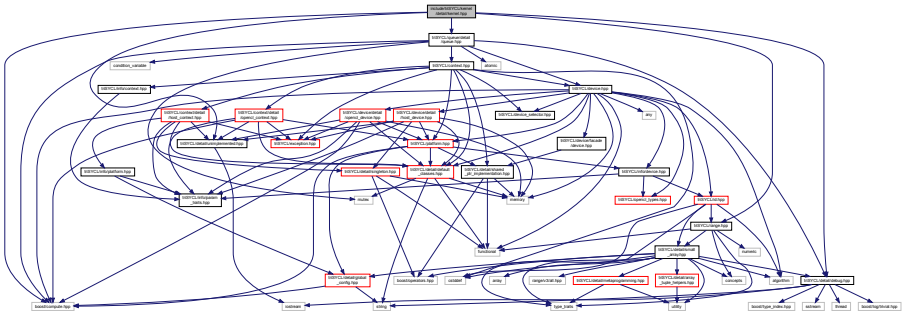
```
00024
00025 namespace trisycl {
00026
00027 /** \addtogroup execution Platforms, contexts, devices and queues
00028 @{
00029 */
00030
00031 /** SYCL kernel
00032
00033 \todo To be implemented
00034
00035 \todo Check specification
00036 */
00037 class kernel
00038   /* Use the underlying kernel implementation that can be shared in
00039 the SYCL model */
00040   :  public detail::shared_ptr_implementation<kernel, detail::kernel> {
00041
00042   // The type encapsulating the implementation
00043   using implementation_t = typename kernel::shared_ptr_implementation;
00044
00045   // The handler class uses the implementation
00046   friend class handler;
00047
00048   // Allows the comparison operation to access the implementation
00049   friend implementation_t;
00050
00051  public:
00052
00053   // Make the implementation member directly accessible in this class
00054   using implementation_t::implementation;
00055
00056 /** The default object is not valid because there is no program or
00057 \code cl_kernel \endcode associated with it */
00058   kernel() = delete;
00059
00060 #ifdef TRISYCL_OPENCL
00061 /** Constructor for SYCL kernel class given an OpenCL kernel object
00062 with set arguments, valid for enqueuing
00063
00064 Retains a reference to the \p cl_kernel object.  The Caller
00065 should release the passed cl_kernel object when it is no longer
00066 needed.
00067 */
00068   kernel(cl_kernel k) :  kernel { boost::compute::kernel { k } } {}
00069
00070
00071 /** Construct a kernel class instance using a boost::compute::kernel
00072
00073 This is a triSYCL extension for boost::compute interoperation.
00074
00075 Return synchronous errors via the SYCL exception class.
00076 */
00077   kernel(const boost::compute::kernel &k)
00078     :  implementation_t { detail::opencl_kernel::instance(k) } {}
00079
00080
00081 /** Return the OpenCL kernel object for this kernel
00082
00083 Retains a reference to the returned cl_kernel object.  Caller
00084 should release it when finished.
00085 */
00086   cl_kernel get()const {
00087     return implementation->get();
00088   }
00089 #endif
00090
00091
00092 #if 0
00093 /// Return the context that this kernel is defined for
00094   //context get_context() const;
00095
00096 /// Return the program that this kernel is part of
00097   //program get_program() const;
00098
00099 /** Query information from the kernel object using the
00100 info::kernel_info descriptor.
00101 */
00102   template <info::kernel param>
00103   typename info::param_traits<info::kernel, param>::type
00104     get_info()const {
00105     detail::unimplemented();
00106   }
00107 #endif
00108
00109 };
00110
```

```
00111 /// @} End the execution Doxygen group
00112
00113 }
00114
00115
00116 /* Inject a custom specialization of std::hash to have the buffer
00117 usable into an unordered associative container
00118
00119 \todo Add this to the spec
00120 */
00121 namespace std {
00122
00123 template <> struct hash<trisycl::kernel> {
00124
00125   auto operator()(const trisycl::kernel &k)const {
00126     // Forward the hashing to the implementation
00127     return k.hash();
00128   }
00129
00130 };
00131
00132 }
00133
00134 /*
00135 # Some Emacs stuff:
00136 ### Local Variables:
00137 ### ispell-local-dictionary:  "american"
00138 ### eval:  (flyspell-prog-mode)
00139 ### End:
00140 */
00141
00142 #endif // TRISYCL_SYCL_KERNEL_HPP
```

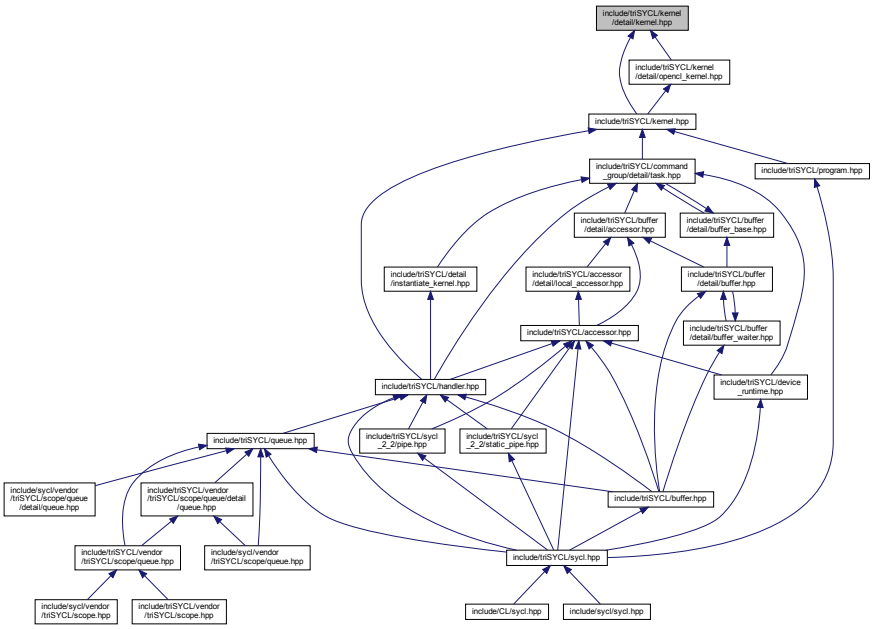
13.137 include/triSYCL/kernel/detail/kernel.hpp File Reference

```
#include <boost/compute.hpp>
#include "triSYCL/detail/debug.hpp"
#include "triSYCL/detail/unimplemented.hpp"
#include "triSYCL/queue/detail/queue.hpp"
#include "triSYCL/range.hpp"
```

Include dependency graph for kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::kernel

  Abstract SYCL kernel. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Macros

- #define TRISYCL_ParallelForKernel_RANGE(N)

  Launch a kernel with a range<>

13.137.1 Macro Definition Documentation

13.137.1.1 TRISYCL_ParallelForKernel_RANGE

```
#define TRISYCL_ParallelForKernel_RANGE(
              N )
```

Value:

```
  virtual void parallel_for(std::shared_ptr<detail::task> task, std::shared_ptr<detail::queue> q,      \
                            const range<N> &num_work_items) = 0;
```

Launch a kernel with a range<>

Do not use a template since it does not work with virtual functions

Todo Think to a cleaner solution

Definition at line 61 of file kernel.hpp.

13.138 kernel.hpp

Go to the documentation of this file.

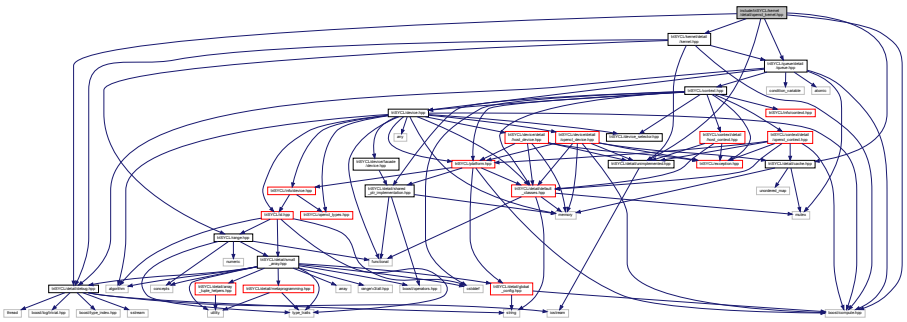
```
00001 #ifndef TRISYCL_SYCL_KERNEL_DETAIL_KERNEL_HPP
00002 #define TRISYCL_SYCL_KERNEL_DETAIL_KERNEL_HPP
00003
00004 /** \file The OpenCL SYCL kernel
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #ifdef TRISYCL_OPENCL
00013 #include <boost/compute.hpp>
00014 #endif
00015
00016 #include "triSYCL/detail/debug.hpp"
00017 #include "triSYCL/detail/unimplemented.hpp"
00018 //#include "triSYCL/info/kernel.hpp"
00019 #include "triSYCL/queue/detail/queue.hpp"
00020 #include "triSYCL/range.hpp"
00021
00022 namespace trisycl::detail {
00023
00024 /** \addtogroup execution Platforms, contexts, devices and queues
00025 @{
00026 */
00027
00028 /// Abstract SYCL kernel
00029 class kernel :  detail::debug<detail::kernel> {
00030
00031  public:
00032
00033 #ifdef TRISYCL_OPENCL
00034 /** Return the OpenCL kernel object for this kernel
00035
00036 Retains a reference to the returned cl_kernel object.  Caller
00037 should release it when finished.
00038 */
00039   virtual cl_kernel get() const = 0;
00040
00041
00042 /** Return the Boost.Compute OpenCL kernel object for this kernel
00043
00044 This is an extension.
00045 */
00046   virtual boost::compute::kernel &get_boost_compute() = 0;
00047 #endif
00048
00049
00050 /// Launch a single task of the kernel
00051   virtual void single_task(std::shared_ptr<detail::task> task,
00052                            std::shared_ptr<detail::queue> q) = 0;
```

```
00053
00054
00055 /** Launch a kernel with a range<>
00056
00057 Do not use a template since it does not work with virtual functions
00058
00059 \todo Think to a cleaner solution
00060 */
00061 #define TRISYCL_ParallelForKernel_RANGE(N)                              \
00062 virtual void parallel_for(std::shared_ptr<detail::task> task, std::shared_ptr<detail::queue> q,     \
00063 const range<N> &num_work_items) = 0;
00064
00065   TRISYCL_ParallelForKernel_RANGE(1)
00066   TRISYCL_ParallelForKernel_RANGE(2)
00067   TRISYCL_ParallelForKernel_RANGE(3)
00068 #undef TRISYCL_ParallelForKernel_RANGE
00069
00070
00071 /// Return the context that this kernel is defined for
00072   //virtual context get_context() const;
00073
00074 /// Return the program that this kernel is part of
00075   //virtual program get_program() const;
00076
00077   // Virtual to call the real destructor
00078   virtual ~kernel() {}
00079
00080 };
00081
00082 /// @} End the execution Doxygen group
00083
00084 }
00085
00086 /*
00087 # Some Emacs stuff:
00088 ### Local Variables:
00089 ### ispell-local-dictionary:  "american"
00090 ### eval:  (flyspell-prog-mode)
00091 ### End:
00092 */
00093
00094 #endif // TRISYCL_SYCL_DETAIL_KERNEL_KERNEL_HPP
```

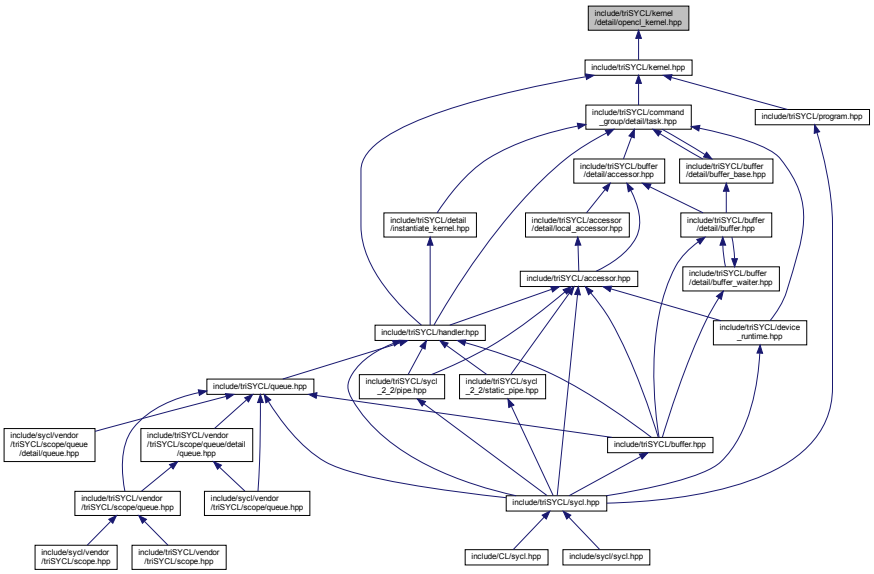
13.139 include/triSYCL/kernel/detail/opencl_kernel.hpp File Reference

```
#include <boost/compute.hpp>
#include "triSYCL/detail/cache.hpp"
#include "triSYCL/detail/debug.hpp"
#include "triSYCL/detail/unimplemented.hpp"
#include "triSYCL/kernel/detail/kernel.hpp"
#include "triSYCL/queue/detail/queue.hpp"
```

Include dependency graph for opencl_kernel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::opencl_kernel

  An abstraction of the OpenCL kernel.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Macros

- #define TRISYCL_ParallelForKernel_RANGE(N)

  Launch an OpenCL kernel with a range<>

13.139.1 Macro Definition Documentation

13.139.1.1 TRISYCL_ParallelForKernel_RANGE

```
#define TRISYCL_ParallelForKernel_RANGE(
              N )
```

Value:

```
  void parallel_for(std::shared_ptr<detail::task> task,                \
                    std::shared_ptr<detail::queue> q,                  \
                    const range<N> &num_work_items)override {          \
    static_assert(sizeof(range<N>::value_type) == sizeof(size_t),      \
                  "num_work_items::value_type compatible with "        \
                  "Boost.Compute");                                    \
    q->get_boost_compute().enqueue_nd_range_kernel                     \
      (k,                                                              \
       static_cast<size_t>(N),                                         \
       NULL,                                                           \
       static_cast<const size_t*>(num_work_items.data()),              \
       NULL);                                                          \
    /* For now use a crude synchronization mechanism to map directly a  \
host task to an accelerator task */                                    \
    q->get_boost_compute().finish();                                   \
  };
```

Launch an OpenCL kernel with a range<>

Do not use a template since it does not work with virtual functions

Todo Think to a cleaner solution

Todo Remove either task or q

Definition at line 106 of file opencl_kernel.hpp.

13.140 opencl_kernel.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_KERNEL_DETAIL_OPENCL_KERNEL_HPP
00002 #define TRISYCL_SYCL_KERNEL_DETAIL_OPENCL_KERNEL_HPP
00003
00004 /** \file The OpenCL SYCL kernel
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #ifdef TRISYCL_OPENCL
00013 #include <boost/compute.hpp>
00014 #endif
00015
00016 #include "triSYCL/detail/cache.hpp"
00017 #include "triSYCL/detail/debug.hpp"
00018 #include "triSYCL/detail/unimplemented.hpp"
00019 //#include "triSYCL/info/kernel.hpp"
00020 #include "triSYCL/kernel/detail/kernel.hpp"
00021 #include "triSYCL/queue/detail/queue.hpp"
00022
00023
00024 namespace trisycl::detail {
00025
00026 /// An abstraction of the OpenCL kernel
00027 class opencl_kernel :  public detail::kernel,
00028                        detail::debug<opencl_kernel> {
00029
00030 /// Use the Boost Compute abstraction of the OpenCL kernel
00031   boost::compute::kernel k;
00032
00033 /** A cache to always return the same alive kernel for a given
00034 OpenCL kernel
```

```
00035
00036 C++11 guaranties the static construction is thread-safe
00037 */
00038   static detail::cache<cl_kernel, detail::opencl_kernel> cache;
00039
00040   opencl_kernel(const boost::compute::kernel &k) :  k { k } {}
00041
00042  public:
00043
00044 ///// Get a singleton instance of the opencl_device
00045   static std::shared_ptr<opencl_kernel>
00046   instance(const boost::compute::kernel &k) {
00047     return cache.get_or_register(k.get(),
00048                                  [&] { return new opencl_kernel { k }; });
00049   }
00050
00051 /** Return the underlying OpenCL object
00052
00053 \todo Improve the spec to deprecate C OpenCL host API and move
00054 to C++ instead to avoid this ugly ownership management
00055 */
00056   cl_kernel get()const override {
00057 /// \todo Test error and throw.  Externalize this feature in Boost.Compute?
00058     clRetainKernel(k);
00059     return k.get();
00060   }
00061
00062
00063 /** Return the Boost.Compute OpenCL kernel object for this kernel
00064
00065 This is an extension.
00066 */
00067   boost::compute::kernel &get_boost_compute()override {
00068     return k;
00069   }
00070
00071
00072   //context get_context() const override
00073
00074   //program get_program() const override
00075
00076 #if 0
00077   template <info::kernel param>
00078   typename info::param_traits<info::kernel, param>::type
00079   get_info()const {
00080     TRISYCL_UNIMPL;
00081   }
00082 #endif
00083
00084
00085 /** Launch a single task of the OpenCL kernel
00086
00087 \todo Remove either task or q
00088 */
00089   void single_task(std::shared_ptr<detail::task> task,
00090                    std::shared_ptr<detail::queue> q)override {
00091     q->get_boost_compute().enqueue_task(k);
00092     /* For now use a crude synchronization mechanism to map directly a
00093 host task to an accelerator task */
00094     q->get_boost_compute().finish();
00095   }
00096
00097
00098 /** Launch an OpenCL kernel with a range<>
00099
00100 Do not use a template since it does not work with virtual functions
00101
00102 \todo Think to a cleaner solution
00103
00104 \todo Remove either task or q
00105 */
00106 #define TRISYCL_ParallelForKernel_RANGE(N)                              \
00107 void parallel_for(std::shared_ptr<detail::task> task,                 \
00108 std::shared_ptr<detail::queue> q,                    \
00109 const range<N> &num_work_items) override {           \
00110 static_assert(sizeof(range<N>::value_type) == sizeof(size_t),       \
00111 "num_work_items::value_type compatible with "          \
00112                   "Boost.Compute");                                     \
00113     q->get_boost_compute().enqueue_nd_range_kernel                      \
00114       (k,                                                               \
00115        static_cast<size_t>(N),                                          \
00116        NULL,                                                            \
00117        static_cast<const size_t*>(num_work_items.data()),               \
00118        NULL);                                                           \
00119     /* For now use a crude synchronization mechanism to map directly a  \
00120 host task to an accelerator task */\
00121 q->get_boost_compute().finish();                                        \
```

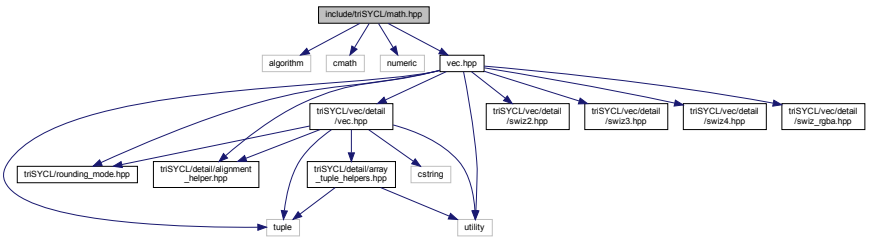
```
00122 };
00123
00124   TRISYCL_ParallelForKernel_RANGE(1)
00125   TRISYCL_ParallelForKernel_RANGE(2)
00126   TRISYCL_ParallelForKernel_RANGE(3)
00127 #undef TRISYCL_ParallelForKernel_RANGE
00128
00129
00130 /// Unregister from the cache on destruction
00131   ~opencl_kernel()override {
00132     cache.remove(k.get());
00133   }
00134
00135 };
00136
00137 /* Allocate the cache here but since this is a pure-header library,
00138 use a weak symbol so that only one remains when SYCL headers are
00139 used in different compilation units of a program
00140 */
00141 TRISYCL_WEAK_ATTRIB_PREFIX
00142 detail::cache<cl_kernel, detail::opencl_kernel> opencl_kernel::cache
00143 TRISYCL_WEAK_ATTRIB_SUFFIX;
00144
00145 }
00146
00147 /*
00148 # Some Emacs stuff:
00149 ### Local Variables:
00150 ### ispell-local-dictionary:  "american"
00151 ### eval:  (flyspell-prog-mode)
00152 ### End:
00153 */
00154
00155 #endif // TRISYCL_SYCL_KERNEL_DETAIL_OPENCL_KERNEL_HPP
```

13.141 include/triSYCL/math.hpp File Reference

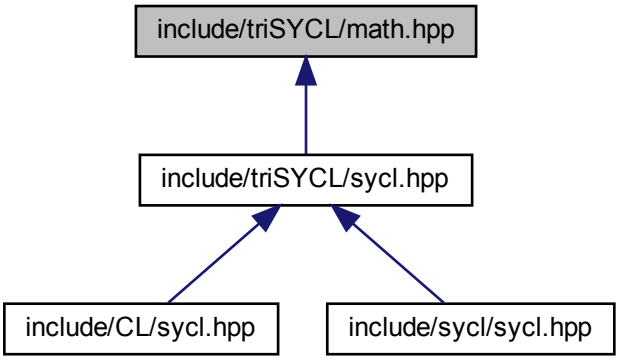
Implement a wrapper around OpenCL math operations Joan.Thibault AT ens-rennes POINT fr This file is distributed under the University of Illinois Open Source License.

```
#include <algorithm>
#include <cmath>
#include <numeric>
#include "vec.hpp"
```

Include dependency graph for math.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::native

Macros

- #define TRISYCL_MATH_WRAP(FUN)

  Declare a FUN unary function for scalar and vector types.
- #define TRISYCL_MATH_WRAP2(FUN)

  Declare a FUN binary function for scalar and vector types.
- #define TRISYCL_MATH_WRAP2s(FUN)

  Declare a FUN binary function for scalar and vector types, with 2 different types.
- #define TRISYCL_MATH_WRAP3(FUN)

  Declare a FUN ternary function for scalar and vector types.
- #define TRISYCL_MATH_WRAP3s(FUN)

  Declare a FUN ternary function for scalar and vector types, with 2 different types, the 2 first arguments have the same type.
- #define TRISYCL_MATH_WRAP3ss(FUN)

  Declare a FUN ternary function for scalar and vector types, with 2 different types, the 2 last arguments have the same type.

Functions

- trisycl::TRISYCL_MATH_WRAP (abs) TRISYCL_MATH_WRAP(atan) TRISYCL_MATH_WRAP3ss(clamp) TRISYCL_MATH_WRAP2s(fmax) template< typename T
- int size auto trisycl::fmax (const vec< T, size > &x, const vec< T, size > &y)
- trisycl::TRISYCL_MATH_WRAP2s (fmin) template< typename T
- int size auto trisycl::fmin (const vec< T, size > &x, const vec< T, size > &y)

- trisycl::TRISYCL_MATH_WRAP2s (frexp) template< typename T > T max(const T &x
- template<typename T , int size>
  auto trisycl::max (const vec< T, size > &x, const vec< T, size > &y)
- template<typename T >
  T trisycl::min (const T &x, const T &y, const T &z)
- template<typename T , int size>
  auto trisycl::min (const vec< T, size > &x, const vec< T, size > &y)
- trisycl::TRISYCL_MATH_WRAP2s (modf) TRISYCL_MATH_WRAP3s(remquo) TRISYCL_MATH_WRAP2(rotate)
  template< typename T
- int size auto trisycl::cross (const vec< T, size > &x, const vec< T, size > &y)
- template<typename T , int size>
  auto trisycl::length (const vec< T, size > &x)
- template<typename T , int size>
  auto trisycl::dot (const vec< T, size > &x, const vec< T, size > &y)
- template<typename T , int size>
  auto trisycl::normalize (const vec< T, size > &x)
- template<typename T >
  T trisycl::sign (const T &val)
- template<typename T , int size>
  auto trisycl::sign (const vec< T, size > &x)

Variables

- const T & trisycl::y
- const T const T & trisycl::z

13.141.1 Detailed Description

Implement a wrapper around OpenCL math operations Joan.Thibault AT ens-rennes POINT fr This file is distributed under the University of Illinois Open Source License.

See LICENSE.TXT for details.

Definition in file math.hpp.

13.141.2 Macro Definition Documentation

13.141.2.1 TRISYCL_MATH_WRAP

```
#define TRISYCL_MATH_WRAP(
              FUN )
```

Value:

```
  template<typename T>                      \
  /* Just forward the scalar case to the C++ math library */           \
  T FUN(const T& x) {                                                  \
    return std::FUN(x);                                                \
  }                                                                    \
  /* Declare the vector version as applying the scalar version on each  \
element */                                                              \
  template <typename T, int size>                                      \
  auto FUN(const vec<T, size>& x) {                                    \
    return x.map(FUN<T>);                                              \
  }
```

Declare a FUN unary function for scalar and vector types.

Definition at line 30 of file math.hpp.

13.141.2.2 TRISYCL_MATH_WRAP2

```
#define TRISYCL_MATH_WRAP2(
              FUN )
```

Value:

```
  template<typename T>                    \
  /* Just forward the scalar case to the C++ math library */           \
  T FUN(const T& x, const T& y) {                                       \
    return std::FUN(x, y);                                              \
  }                                                                     \
  /* Declare the vector version as applying the scalar version on each  \
element */                                                             \
  template <typename T, int size>                                       \
  auto FUN(const vec<T, size>& x,                                       \
           const vec<T, size>& y) {                                     \
    return x.zip(y, FUN<T,T>);                                          \
  }
```

Declare a FUN binary function for scalar and vector types.

Definition at line 43 of file math.hpp.

13.141.2.3 TRISYCL_MATH_WRAP2s

```
#define TRISYCL_MATH_WRAP2s(
              FUN )
```

Value:

```
  template<typename T, typename U>        \
  /* Just forward the scalar case to the C++ math library */           \
  T FUN(const T& x, const U& y) {                                       \
    return std::FUN(x, y);                                              \
  }                                                                     \
  /* Declare the vector version as applying the scalar version on each  \
element */                                                             \
  template <typename T, typename U, int size>                           \
  auto FUN(const vec<T, size>& x,                                       \
           const U& y) {                                                \
    return x.map([&] (auto e) { return FUN(e, y); });                   \
  }
```

Declare a FUN binary function for scalar and vector types, with 2 different types.

Definition at line 58 of file math.hpp.

13.141.2.4 TRISYCL_MATH_WRAP3

```
#define TRISYCL_MATH_WRAP3(
              FUN )
```

Value:

```
  template<typename T>                    \
  /* Just forward the scalar case to the C++ math library */           \
  T FUN(const T& x, const T& y, const T& z) {                           \
    return std::FUN(x, y, z);                                           \
  }                                                                     \
  /* Declare the vector version as applying the scalar version on each  \
element */                                                             \
  template <typename T, int size>                                       \
  auto FUN(const vec<T, size>& x,                                       \
           const vec<T, size>& y,                                       \
           const vec<T, size>& z) {                                     \
  return x.zip(y, z, FUN<T,T,T>);                                       \
}
```

Declare a FUN ternary function for scalar and vector types.

Definition at line 72 of file math.hpp.

13.141.2.5 TRISYCL_MATH_WRAP3s

```
#define TRISYCL_MATH_WRAP3s(
               FUN )
```

Value:

```
  template<typename T, typename U>       \
  /* Just forward the scalar case to the C++ math library */           \
  T FUN(const T& x, const T& y, const U& z) {                          \
    return std::FUN(x, y, z);                                          \
  }                                                                    \
  /* Declare the vector version as applying the scalar version on each  \
element */                                                             \
  template <typename T, typename U, int size>                          \
  auto FUN(const vec<T, size>& x,                                      \
           const vec<T, size>& y,                                      \
           const U& z) {                                               \
    return x.zip(y, [&] (auto ex, auto ey) { return FUN(ex, ey, z); }); \
}
```

Declare a FUN ternary function for scalar and vector types, with 2 different types, the 2 first arguments have the same type.

Definition at line 88 of file math.hpp.

13.141.2.6 TRISYCL_MATH_WRAP3ss

```
#define TRISYCL_MATH_WRAP3ss(
               FUN )
```

Value:

```
  template<typename T, typename U>      \
  /* Just forward the scalar case to the C++ math library */           \
  T FUN(const T& x, const U& y, const U& z) {                          \
    return std::FUN(x, y, z);                                          \
  }                                                                    \
  /* Declare the vector version as applying the scalar version on each  \
element */                                                             \
  template <typename T, typename U, int size>                          \
  auto FUN(const vec<T, size>& x,                                      \
           const U& y,                                                 \
           const U& z) {                                               \
    return x.map([&] (auto e) { return FUN(e, y, z); });               \
}
```

Declare a FUN ternary function for scalar and vector types, with 2 different types, the 2 last arguments have the same type.

Definition at line 104 of file math.hpp.

13.142 math.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_MATH_HPP
00002 #define TRISYCL_SYCL_MATH_HPP
00003
00004 /** \file
00005 Implement a wrapper around OpenCL math operations
00006 Joan.Thibault AT ens-rennes POINT fr
00007 This file is distributed under the University of Illinois Open Source
00008 License.  See LICENSE.TXT for details.
00009 */
00010
00011 #include <algorithm>
```

```
00012 #include <cmath>
00013 #include <numeric>
00014
00015 #include "vec.hpp"
00016
00017 // Include order and configure insensitive treating of unwanted macros
00018 #ifdef _MSC_VER
00019 #ifdef min
00020 #undef min
00021 #endif
00022 #ifdef max
00023 #undef max
00024 #endif
00025 #endif
00026
00027 namespace trisycl {
00028
00029 /// Declare a FUN unary function for scalar and vector types
00030 #define TRISYCL_MATH_WRAP(FUN) template<typename T>                    \
00031 /* Just forward the scalar case to the C++ math library */\
00032 T FUN(const T& x) {                                                   \
00033 return std::FUN(x);                                                \
00034 }                                                                     \
00035 /* Declare the vector version as applying the scalar version on each  \
00036 element */\
00037 template <typename T, int size>                                       \
00038 auto FUN(const vec<T, size>& x) {                                     \
00039 return x.map(FUN<T>);                                              \
00040 }
00041
00042
00043 /// Declare a FUN binary function for scalar and vector types
00044 #define TRISYCL_MATH_WRAP2(FUN) template<typename T>                   \
00045 /* Just forward the scalar case to the C++ math library */\
00046 T FUN(const T& x, const T& y) {                                       \
00047 return std::FUN(x, y);                                             \
00048 }                                                                     \
00049 /* Declare the vector version as applying the scalar version on each  \
00050 element */\
00051 template <typename T, int size>                                       \
00052 auto FUN(const vec<T, size>& x,                                       \
00053 const vec<T, size>& y) {                                          \
00054 return x.zip(y, FUN<T,T>);                                         \
00055 }
00056
00057
00058 /** Declare a FUN binary function for scalar and vector types, with 2
00059 different types */
00060 #define TRISYCL_MATH_WRAP2s(FUN) template<typename T, typename U>      \
00061 /* Just forward the scalar case to the C++ math library */\
00062 T FUN(const T& x, const U& y) {                                       \
00063 return std::FUN(x, y);                                             \
00064 }                                                                     \
00065 /* Declare the vector version as applying the scalar version on each  \
00066 element */\
00067 template <typename T, typename U, int size>                           \
00068 auto FUN(const vec<T, size>& x,                                       \
00069 const U& y) {                                                     \
00070 return x.map([&] (auto e) { return FUN(e, y); });                  \
00071 }
00072
00073
00074 /// Declare a FUN ternary function for scalar and vector types
00075 #define TRISYCL_MATH_WRAP3(FUN) template<typename T>                   \
00076 /* Just forward the scalar case to the C++ math library */\
00077 T FUN(const T& x, const T& y, const T& z) {                           \
00078 return std::FUN(x, y, z);                                          \
00079 }                                                                     \
00080 /* Declare the vector version as applying the scalar version on each  \
00081 element */\
00082 template <typename T, int size>                                       \
00083 auto FUN(const vec<T, size>& x,                                       \
00084 const vec<T, size>& y,                                            \
00085 const vec<T, size>& z) {                                          \
00086 return x.zip(y, z, FUN<T,T,T>);                                    \
00087 }
00088
00089
00090 /** Declare a FUN ternary function for scalar and vector types, with 2
00091 different types, the 2 first arguments have the same type */
00092 #define TRISYCL_MATH_WRAP3s(FUN) template<typename T, typename U>      \
00093 /* Just forward the scalar case to the C++ math library */\
00094 T FUN(const T& x, const T& y, const U& z) {                           \
00095 return std::FUN(x, y, z);                                          \
00096 }                                                                     \
00097 /* Declare the vector version as applying the scalar version on each  \
00098 element */\
```

```
00099 template <typename T, typename U, int size>                          \
00100 auto FUN(const vec<T, size>& x,                                       \
00101 const vec<T, size>& y,                                               \
00102 const U& z) {                                                        \
00103 return x.zip(y, [&] (auto ex, auto ey) { return FUN(ex, ey, z); }); \
00104 }
00105
00106
00107 /** Declare a FUN ternary function for scalar and vector types, with 2
00108 different types, the 2 last arguments have the same type */
00109 #define TRISYCL_MATH_WRAP3ss(FUN) template<typename T, typename U>      \
00110 /* Just forward the scalar case to the C++ math library */\
00111 T FUN(const T& x, const U& y, const U& z) {                            \
00112 return std::FUN(x, y, z);                                             \
00113 }                                                                     \
00114 /* Declare the vector version as applying the scalar version on each  \
00115 element */\
00116 template <typename T, typename U, int size>                            \
00117 auto FUN(const vec<T, size>& x,                                       \
00118 const U& y,                                                           \
00119 const U& z) {                                                        \
00120 return x.map([&] (auto e) { return FUN(e, y, z); });                  \
00121 }
00122
00123
00124 TRISYCL_MATH_WRAP(abs)//I
00125 //*TRISYCL_MATH_WRAP2(abs_diff)//I
00126 //*TRISYCL_MATH_WRAP2(add_sat)//I
00127 TRISYCL_MATH_WRAP(acos)
00128 TRISYCL_MATH_WRAP(acosh)
00129 //*TRISYCL_MATH_WRAP(acospi)
00130 TRISYCL_MATH_WRAP(asin)
00131 TRISYCL_MATH_WRAP(asinh)
00132 //*TRISYCL_MATH_WRAP(asinpi)
00133 TRISYCL_MATH_WRAP(atan) // atan(y/x)
00134 TRISYCL_MATH_WRAP2(atan2)
00135 TRISYCL_MATH_WRAP(atanh)
00136 //*TRISYCL_MATH_WRAP(atanpi)
00137 //*TRISYCL_MATH_WRAP2(atan2pi)
00138 TRISYCL_MATH_WRAP(cbrt)
00139 TRISYCL_MATH_WRAP(ceil)
00140 //geninteger clamp(geninteger, sgeninteger, sgeninteger)
00141 TRISYCL_MATH_WRAP3ss(clamp)//I
00142
00143 //*TRISYCL_MATH_WRAP(clz)
00144 TRISYCL_MATH_WRAP2(copysign)
00145 TRISYCL_MATH_WRAP(cos)
00146 TRISYCL_MATH_WRAP(cosh)
00147 //*TRISYCL_MATH_WRAP(cospi)
00148 TRISYCL_MATH_WRAP(erfc)
00149 TRISYCL_MATH_WRAP(erf)
00150 TRISYCL_MATH_WRAP(exp)
00151 TRISYCL_MATH_WRAP(exp2)
00152 //*TRISYCL_MATH_WRAP(exp10)
00153 TRISYCL_MATH_WRAP(expm1)
00154 TRISYCL_MATH_WRAP(fabs)
00155 TRISYCL_MATH_WRAP2(fdim)
00156 // Round to integral value using the round to
00157 // negative infinity rounding mode.
00158 TRISYCL_MATH_WRAP(floor)
00159 TRISYCL_MATH_WRAP3(fma)
00160 /* genfloat fmax ( genfloat x, genfloat y)
00161 * genfloat fmax ( genfloat x, sgenfloat y)
00162 */
00163
00164 TRISYCL_MATH_WRAP2s(fmax)
00165 // Returns y if x < y, otherwise it returns x.
00166 // If one argument is a NaN, fmax() returns
00167 // the other argument.  If both arguments are
00168 // NaNs, fmax() returns a NaN
00169 template <typename T, int size>
00170 auto fmax(const vec<T, size>& x,
00171           const vec<T, size>& y) {
00172   return x.zip(y, fmax<T,T>);
00173 }
00174
00175 TRISYCL_MATH_WRAP2s(fmin)
00176
00177 // Returns y if y < x, otherwise it returns x.
00178 // genfloat fmin ( genfloat x, sgenfloat y )
00179 // If one argument is a NaN, fmin() returns
00180 // the other argument.  If both arguments are
00181 // NaNs, fmin() returns a NaN.
00182 //
00183 // \todo Perhaps worth adding an isNaN check as I'm not sure if the < operator
00184 //       conforms to what the function requires
00185 template <typename T, int size>
```

```
00186 auto fmin(const vec<T, size>& x,
00187           const vec<T, size>& y) {
00188   return x.zip(y, fmin<T,T>);
00189 }
00190
00191 TRISYCL_MATH_WRAP2(fmod)
00192 //*TRISYCL_MATH_WRAP2s(fract)
00193 TRISYCL_MATH_WRAP2s(frexp)
00194 //*TRISYCL_MATH_WRAP(hadd)
00195 TRISYCL_MATH_WRAP2(hypot)
00196 //log
00197 //ilogb
00198 TRISYCL_MATH_WRAP(isnan)
00199 //ldexp
00200 TRISYCL_MATH_WRAP(lgamma)
00201 //*TRISYCL_MATH_WRAP2s(lgamma_r)
00202 TRISYCL_MATH_WRAP(log)
00203 TRISYCL_MATH_WRAP(log2)
00204 TRISYCL_MATH_WRAP(log10)
00205 TRISYCL_MATH_WRAP(log1p)
00206 TRISYCL_MATH_WRAP(logb)
00207 //*TRISYCL_MATH_WRAP3(mad)
00208 //*TRISYCL_MATH_WRAP3(mad_hi)//I
00209 //*TRISYCL_MATH_WRAP3(mad_sat)
00210 //
00211 //TRISYCL_MATH_WRAP3s(max) //I
00212 template<typename T>
00213 T max(const T& x, const T& y, const T& z) {
00214   return std::max(x, std::max(y, z));
00215 }
00216
00217 /* geninteger max (geninteger, geninteger)
00218 * geninteger max (geninteger, sgeninteger)
00219 */
00220 // Just import std::max from <algorithm>
00221 using std::max;
00222
00223 // seems to essentially be the same as fmax but cover different types, even the
00224 // first type definition matches fmax's first?  Is there a need for two
00225 // similar definitions of max?  same goes for fmin...
00226 // genfloat max (genfloat x, genfloat y)
00227 // genfloatf max (genfloatf x, float y)
00228 // genfloatd max (genfloatd x, double y)
00229 // geninteger max (geninteger x, geninteger y)
00230 // geninteger max (geninteger x, sgeninteger y)
00231 template <typename T, int size>
00232 auto max(const vec<T, size>& x,
00233          const vec<T, size>& y) {
00234   return fmax(x,y);
00235 }
00236
00237 //*TRISYCL_MATH_WRAP2(maxmag)
00238 //
00239 //TRISYCL_MATH_WRAP3s(min) //I
00240 template<typename T>
00241 T min(const T& x, const T& y, const T& z) {
00242   return std::min(x, std::min(y, z));
00243 }
00244
00245 /* geninteger min (geninteger, geninteger)
00246 * geninteger min (geninteger, sgeninteger)
00247 */
00248   // Just import std::min from <algorithm>
00249 using std::min;
00250
00251 template <typename T, int size>
00252 auto min(const vec<T, size>& x,
00253          const vec<T, size>& y) {
00254   return fmin(x,y);
00255 }
00256
00257 //*TRISYCL_MATH_WRAP2(minmag)
00258 TRISYCL_MATH_WRAP2s(modf)
00259 //*TRISYCL_MATH_WRAP2(mul_hi)//I
00260 //nan
00261 TRISYCL_MATH_WRAP2(pow)
00262 //*TRISYCL_MATH_WRAP2s(posn)
00263 //*TRISYCL_MATH_WRAP2(powr)
00264 TRISYCL_MATH_WRAP2(remainder)
00265 TRISYCL_MATH_WRAP3s(remquo)
00266 //*TRISYCL_MATH_WRAP(rhadd)//I
00267 TRISYCL_MATH_WRAP(rint)
00268 //*TRISYCL_MATH_WRAP3s(rootn)
00269 TRISYCL_MATH_WRAP2(rotate)//I
00270 TRISYCL_MATH_WRAP(round)
00271 //*TRISYCL_MATH_WRAP(rsqrt)
00272 TRISYCL_MATH_WRAP(sin)
```

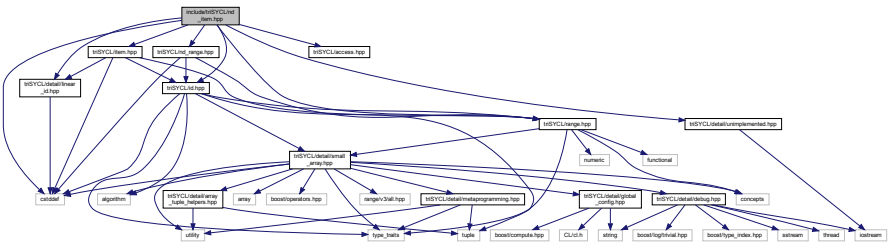
```
00273 //*TRISYCL_MATH_WRAP2s(sincos)
00274 TRISYCL_MATH_WRAP(sinh)
00275 //*TRISYCL_MATH_WRAP(sinpi)
00276 TRISYCL_MATH_WRAP(sqrt)
00277 //*TRISYCL_MATH_WRAP2(sub_sat)
00278 TRISYCL_MATH_WRAP(tan)
00279 TRISYCL_MATH_WRAP(tanh)
00280 //*TRISYCL_MATH_WRAP(tanpi)
00281 TRISYCL_MATH_WRAP(tgamma)
00282 TRISYCL_MATH_WRAP(trunc)
00283 /* Integer concatenation
00284 * shortn upsample (charn hi, ucharn lo)
00285 * ushortn upsample (ucharn hi, ucharn lo)
00286 * intn upsample (shortn hi, ushortn lo)
00287 * uintn upsample (ushortn hi, ushortn lo)
00288 * longlongn upsample(intn hi, uintn lo)
00289 * ulonglongn upsample(uintn hi, uintn l)
00290 */
00291 //*TRISYCL_MATH_WRAP(popcount)//I
00292 //*TRISYCL_MATH_WRAP3(mad24)
00293 //*TRISYCL_MATH_WRAP3(mul24)
00294
00295 // Note:  A lot of these vec math functions could possibly be forwarded onto the
00296 // OpenCL implementations when compiling for OpenCL, perhaps the above OpenCL
00297 // forwarding will already allow that partially?
00298
00299 // Returns the cross product of x.xyz and y.xyz.  The w component of float4
00300 // result returned will be 0.0.
00301 template <typename T, int size>
00302 auto cross(const vec<T, size>& x, const vec<T, size>& y) {
00303   static_assert((size == 4 || size == 3),
00304                 "vec type should be 3 or 4 dimensions");
00305   if constexpr (size > 3) {
00306     return vec<T, size> {
00307             (x[1] * y[2]) - (x[2] * y[1]),
00308             (x[2] * y[0]) - (x[0] * y[2]),
00309             (x[0] * y[1]) - (x[1] * y[0]),
00310             0.0};
00311   } else {
00312     return vec<T, size> {
00313             (x[1] * y[2]) - (x[2] * y[1]),
00314             (x[2] * y[0]) - (x[0] * y[2]),
00315             (x[0] * y[1]) - (x[1] * y[0])};
00316   }
00317 }
00318
00319 // Return the length of vector x, i.e., sqrt(x.x^2 + x.y^2 + ...)
00320 template <typename T, int size>
00321 auto length(const vec<T, size>& x) {
00322   auto temp = x * x;
00323   return sqrt(std::accumulate(temp.begin(), temp.end(), T{0}));
00324 }
00325
00326 // Compute dot product.
00327 template <typename T, int size>
00328 auto dot(const vec<T, size>& x, const vec<T, size>& y) {
00329   return std::inner_product(x.begin(), x.end(), y.begin(), T{0});
00330 }
00331
00332 // Returns a vector in the same direction as x but with a length of 1.
00333 template <typename T, int size>
00334 auto normalize(const vec<T, size>& x) {
00335   return x / length(x);
00336 }
00337
00338 // returns     x if x is +/- 0,
00339 // otherwise  +0 if x is NaN,
00340 // otherwise   1 if x is > 0,
00341 // otherwise  -1
00342 template<typename T>
00343 T sign(const T & val) {
00344   if (val == 0) return val;
00345   if (isnan(val)) return 0;
00346   return (val > 0) ?  1 :  -1;
00347 }
00348
00349 template <typename T, int size>
00350 auto sign(const vec<T, size>& x) {
00351   return x.map(sign<T>);
00352 }
00353 //
00354 namespace native {
00355 TRISYCL_MATH_WRAP(cos)
00356 //*TRISYCL_MATH_WRAP2(divide)
00357 TRISYCL_MATH_WRAP(exp)
00358 TRISYCL_MATH_WRAP(exp2)
00359 //*TRISYCL_MATH_WRAP(exp10)
```

```
00360 TRISYCL_MATH_WRAP(log)
00361 TRISYCL_MATH_WRAP(log2)
00362 TRISYCL_MATH_WRAP(log10)
00363 //*TRISYCL_MATH_WRAP(powr)
00364 //*TRISYCL_MATH_WRAP(recip)
00365 //*TRISYCL_MATH_WRAP(rsqrt)
00366 TRISYCL_MATH_WRAP(sin)
00367 TRISYCL_MATH_WRAP(sqrt)
00368 TRISYCL_MATH_WRAP(tan)
00369 }
00370 #undef TRISYCL_MATH_WRAP
00371 #undef TRISYCL_MATH_WRAP2
00372 #undef TRISYCL_MATH_WRAP2s
00373 #undef TRISYCL_MATH_WRAP3
00374 #undef TRISYCL_MATH_WRAP3s
00375 #undef TRISYCL_MATH_WRAP3ss
00376
00377 }
00378
00379 #endif
```

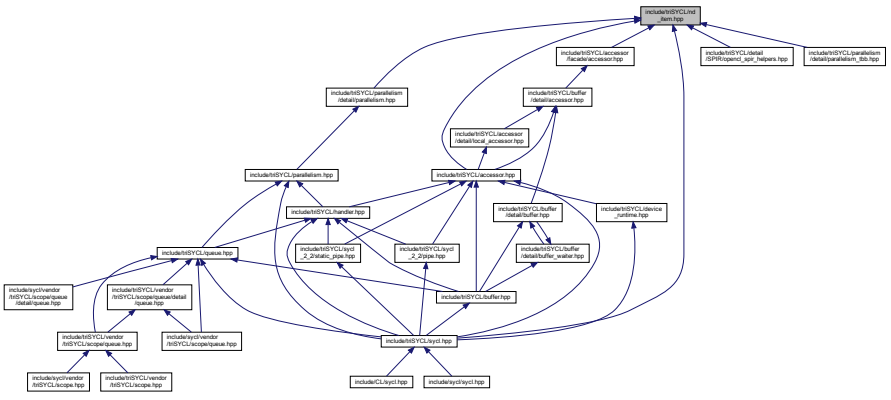
13.143 include/triSYCL/nd_item.hpp File Reference

```
#include <cstddef>
#include "triSYCL/access.hpp"
#include "triSYCL/detail/linear_id.hpp"
#include "triSYCL/detail/unimplemented.hpp"
#include "triSYCL/id.hpp"
#include "triSYCL/item.hpp"
#include "triSYCL/nd_range.hpp"
#include "triSYCL/range.hpp"
```

Include dependency graph for nd_item.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::nd_item< Dimensions >

  A SYCL nd_item stores information on a work-item within a work-group, with some more context such as the definition ranges. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

13.144 nd_item.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_ND_ITEM_HPP
00002 #define TRISYCL_SYCL_ND_ITEM_HPP
00003
00004 /** \file The OpenCL SYCL nd_item<>
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <cstddef>
00013
00014 #include "triSYCL/access.hpp"
00015 #include "triSYCL/detail/linear_id.hpp"
00016 #include "triSYCL/detail/unimplemented.hpp"
00017 #include "triSYCL/id.hpp"
00018 #include "triSYCL/item.hpp"
00019 #include "triSYCL/nd_range.hpp"
00020 #include "triSYCL/range.hpp"
00021
00022 namespace trisycl {
00023
00024 /** \addtogroup parallelism Expressing parallelism through kernels
00025 @{
00026 */
00027 /** A SYCL nd_item stores information on a work-item within a work-group,
00028 with some more context such as the definition ranges.
00029 */
00030 template <int Dimensions = 1>
00031 struct nd_item :  boost::equality_comparable<nd_item<Dimensions>> {
00032 /// The number of dimensions of the nd_item
00033   static auto constexpr rank() { return Dimensions; }
00034
00035 private:
00036
00037   id<Dimensions> global_index;
00038   /* This is a cached value since it can be computed from global_index and
00039 ND_range */
00040   id<Dimensions> local_index;
00041   nd_range<Dimensions> ND_range;
00042
00043 public:
00044
00045 /** Create an empty nd_item<> from an nd_range<>
00046
00047 \todo This is for the triSYCL implementation which is expected to
00048 call set_global() and set_local() later.  This should be hidden to
00049 the user.
00050 */
00051   nd_item(nd_range<Dimensions> ndr) :  ND_range { ndr } {}
00052
00053
00054 /** Create a full nd_item
00055
00056 \todo This is for validation purpose.  Hide this to the programmer
00057 somehow
00058 */
00059   nd_item(id<Dimensions> global_index,
00060           nd_range<Dimensions> ndr) :
```

```
00061     global_index { global_index },
00062     // Compute the local index using the offset and the group size
00063     local_index { (global_index - ndr.get_offset())%id<Dimensions> {
00064         ndr.get_local_range() } },
00065     ND_range { ndr }
00066   {}
00067
00068
00069 /** To be able to copy and assign nd_item, use default constructors too
00070
00071 \todo Make most of them protected, reserved to implementation
00072 */
00073   nd_item() = default;
00074
00075
00076 /** Return the constituent global id representing the work-item's
00077 position in the global iteration space
00078 */
00079   id<Dimensions> get_global_id()const { return global_index; }
00080
00081
00082 /** Return the constituent element of the global id representing the
00083 work-item's position in the global iteration space in the given
00084 dimension
00085 */
00086   size_t get_global_id(int dimension)const {
00087     return get_global_id()[dimension];
00088   }
00089
00090
00091 /** Return the flattened id of the current work-item after subtracting
00092 the offset
00093 */
00094   size_t get_global_linear_id()const {
00095     return detail::linear_id(get_global_range(),
00096                              get_global_id(),
00097                              get_offset());
00098   }
00099
00100
00101 /** Return the constituent local id representing the work-item's
00102 position within the current work-group
00103 */
00104   id<Dimensions> get_local_id()const { return local_index; }
00105
00106
00107 /** Return the constituent element of the local id representing the
00108 work-item's position within the current work-group in the given
00109 dimension
00110 */
00111   size_t get_local_id(int dimension)const {
00112     return get_local_id()[dimension];
00113   }
00114
00115
00116 /** Return the flattened id of the current work-item within the current
00117 work-group
00118 */
00119   size_t get_local_linear_id()const {
00120     return detail::linear_id(get_local_range(), get_local_id());
00121   }
00122
00123
00124 /** Return the constituent group representing the work-group's
00125 position within the overall nd_range
00126 */
00127   id<Dimensions> get_group()const {
00128     /* Convert get_local_range() to an id<> to remove ambiguity into using
00129 implicit conversion either from range<> to id<> or the opposite */
00130     return get_global_id()/id<Dimensions> { get_local_range() };
00131   }
00132
00133
00134 /** Return the constituent element of the group id representing the
00135 work-group;s position within the overall nd_range in the given
00136 dimension.
00137 */
00138   size_t get_group(int dimension)const {
00139     return get_group()[dimension];
00140   }
00141
00142
00143 /// Return the flattened id of the current work-group
00144   size_t get_group_linear_id()const {
00145     return detail::linear_id(get_group_range(), get_group());
00146   }
00147
```

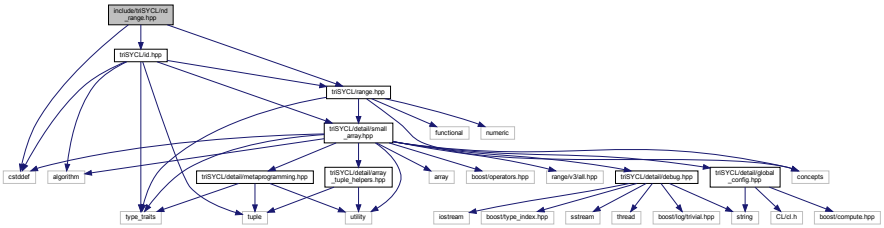
```
00148
00149 /// Return the number of groups in the nd_range
00150   id<Dimensions> get_group_range()const {
00151     return get_nd_range().get_group_range();
00152   }
00153
00154 /// Return the number of groups for dimension in the nd_range
00155   size_t get_group_range(int dimension)const {
00156      return get_group_range()[dimension];
00157   }
00158
00159
00160 /// Return a range<> representing the dimensions of the nd_range<>
00161   range<Dimensions> get_global_range()const {
00162     return get_nd_range().get_global_range();
00163   }
00164
00165
00166 /// Return a range<> representing the dimensions of the current work-group
00167   range<Dimensions> get_local_range()const {
00168     return get_nd_range().get_local_range();
00169   }
00170
00171
00172 /** Return an id<> representing the n-dimensional offset provided to the
00173 constructor of the nd_range<> and that is added by the runtime to the
00174 global-ID of each work-item
00175 */
00176   id<Dimensions> get_offset()const { return get_nd_range().get_offset(); }
00177
00178
00179 /// Return the nd_range<> of the current execution
00180   nd_range<Dimensions> get_nd_range()const { return ND_range; }
00181
00182
00183 /** Allows projection down to an item
00184
00185 \todo Add to the specification
00186 */
00187   item<Dimensions> get_item()const {
00188     return { get_global_range(), get_global_id(), get_offset() };
00189   }
00190
00191
00192 /** Execute a barrier with memory ordering on the local address space,
00193 global address space or both based on the value of flag
00194
00195 The current work-item will wait at the barrier until all work-items
00196 in the current work-group have reached the barrier.
00197
00198 In addition, the barrier performs a fence operation ensuring that all
00199 memory accesses in the specified address space issued before the
00200 barrier complete before those issued after the barrier
00201 */
00202   void barrier(access::fence_space flag =
00203                access::fence_space::global_and_local)const {
00204 #if defined(_OPENMP) && !defined(TRISYCL_NO_BARRIER)
00205     /* Use OpenMP barrier in the implementation with 1 OpenMP thread per
00206 work-item of the work-group */
00207 #pragma omp barrier
00208 #else
00209     // \todo To be implemented efficiently otherwise
00210     TRISYCL_UNIMPL;
00211 #endif
00212   }
00213
00214
00215   // For the triSYCL implementation, need to set the local index
00216   void set_local(id<Dimensions> Index) { local_index = Index; }
00217
00218
00219   // For the triSYCL implementation, need to set the global index
00220   void set_global(id<Dimensions> Index) { global_index = Index; }
00221
00222   // Comparison operators
00223   bool operator==(const nd_item<Dimensions> &nd_itemB)const {
00224     return (ND_range == nd_itemB.ND_range &&
00225         global_index == nd_itemB.global_index);
00226   }
00227 };
00228
00229 /// @} End the parallelism Doxygen group
00230
00231 }
00232
00233 /*
00234 # Some Emacs stuff:
```

```
00235 ### Local Variables:
00236 ### ispell-local-dictionary:  "american"
00237 ### eval:  (flyspell-prog-mode)
00238 ### End:
00239 */
00240
00241 #endif // TRISYCL_SYCL_ND_ITEM_HPP
```

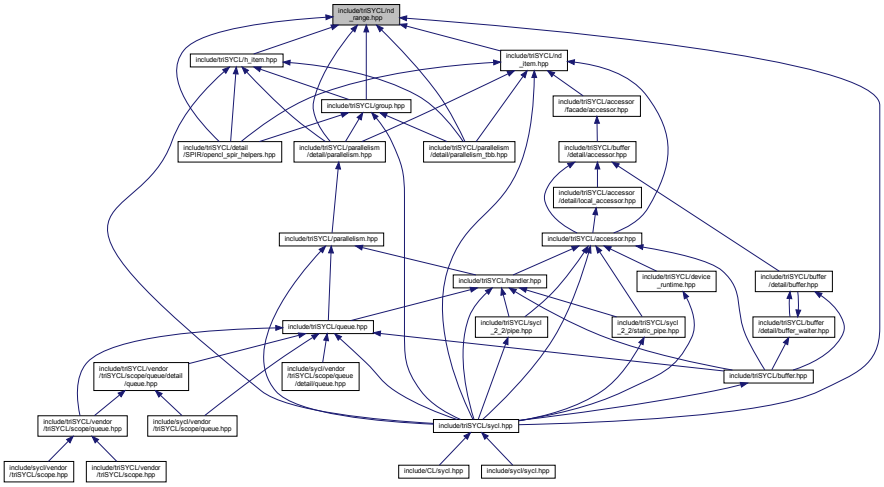
13.145 include/triSYCL/nd_range.hpp File Reference

```
#include <cstddef>
#include "triSYCL/id.hpp"
#include "triSYCL/range.hpp"
```

Include dependency graph for nd_range.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::nd_range< Dimensions >

  A ND-range, made by a global and local range, to specify work-group and work-item organization. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

13.146 nd_range.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_ND_RANGE_HPP
00002 #define TRISYCL_SYCL_ND_RANGE_HPP
00003
00004 /** \file The OpenCL SYCL nd_range<>
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <cstddef>
00013
00014 #include "triSYCL/id.hpp"
00015 #include "triSYCL/range.hpp"
00016
00017 namespace trisycl {
00018
00019 /** \addtogroup parallelism Expressing parallelism through kernels
00020 @{
00021 */
00022
00023 /** A ND-range, made by a global and local range, to specify work-group
00024 and work-item organization.
00025
00026 The local offset is used to translate the iteration space origin if
00027 needed.
00028
00029 \todo add copy constructors in the specification
00030 */
00031 template <int Dimensions = 1>
00032 struct nd_range :  boost::equality_comparable<nd_range<Dimensions>> {
00033 /// The number of dimensions of the nd_range
00034   static auto constexpr rank() { return Dimensions; }
00035
00036 private:
00037
00038   range<rank()> global_range;
00039   range<rank()> local_range;
00040   id<rank()> offset;
00041
00042 public:
00043
00044 /** Construct a ND-range with all the details available in OpenCL
00045
00046 By default use a zero offset, that is iterations start at 0
00047 */
00048   nd_range(range<Dimensions> global_size,
00049            range<Dimensions> local_size,
00050            id<Dimensions> offset = {}) :
00051     global_range { global_size }, local_range { local_size }, offset { offset }
00052   { }
00053
00054
00055 /// Get the global iteration space range
00056   range<Dimensions> get_global_range()const { return global_range; }
00057
00058
00059 /// Get the local part of the iteration space range
00060   range<Dimensions> get_local_range()const { return local_range; }
00061
00062
00063 /// Get the range of work-groups needed to run this ND-range
00064   auto get_group_range()const {
00065     /* This is basically global_range/local_range, round up to the
00066 next integer, in case the global range is not a multiple of the
00067 local range.  Note this is a motivating example to build a range
00068 from a scalar with a broadcasting constructor.  */
00069     return (global_range + local_range - range<Dimensions>{ 1 })/local_range;
00070   }
00071
00072
00073 /// \todo get_offset() is lacking in the specification
00074   id<Dimensions> get_offset()const { return offset; }
00075
00076
00077 /// Display the value for debugging and validation purpose
00078   void display()const {
00079     global_range.display();
00080     local_range.display();
00081     offset.display();
00082   }
```

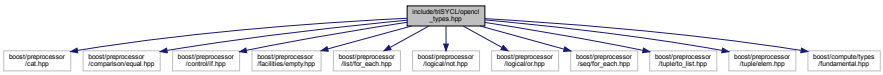
```
00083
00084 /// Comparison operators for nd_range.
00085   bool operator==(const nd_range &nd_rangeB)const {
00086     return (global_range == nd_rangeB.global_range &&
00087             offset == nd_rangeB.offset);
00088   }
00089 };
00090
00091 /// @} End the parallelism Doxygen group
00092
00093 }
00094
00095 /*
00096 # Some Emacs stuff:
00097 ### Local Variables:
00098 ### ispell-local-dictionary:  "american"
00099 ### eval:  (flyspell-prog-mode)
00100 ### End:
00101 */
00102
00103 #endif // TRISYCL_SYCL_ND_RANGE_HPP
```

13.147 include/triSYCL/opencl_types.hpp File Reference

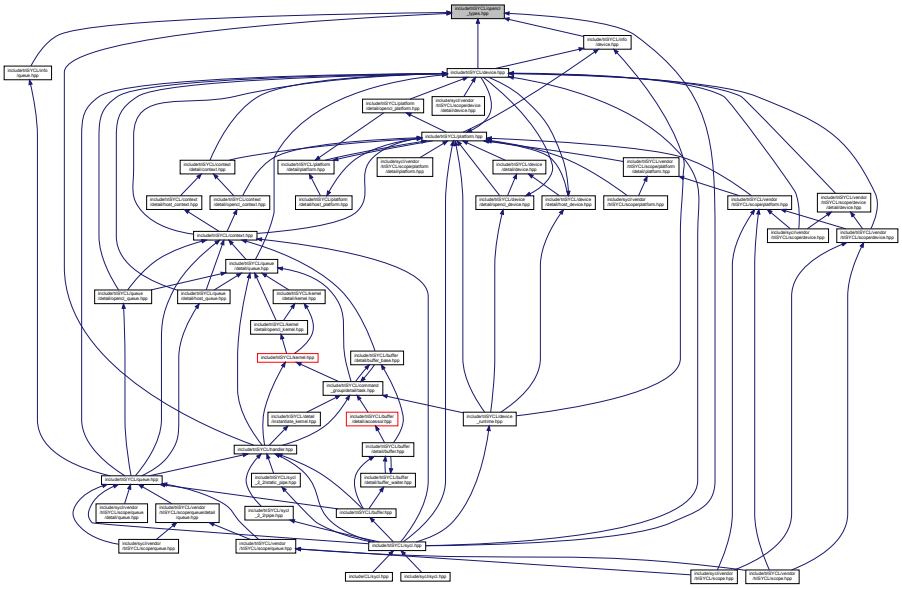
triSYCL wrapper for OpenCL types

```
#include <boost/preprocessor/cat.hpp>
#include <boost/preprocessor/comparison/equal.hpp>
#include <boost/preprocessor/control/if.hpp>
#include <boost/preprocessor/facilities/empty.hpp>
#include <boost/preprocessor/list/for_each.hpp>
#include <boost/preprocessor/logical/not.hpp>
#include <boost/preprocessor/logical/or.hpp>
#include <boost/preprocessor/seq/for_each.hpp>
#include <boost/preprocessor/tuple/to_list.hpp>
#include <boost/preprocessor/tuple/elem.hpp>
#include <boost/compute/types/fundamental.hpp>
```

Include dependency graph for opencl_types.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::is_wrapper< T >

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

Macros

- #define TRISYCL_SCALAR_TYPES
- #define TRISYCL_TYPE_NAME(T) BOOST_PP_TUPLE_ELEM(3, 0, T)
- #define TRISYCL_TYPE_CL_NAME(T) BOOST_PP_TUPLE_ELEM(3, 1, T)
- #define TRISYCL_TYPE_ACTUAL_NAME(T) BOOST_PP_TUPLE_ELEM(3, 2, T)
- #define TRISYCL_SIZED_NAME(T, size)
- #define TRISYCL_IS_WRAPPER_TRAIT(type) template <> struct is_wrapper<type> : std::true_type {};
- #define TRISYCL_WRAPPER_CLASS_2(cl_type, boost_name, scalar_type)
- #define TRISYCL_WRAPPER_CLASS_3(cl_type, boost_name, scalar_type)
- #define TRISYCL_WRAPPER_CLASS_4(cl_type, boost_name, scalar_type)
- #define TRISYCL_BOOST_COMPUTE_NAME(scalar, size) TRISYCL_SIZED_NAME(boost::compute::TRISYCL_TYPE_NAM size)
- #define TRISYCL_TYPEDEF_TYPE(cl_type, boost_name, scalar_type) using cl_type = boost_name##_;
- #define TRISYCL_H_DEFINE_TYPE(cl_type, boost_name, scalar_type, i)
- #define TRISYCL_DEFINE_TYPES(scalar, i)
- #define TRISYCL_DECLARE_CL_TYPES(r, data, scalar)

13.147.1 Detailed Description

triSYCL wrapper for OpenCL types

Joan DOT Thibault AT ens-rennes DOT fr a.doumoulakis AT gmail DOT com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file opencl_types.hpp.

13.147.2 Macro Definition Documentation

13.147.2.1 TRISYCL_BOOST_COMPUTE_NAME

```
#define TRISYCL_BOOST_COMPUTE_NAME(
              scalar,
              size )  TRISYCL_SIZED_NAME(boost::compute::TRISYCL_TYPE_NAME(scalar), size)
```

Definition at line 139 of file opencl_types.hpp.

13.147.2.2 TRISYCL_DECLARE_CL_TYPES

```
#define TRISYCL_DECLARE_CL_TYPES(
              r,
              data,
              scalar )
```

Value:

```
  TRISYCL_DEFINE_TYPES(scalar, 1)                                   \
  TRISYCL_DEFINE_TYPES(scalar, 2)                                   \
  TRISYCL_DEFINE_TYPES(scalar, 3)                                   \
  TRISYCL_DEFINE_TYPES(scalar, 4)                                   \
  TRISYCL_DEFINE_TYPES(scalar, 8)                                   \
  TRISYCL_DEFINE_TYPES(scalar, 16)
```

Definition at line 170 of file opencl_types.hpp.

13.147.2.3 TRISYCL_DEFINE_TYPES

```
#define TRISYCL_DEFINE_TYPES(
              scalar,
              i )
```

Value:

```
  TRISYCL_H_DEFINE_TYPE(TRISYCL_SIZED_NAME(TRISYCL_TYPE_CL_NAME(scalar), i),    \
                        TRISYCL_BOOST_COMPUTE_NAME(scalar, i),                  \
                        TRISYCL_TYPE_ACTUAL_NAME(scalar), i)
```

Definition at line 160 of file opencl_types.hpp.

13.147.2.4 TRISYCL_H_DEFINE_TYPE

```
#define TRISYCL_H_DEFINE_TYPE(
              cl_type,
              boost_name,
              scalar_type,
              i )
```

Value:

```
  BOOST_PP_IF(BOOST_PP_EQUAL(i, 1), TRISYCL_TYPEDEF_TYPE,                    \
              BOOST_PP_IF(BOOST_PP_EQUAL(i, 2), TRISYCL_WRAPPER_CLASS_2,     \
                          BOOST_PP_IF(BOOST_PP_EQUAL(i, 3),                  \
                                      TRISYCL_WRAPPER_CLASS_3,               \
                                      BOOST_PP_IF(BOOST_PP_EQUAL(i, 4),      \
                                                  TRISYCL_WRAPPER_CLASS_4,   \
                                                  TRISYCL_TYPEDEF_TYPE))))   \
  (cl_type, boost_name, scalar_type)
```

Definition at line 149 of file opencl_types.hpp.

13.147.2.5 TRISYCL_IS_WRAPPER_TRAIT

```
#define TRISYCL_IS_WRAPPER_TRAIT(
              type ) template <> struct is_wrapper<type> :  std::true_type {};
```

Definition at line 85 of file opencl_types.hpp.

13.147.2.6 TRISYCL_SCALAR_TYPES

```
#define TRISYCL_SCALAR_TYPES
```

Value:

```
  BOOST_PP_TUPLE_TO_LIST(                                                    \
    10,                                                                      \
    (                                                                        \
      ( char   ,cl_char,   char),                                            \
      ( uchar  ,cl_uchar,  unsigned char),                                   \
      ( short  ,cl_short,  short int),                                       \
      ( ushort ,cl_ushort, unsigned short int),                              \
      ( int    ,cl_int,    int),                                             \
      ( uint   ,cl_uint,   unsigned int),                                    \
      ( long   ,cl_long,   long int),                                        \
      ( ulong  ,cl_ulong,  unsigned long int),                               \
      ( float  ,cl_float,  float),                                           \
      ( double ,cl_double, double)                                           \
     )                                                                       \
    )
```

Definition at line 35 of file opencl_types.hpp.

13.147.2.7 TRISYCL_SIZED_NAME

```
#define TRISYCL_SIZED_NAME(
              T,
              size )
```

Value:

```
  BOOST_PP_IF(                                                          \
    BOOST_PP_EQUAL(size, 1), T,                                         \
    BOOST_PP_CAT(T, size))
```

Definition at line 58 of file opencl_types.hpp.

13.147.2.8 TRISYCL_TYPE_ACTUAL_NAME

```
#define TRISYCL_TYPE_ACTUAL_NAME(
              T ) BOOST_PP_TUPLE_ELEM(3, 2, T)
```

Definition at line 55 of file opencl_types.hpp.

13.147.2.9 TRISYCL_TYPE_CL_NAME

```
#define TRISYCL_TYPE_CL_NAME(
              T ) BOOST_PP_TUPLE_ELEM(3, 1, T)
```

Definition at line 54 of file opencl_types.hpp.

13.147.2.10 TRISYCL_TYPE_NAME

```
#define TRISYCL_TYPE_NAME(
              T ) BOOST_PP_TUPLE_ELEM(3, 0, T)
```

Definition at line 53 of file opencl_types.hpp.

13.147.2.11 TRISYCL_TYPEDEF_TYPE

```
#define TRISYCL_TYPEDEF_TYPE(
              cl_type,
              boost_name,
              scalar_type )  using cl_type = boost_name##_;
```

Definition at line 145 of file opencl_types.hpp.

13.147.2.12 TRISYCL_WRAPPER_CLASS_2

```
#define TRISYCL_WRAPPER_CLASS_2(
              cl_type,
              boost_name,
              scalar_type )
```

Value:

```
  class cl_type {                                                          \
    ::cl_type self;                                                        \
                                                                           \
  public:                                                                  \
    cl_type () = default;                                                  \
    cl_type (::cl_type self_) :  self { self_ } {}                          \
    cl_type (scalar_type x, scalar_type y) :  self { x, y } {}              \
    auto& x() { return self.s[0]; }                                        \
    auto& y() { return self.s[1]; }                                        \
    auto& unwrap()const { return self; }};                                 \
  TRISYCL_IS_WRAPPER_TRAIT(cl_type)
```

Definition at line 93 of file opencl_types.hpp.

13.147.2.13 TRISYCL_WRAPPER_CLASS_3

```
#define TRISYCL_WRAPPER_CLASS_3(
              cl_type,
              boost_name,
              scalar_type )
```

Value:

```
  class cl_type {                                                          \
    ::cl_type self;                                                        \
                                                                           \
  public:                                                                  \
    cl_type () = default;                                                  \
    cl_type (::cl_type self_) :  self { self_ } {}                          \
    cl_type (scalar_type x, scalar_type y, scalar_type z) :                                \
      self { x, y, z } {}                                                  \
    auto& x() { return self.s[0]; }                                        \
    auto& y() { return self.s[1]; }                                        \
    auto& z() { return self.s[2]; }                                        \
    auto& unwrap()const { return self; }};                                 \
  TRISYCL_IS_WRAPPER_TRAIT(cl_type)
```

Definition at line 106 of file opencl_types.hpp.

13.147.2.14 TRISYCL_WRAPPER_CLASS_4

```
#define TRISYCL_WRAPPER_CLASS_4(
              cl_type,
              boost_name,
              scalar_type )
```

Value:

```
  class cl_type {                                                          \
    ::cl_type self;                                                        \
                                                                           \
  public:                                                                  \
```

```
  cl_type () = default;                                                          \
  cl_type (::cl_type self_) :  self { self_ } {}                                  \
  cl_type (scalar_type x, scalar_type y, scalar_type z, scalar_type w) :          \
    self { x, y, z, w } {}                                                        \
  auto& x() { return self.s[0]; }                                                 \
  auto& y() { return self.s[1]; }                                                 \
  auto& z() { return self.s[2]; }                                                 \
  auto& w() { return self.s[3]; }                                                 \
  auto& unwrap()const { return self; }};                                         \
TRISYCL_IS_WRAPPER_TRAIT(cl_type)
```

Definition at line 121 of file opencl_types.hpp.

13.148 opencl_types.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_OPENCL_TYPES_HPP
00002 #define TRISYCL_SYCL_OPENCL_TYPES_HPP
00003
00004 /** \file
00005 triSYCL wrapper for OpenCL types
00006
00007 Joan DOT Thibault AT ens-rennes DOT fr
00008 a.doumoulakis AT gmail DOT com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include <boost/preprocessor/cat.hpp>
00015 #include <boost/preprocessor/comparison/equal.hpp>
00016 #include <boost/preprocessor/control/if.hpp>
00017 #include <boost/preprocessor/facilities/empty.hpp>
00018 #include <boost/preprocessor/list/for_each.hpp>
00019 #include <boost/preprocessor/logical/not.hpp>
00020 #include <boost/preprocessor/logical/or.hpp>
00021 #include <boost/preprocessor/seq/for_each.hpp>
00022 #include <boost/preprocessor/tuple/to_list.hpp>
00023 #include <boost/preprocessor/tuple/elem.hpp>
00024
00025 #ifdef TRISYCL_OPENCL
00026 #include <boost/compute/types/fundamental.hpp>
00027 #else
00028 #include "triSYCL/vec.hpp"
00029 #endif
00030
00031
00032 namespace trisycl {
00033
00034 // List of the cl_types that will be iterated upon to generate the vector types
00035 #define TRISYCL_SCALAR_TYPES                                                  \
00036 BOOST_PP_TUPLE_TO_LIST(                                                       \
00037 10,                                                                           \
00038 (                                                                             \
00039 ( char   ,cl_char,   char),                                                   \
00040 ( uchar  ,cl_uchar,  unsigned char),                                          \
00041 ( short  ,cl_short,  short int),                                             \
00042 ( ushort ,cl_ushort, unsigned short int),                                     \
00043 ( int    ,cl_int,    int),                                                    \
00044 ( uint   ,cl_uint,   unsigned int),                                          \
00045 ( long   ,cl_long,   long int),                                             \
00046 ( ulong  ,cl_ulong,  unsigned long int),                                     \
00047 ( float  ,cl_float,  float),                                                 \
00048 ( double ,cl_double, double)                                                \
00049 )                                                                             \
00050 )
00051
00052 // Accessors to get the type name, the cl_type name and the actual type
00053 #define TRISYCL_TYPE_NAME(T)         BOOST_PP_TUPLE_ELEM(3, 0, T)
00054 #define TRISYCL_TYPE_CL_NAME(T)      BOOST_PP_TUPLE_ELEM(3, 1, T)
00055 #define TRISYCL_TYPE_ACTUAL_NAME(T)  BOOST_PP_TUPLE_ELEM(3, 2, T)
00056
00057 // Return the name of the type concatenated with the size unless the size is 1
00058 #define TRISYCL_SIZED_NAME(T, size)                                           \
00059 BOOST_PP_IF(                                                                  \
00060 BOOST_PP_EQUAL(size, 1), T,                                                  \
00061 BOOST_PP_CAT(T, size))
00062
00063 #ifndef TRISYCL_OPENCL
00064
```

```
00065 /* If we are not using Boost Compute we just every cl_type with its \c vec
00066 equivalent
00067 */
00068 #define TRISYCL_DEFINE_TYPES(scalar, i)                                        \
00069 using TRISYCL_SIZED_NAME(TRISYCL_TYPE_CL_NAME(scalar), i) =                    \
00070 BOOST_PP_IF(                                                                   \
00071 BOOST_PP_EQUAL(i, 1),                                                          \
00072 TRISYCL_TYPE_ACTUAL_NAME(scalar),                                              \
00073 BOOST_PP_CAT(TRISYCL_TYPE_NAME(scalar), i));
00074
00075 #else
00076
00077 /* When passing the arguments to Boost Compute, we need to know if we have to
00078 unwrap our type and give Boost Compute the actual OpenCL type.  To do this
00079 we define this meta-function that returns true if \c T is a wrapper type
00080 to an OpenCL type.  For example \c is_wrapper<trisycl::int3>::value == true
00081 */
00082 template<class T>
00083 struct is_wrapper :  std::false_type {};
00084
00085 #define TRISYCL_IS_WRAPPER_TRAIT(type)                                         \
00086 template <> struct is_wrapper<type> :  std::true_type {};
00087
00088 /* We define 3 different wrapper classes around OpenCL types.
00089 These classes allow us to use t.x() and t.y() for vector types of size 2.
00090 And in addition we can also use t.z() and t.w() for vector types of size
00091 3 and 4 similarly to OpenCL vector types
00092 */
00093 #define TRISYCL_WRAPPER_CLASS_2(cl_type, boost_name, scalar_type)              \
00094 class cl_type {                                                                \
00095 ::cl_type self;                                                                \
00096 \
00097 public:
    \
00098 cl_type () = default;                                                          \
00099 cl_type (::cl_type self_) :  self { self_ } {}                                 \
00100 cl_type (scalar_type x, scalar_type y) :  self { x, y } {}                     \
00101 auto& x() { return self.s[0]; }                                                \
00102 auto& y() { return self.s[1]; }                                                \
00103 auto& unwrap() const { return self; }};                                        \
00104 TRISYCL_IS_WRAPPER_TRAIT(cl_type)
00105
00106 #define TRISYCL_WRAPPER_CLASS_3(cl_type, boost_name, scalar_type)              \
00107 class cl_type {                                                                \
00108 ::cl_type self;                                                                \
00109 \
00110 public:
    \
00111 cl_type () = default;                                                          \
00112 cl_type (::cl_type self_) :  self { self_ } {}                                 \
00113 cl_type (scalar_type x, scalar_type y, scalar_type z) :                                \
00114 self { x, y, z } {}                                                            \
00115 auto& x() { return self.s[0]; }                                                \
00116 auto& y() { return self.s[1]; }                                                \
00117 auto& z() { return self.s[2]; }                                                \
00118 auto& unwrap() const { return self; }};                                        \
00119 TRISYCL_IS_WRAPPER_TRAIT(cl_type)
00120
00121 #define TRISYCL_WRAPPER_CLASS_4(cl_type, boost_name, scalar_type)              \
00122 class cl_type {                                                                \
00123 ::cl_type self;                                                                \
00124 \
00125 public:
    \
00126 cl_type () = default;                                                          \
00127 cl_type (::cl_type self_) :  self { self_ } {}                                 \
00128 cl_type (scalar_type x, scalar_type y, scalar_type z, scalar_type w) :         \
00129 self { x, y, z, w } {}                                                         \
00130 auto& x() { return self.s[0]; }                                                \
00131 auto& y() { return self.s[1]; }                                                \
00132 auto& z() { return self.s[2]; }                                                \
00133 auto& w() { return self.s[3]; }                                                \
00134 auto& unwrap() const { return self; }};                                        \
00135 TRISYCL_IS_WRAPPER_TRAIT(cl_type)
00136
00137
00138 // Return the Boost Compute type for OpenCL vector types
00139 #define TRISYCL_BOOST_COMPUTE_NAME(scalar, size)                               \
00140 TRISYCL_SIZED_NAME(boost::compute::TRISYCL_TYPE_NAME(scalar), size)
00141
00142 /* For vector types of size above 4 and 1 we typedef the vector type with
00143 its Boost Compute equivalent
00144 */
00145 #define TRISYCL_TYPEDEF_TYPE(cl_type, boost_name, scalar_type)                 \
00146 using cl_type = boost_name##_;
00147
00148 // Helper macro to properly define each OpenCL type
```

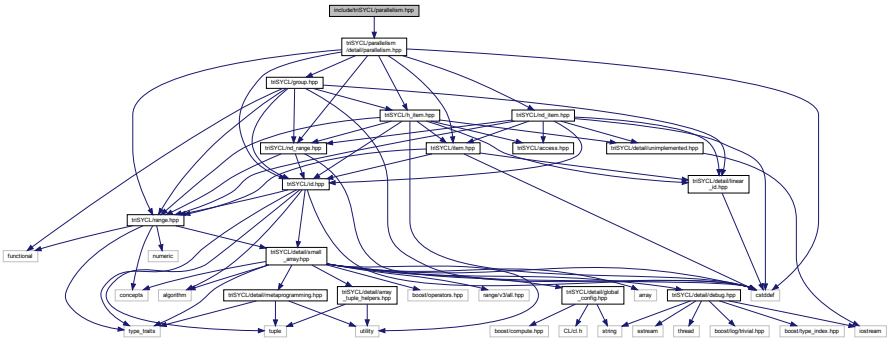
```
00149 #define TRISYCL_H_DEFINE_TYPE(cl_type, boost_name, scalar_type, i)            \
00150 BOOST_PP_IF(BOOST_PP_EQUAL(i, 1), TRISYCL_TYPEDEF_TYPE,                      \
00151 BOOST_PP_IF(BOOST_PP_EQUAL(i, 2), TRISYCL_WRAPPER_CLASS_2,       \
00152 BOOST_PP_IF(BOOST_PP_EQUAL(i, 3),                     \
00153 TRISYCL_WRAPPER_CLASS_3,                \
00154 BOOST_PP_IF(BOOST_PP_EQUAL(i, 4),        \
00155 TRISYCL_WRAPPER_CLASS_4,     \
00156 TRISYCL_TYPEDEF_TYPE))))     \
00157 (cl_type, boost_name, scalar_type)
00158
00159 // Helper macro to properly define each OpenCL type
00160 #define TRISYCL_DEFINE_TYPES(scalar, i)                                       \
00161 TRISYCL_H_DEFINE_TYPE(TRISYCL_SIZED_NAME(TRISYCL_TYPE_CL_NAME(scalar), i),   \
00162 TRISYCL_BOOST_COMPUTE_NAME(scalar, i),                \
00163 TRISYCL_TYPE_ACTUAL_NAME(scalar), i)
00164
00165 #endif
00166
00167 /* We declare vector types of sizes 1,2,3,4,8 and 16 as per
00168 the SYCL specification
00169 */
00170 #define TRISYCL_DECLARE_CL_TYPES(r, data, scalar)                             \
00171 TRISYCL_DEFINE_TYPES(scalar, 1)                                             \
00172 TRISYCL_DEFINE_TYPES(scalar, 2)                                             \
00173 TRISYCL_DEFINE_TYPES(scalar, 3)                                             \
00174 TRISYCL_DEFINE_TYPES(scalar, 4)                                             \
00175 TRISYCL_DEFINE_TYPES(scalar, 8)                                             \
00176 TRISYCL_DEFINE_TYPES(scalar, 16)
00177
00178 // Generate the vector types for all listed scalar types
00179 BOOST_PP_LIST_FOR_EACH(TRISYCL_DECLARE_CL_TYPES, _, TRISYCL_SCALAR_TYPES)
00180
00181 } // cl
00182
00183
00184 // Undef macros to avoid name collision
00185 #undef TRISYCL_SCALAR_TYPES
00186 #undef TRISYCL_TYPE_NAME
00187 #undef TRISYCL_TYPE_CL_NAME
00188 #undef TRISYCL_TYPE_ACTUAL_NAME
00189 #undef TRISYCL_SIZED_NAME
00190 #undef TRISYCL_IS_WRAPPER_TRAIT
00191 #undef TRISYCL_WRAPPER_CLASS_2
00192 #undef TRISYCL_WRAPPER_CLASS_3
00193 #undef TRISYCL_WRAPPER_CLASS_4
00194 #undef TRISYCL_BOOST_COMPUTE_NAME
00195 #undef TRISYCL_TYPEDEF_TYPE
00196 #undef TRISYCL_DEFINE_TYPES
00197 #undef TRISYCL_DECLARE_CL_TYPES
00198
00199 /*
00200 # Some Emacs stuff:
00201 ### Local Variables:
00202 ### ispell-local-dictionary:  "american"
00203 ### eval:  (flyspell-prog-mode)
00204 ### End:
00205 */
00206
00207 #endif // TRISYCL_SYCL_OPENCL_TYPES_HPP
```

13.149 include/triSYCL/parallelism.hpp File Reference

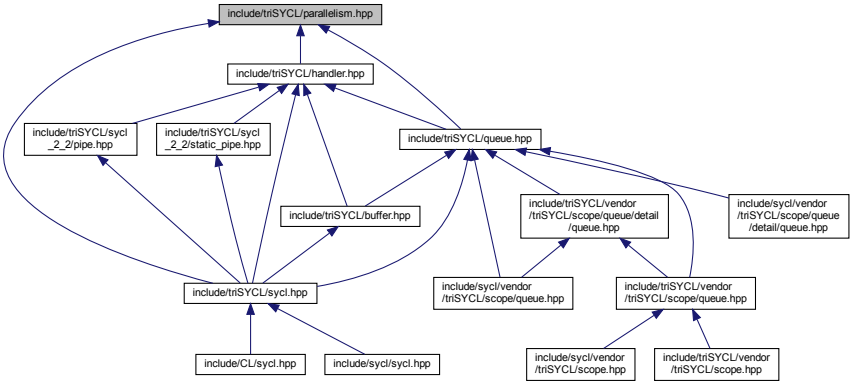
Implement parallel constructions to launch kernels.

```
#include "triSYCL/parallelism/detail/parallelism.hpp"
```

Include dependency graph for parallelism.hpp:



This graph shows which files directly or indirectly include this file:



13.149.1 Detailed Description

Implement parallel constructions to launch kernels.

Ronan at keryell dot FR

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file parallelism.hpp.

13.150 parallelism.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_PARALLELISM_HPP
00002 #define TRISYCL_SYCL_PARALLELISM_HPP
00003
00004 /** \file
00005
00006 Implement parallel constructions to launch kernels
00007
```

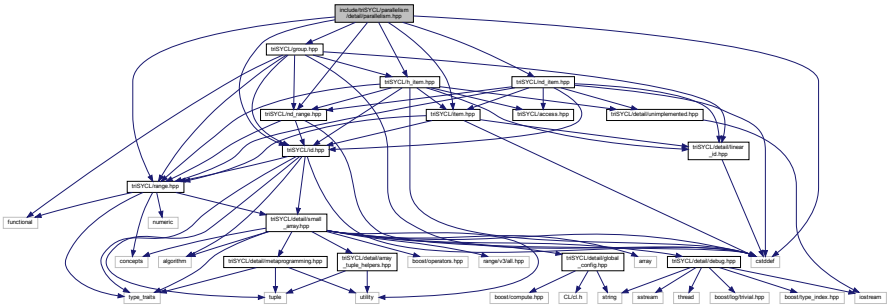
```
00008 Ronan at keryell dot FR
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #ifdef TRISYCL_TBB
00015 #include "triSYCL/parallelism/detail/parallelism_tbb.hpp"
00016 #else
00017 #include "triSYCL/parallelism/detail/parallelism.hpp"
00018 #endif
00019
00020 /*
00021 # Some Emacs stuff:
00022 ### Local Variables:
00023 ### ispell-local-dictionary:  "american"
00024 ### eval:  (flyspell-prog-mode)
00025 ### End:
00026 */
00027
00028 #endif // TRISYCL_SYCL_PARALLELISM_HPP
```

13.151 include/triSYCL/parallelism/detail/parallelism.hpp File Reference

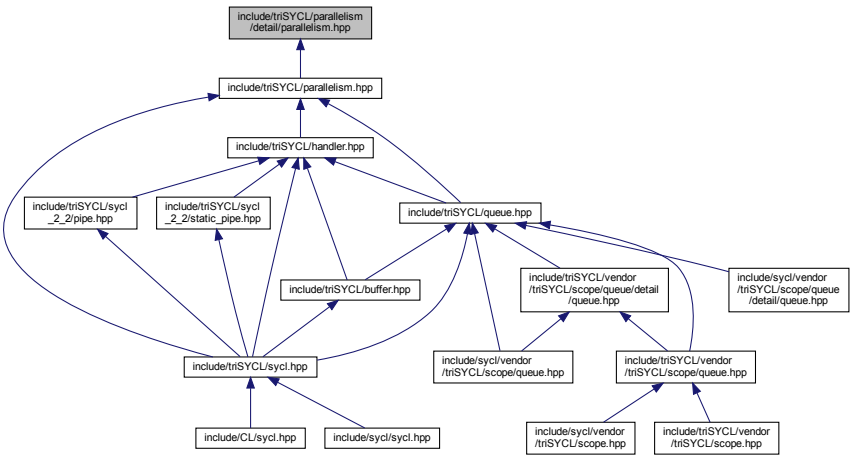
Implement the detail of the parallel constructions to launch kernels.

```
#include <cstddef>
#include "triSYCL/group.hpp"
#include "triSYCL/h_item.hpp"
#include "triSYCL/id.hpp"
#include "triSYCL/item.hpp"
#include "triSYCL/nd_item.hpp"
#include "triSYCL/nd_range.hpp"
#include "triSYCL/range.hpp"
```

Include dependency graph for parallelism.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::detail::parallel_for_iterate< level, Range, ParallelForFunctor, Id >

  A recursive multi-dimensional iterator that ends up calling f. More...
- struct trisycl::detail::parallel_for_iterate< 0, Range, ParallelForFunctor, Id >

  Stop the recursion when level reaches 0 by simply calling the kernel functor with the constructed id. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Functions

- template<int Dimensions = 1, typename ParallelForFunctor , typename Id >
  void trisycl::detail::parallel_for (range< Dimensions > r, ParallelForFunctor f, Id)

  Implementation of a data parallel computation with parallelism specified at launch time by a range<>.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for (range< Dimensions > r, ParallelForFunctor f, item< Dimensions >)

  Implementation of a data parallel computation with parallelism specified at launch time by a range<>.
- template<typename F , typename R , typename A >
  auto trisycl::detail::capture_arg_v (R(F::*)(A))
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for (range< Dimensions > r, ParallelForFunctor f)

  Calls the appropriate ternary parallel_for overload based on the index type of the kernel function object f.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for_global_offset (range< Dimensions > global_size, id< Dimensions > offset, ParallelForFunctor f)

  Implementation of parallel_for with a range<> and an offset.

- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for_workgroup (nd_range< Dimensions > r, ParallelForFunctor f)

  Implement the loop on the work-groups.
- template<int Dimensions, typename T_Item , typename ParallelForFunctor >
  void trisycl::detail::parallel_for_workitem (const group< Dimensions > &g, ParallelForFunctor f)

  Implement the loop on the work-items inside a work-group.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for (nd_range< Dimensions > r, ParallelForFunctor f)

  Implement a variation of parallel_for to take into account a nd_range<>
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for_workitem_in_group (const group< Dimensions > &g, ParallelForFunctor f)

  Implement the loop on the work-items inside a work-group.

13.151.1 Detailed Description

Implement the detail of the parallel constructions to launch kernels.

Todo Refactor this file

Ronan at keryell dot FR

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file parallelism.hpp.

13.152 parallelism.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_PARALLELISM_DETAIL_PARALLELISM_HPP
00002 #define TRISYCL_SYCL_PARALLELISM_DETAIL_PARALLELISM_HPP
00003
00004 /** \file
00005
00006 Implement the detail of the parallel constructions to launch kernels
00007
00008 \todo Refactor this file
00009
00010 Ronan at keryell dot FR
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include <cstddef>
00017
00018 #include "triSYCL/group.hpp"
00019 #include "triSYCL/h_item.hpp"
00020 #include "triSYCL/id.hpp"
00021 #include "triSYCL/item.hpp"
00022 #include "triSYCL/nd_item.hpp"
00023 #include "triSYCL/nd_range.hpp"
00024 #include "triSYCL/range.hpp"
00025
00026 #if defined(TRISYCL_USE_OPENCL_ND_RANGE)
00027 #include "triSYCL/detail/SPIR/opencl_spir_helpers.hpp"
00028 #endif
00029
00030 #ifdef _OPENMP
00031 #include <omp.h>
00032 #endif
00033
00034
00035 /** \addtogroup parallelism
```

```
00036 @{
00037 */
00038
00039 namespace trisycl::detail {
00040
00041
00042 /** A recursive multi-dimensional iterator that ends up calling f
00043
00044 The iteration order may be changed later.
00045
00046 Since partial specialization of function template is not possible in
00047 C++14, use a class template instead with everything in the
00048 constructor.
00049 */
00050 template <std::size_t level,
00051           typename Range,
00052           typename ParallelForFunctor,
00053           typename Id>
00054 struct parallel_for_iterate {
00055   parallel_for_iterate(Range r, ParallelForFunctor &f, Id &index) {
00056     for (std::size_t _sycl_index = 0,
00057            _sycl_end = r[r.rank() - level];
00058          _sycl_index < _sycl_end;
00059          _sycl_index++) {
00060       // Set the current value of the index for this dimension
00061       index[r.rank() - level] = _sycl_index;
00062       // Iterate further on lower dimensions
00063       parallel_for_iterate<level - 1,
00064                            Range,
00065                            ParallelForFunctor,
00066                            Id> { r, f, index };
00067     }
00068   }
00069 };
00070
00071 #ifdef _OPENMP
00072 /** A top-level recursive multi-dimensional iterator variant using OpenMP
00073
00074 Only the top-level loop uses OpenMP and goes on with the normal
00075 recursive multi-dimensional.
00076 */
00077 template <std::size_t level,
00078           typename Range,
00079           typename ParallelForFunctor,
00080           typename Id>
00081 struct parallel_OpenMP_for_iterate {
00082   parallel_OpenMP_for_iterate(Range r, ParallelForFunctor &f) {
00083     // Create the OpenMP threads before the for-loop to avoid creating an
00084     // index in each iteration
00085 #pragma omp parallel
00086     {
00087       // Allocate an OpenMP thread-local index
00088       Id index;
00089       // Make a simple loop end condition for OpenMP
00090       std::size_t _sycl_end = r[r.rank() - level];
00091       /* Distribute the iterations on the OpenMP threads.  Some OpenMP
00092 "collapse" could be useful for small iteration space, but it
00093 would need some template specialization to have real contiguous
00094 loop nests */
00095 #pragma omp for
00096       for (std::size_t _sycl_index = 0;
00097            _sycl_index < _sycl_end;
00098            _sycl_index++) {
00099         // Set the current value of the index for this dimension
00100         index[r.rank() - level] = _sycl_index;
00101         // Iterate further on lower dimensions
00102         parallel_for_iterate<level - 1,
00103                              Range,
00104                              ParallelForFunctor,
00105                              Id> { r, f, index };
00106       }
00107     }
00108   }
00109 };
00110 #endif
00111
00112
00113 /** Stop the recursion when level reaches 0 by simply calling the
00114 kernel functor with the constructed id */
00115 template <typename Range, typename ParallelForFunctor, typename Id>
00116 struct parallel_for_iterate<0, Range, ParallelForFunctor, Id> {
00117   parallel_for_iterate(Range r, ParallelForFunctor &f, Id &index) {
00118     f(index);
00119   }
00120 };
00121
00122
```

```
00123 /** Implementation of a data parallel computation with parallelism
00124 specified at launch time by a range<>.  Kernel index is id or int.
00125
00126 This implementation use OpenMP 3 if compiled with the right flag.
00127 */
00128 template <int Dimensions = 1, typename ParallelForFunctor, typename Id>
00129 void parallel_for(range<Dimensions> r,
00130                   ParallelForFunctor f,
00131                   Id) {
00132 #ifdef _OPENMP
00133   // Use OpenMP for the top loop level
00134   parallel_OpenMP_for_iterate<Dimensions,
00135                               range<Dimensions>,
00136                               ParallelForFunctor,
00137                               id<Dimensions>> { r, f };
00138 #else
00139   // In a sequential execution there is only one index processed at a time
00140   id<Dimensions> index;
00141   parallel_for_iterate<Dimensions,
00142                        range<Dimensions>,
00143                        ParallelForFunctor,
00144                        id<Dimensions>> { r, f, index };
00145 #endif
00146 }
00147
00148
00149 /** Implementation of a data parallel computation with parallelism
00150 specified at launch time by a range<>.  Kernel index is item.
00151
00152 This implementation use OpenMP 3 if compiled with the right flag.
00153 */
00154 template <int Dimensions = 1, typename ParallelForFunctor>
00155 void parallel_for(range<Dimensions> r,
00156                   ParallelForFunctor f,
00157                   item<Dimensions>) {
00158   auto reconstruct_item = [&] (id<Dimensions> l) {
00159     // Reconstruct the global item
00160     item<Dimensions> index { r, l };
00161     // Call the user kernel with the item<> instead of the id<>
00162     f(index);
00163   };
00164 #ifdef _OPENMP
00165   // Use OpenMP for the top loop level
00166   parallel_OpenMP_for_iterate<Dimensions,
00167                               range<Dimensions>,
00168                               decltype(reconstruct_item),
00169                               id<Dimensions>> { r, reconstruct_item };
00170 #else
00171   // In a sequential execution there is only one index processed at a time
00172   id<Dimensions> index;
00173   parallel_for_iterate<Dimensions,
00174                        range<Dimensions>,
00175                        decltype(reconstruct_item),
00176                        id<Dimensions>> { r, reconstruct_item, index };
00177 #endif
00178 }
00179
00180
00181 /* These helpers work specifically for the parallel_for overload:
00182 parallel_for(range<Dimensions> r, ParallelForFunctor f)
00183
00184 They generate a value from the type of the passed in kernels(lambda)
00185 argument (item or id) and pass it onwards to parallel_for.
00186
00187 From the C++ spec I believe lambda's should only require const or no
00188 qualification for the forseeable future (at least by default).  Const in the
00189 case where the mutable keyword has not been used and no const when mutable
00190 has been used.
00191
00192 \todo A future change may be to modify htat to capture_arg_t and alter the
00193 parallel_for call stack to take a type rather than value.
00194 \todo In c++20 these may be redefineable as inline lambda's of the form:
00195 [captures] <tparams> (params) {body}
00196 */
00197 template<typename F, typename R, typename A>
00198 auto capture_arg_v(R(F::*)(A)) { return A{}; }
00199
00200 template<typename F, typename R, typename A>
00201 auto capture_arg_v(R(F::*)(A) const) { return A{}; }
00202
00203 /** Calls the appropriate ternary parallel_for overload based on the
00204 index type of the kernel function object f
00205
00206 capture_arg_v generates a value from the type of the passed in kernels
00207 (lambdas) argument (item or id in this case) and passes it onwards to
00208 parallel_for so that it's aware which index class it's supposed to generate
00209 data for.
```

```
00210
00211 We erase the type then retrieve it again so that we can avoid redefining
00212 the parallel_for interface in handler.hpp for item and id as the
00213 parallel_for in the SYCL 1.2.1 spec that accepts a range can be overloaded
00214 for both.
00215 */
00216 #if !defined(TRISYCL_USE_OPENCL_ND_RANGE)
00217 template <int Dimensions = 1, typename ParallelForFunctor>
00218 void parallel_for(range<Dimensions> r, ParallelForFunctor f) {
00219   parallel_for(r,f, capture_arg_v(&ParallelForFunctor::operator()));
00220 }
00221 #else
00222 template <int Dimensions = 1, typename ParallelForFunctor>
00223 void parallel_for(range<Dimensions> r, ParallelForFunctor f) {
00224   f(sycl::detail::spir::create_parallel_for_arg<Dimensions>(capture_arg_v(
00225     &ParallelForFunctor::operator())));
00226 }
00227 #endif
00228
00229
00230 /** Implementation of parallel_for with a range<> and an offset */
00231 template <int Dimensions = 1, typename ParallelForFunctor>
00232 void parallel_for_global_offset(range<Dimensions> global_size,
00233                                 id<Dimensions> offset,
00234                                 ParallelForFunctor f) {
00235   // Reconstruct the item from its id<> and its offset
00236   auto reconstruct_item = [&] (id<Dimensions> l) {
00237     // Reconstruct the global item
00238     item<Dimensions> index { global_size, l + offset, offset };
00239     // Call the user kernel with the item<> instead of the id<>
00240     f(index);
00241   };
00242
00243   // First iterate on all the work-groups
00244   parallel_for(global_size, reconstruct_item);
00245 }
00246
00247
00248 /// Implement the loop on the work-groups
00249 template <int Dimensions = 1, typename ParallelForFunctor>
00250 void parallel_for_workgroup(nd_range<Dimensions> r,
00251                             ParallelForFunctor f) {
00252   // In a sequential execution there is only one index processed at a time
00253   group<Dimensions> g { r };
00254
00255   // First iterate on all the work-groups
00256   parallel_for_iterate<Dimensions,
00257                        range<Dimensions>,
00258                        ParallelForFunctor,
00259                        group<Dimensions>> {
00260     r.get_group_range(),
00261     f,
00262     g };
00263 }
00264
00265
00266 /** Implement the loop on the work-items inside a work-group
00267
00268 \todo Better type the functor
00269 */
00270 template <int Dimensions, typename T_Item, typename ParallelForFunctor>
00271 void parallel_for_workitem(const group<Dimensions> &g,
00272                            ParallelForFunctor f) {
00273 #if defined(_OPENMP) && (!defined(TRISYCL_NO_BARRIER) && !defined(_MSC_VER))
00274   /* To implement barriers with OpenMP, one thread is created for each
00275 work-item in the group and thus an OpenMP barrier has the same effect
00276 of an OpenCL barrier executed by the work-items in a workgroup
00277
00278 The issue is that the parallel_for_workitem() execution is slow even
00279 when nd_item::barrier() is not used
00280 */
00281   range<Dimensions> l_r = g.get_nd_range().get_local_range();
00282   id<Dimensions> id_l_r { l_r };
00283
00284   auto tot = l_r.size();
00285
00286   if constexpr (Dimensions == 1) {
00287 #pragma omp parallel for collapse(1) schedule(static) num_threads(tot)
00288     for (size_t i = 0; i < l_r.get(0); ++i) {
00289       T_Item index{g.get_nd_range()};
00290       index.set_local(i);
00291       index.set_global(index.get_local_id() + id_l_r * g.get_id());
00292       f(index);
00293     }
00294   } else if constexpr (Dimensions == 2) {
00295 #pragma omp parallel for collapse(2) schedule(static) num_threads(tot)
00296     for (size_t i = 0; i < l_r.get(0); ++i) {
```

```
00297       for (size_t j = 0; j < l_r.get(1); ++j) {
00298         T_Item index{g.get_nd_range()};
00299         index.set_local({i,j});
00300         index.set_global(index.get_local_id() + id_l_r * g.get_id());
00301         f(index);
00302       }
00303     }
00304   } else if constexpr (Dimensions == 3) {
00305 #pragma omp parallel for collapse(3) schedule(static) num_threads(tot)
00306     for (size_t i = 0; i < l_r.get(0); ++i)
00307       for (size_t j = 0; j < l_r.get(1); ++j)
00308         for (size_t k = 0; k < l_r.get(2); ++k) {
00309           T_Item index{g.get_nd_range()};
00310           index.set_local({i,j,k});
00311           index.set_global(index.get_local_id() + id_l_r * g.get_id());
00312           f(index);
00313         }
00314   }
00315 #elif defined(_OPENMP) && (defined(TRISYCL_NO_BARRIER) && !defined(_MSC_VER))
00316   range<Dimensions> l_r = g.get_nd_range().get_local_range();
00317   id<Dimensions> id_l_r { l_r };
00318
00319   if constexpr (Dimensions == 1) {
00320 #pragma omp parallel for simd collapse(1)
00321     for (size_t i = 0; i < l_r.get(0); ++i) {
00322       T_Item index{g.get_nd_range()};
00323       index.set_local(i);
00324       index.set_global(index.get_local_id() + id_l_r * g.get_id());
00325       f(index);
00326     }
00327   } else if constexpr (Dimensions == 2) {
00328 #pragma omp parallel for simd collapse(2)
00329     for (size_t i = 0; i < l_r.get(0); ++i) {
00330       for (size_t j = 0; j < l_r.get(1); ++j) {
00331         T_Item index{g.get_nd_range()};
00332         index.set_local({i,j});
00333         index.set_global(index.get_local_id() + id_l_r * g.get_id());
00334         f(index);
00335       }
00336     }
00337   } else if constexpr (Dimensions == 3) {
00338 #pragma omp parallel for simd collapse(3)
00339     for (size_t i = 0; i < l_r.get(0); ++i)
00340       for (size_t j = 0; j < l_r.get(1); ++j)
00341         for (size_t k = 0; k < l_r.get(2); ++k) {
00342           T_Item index{g.get_nd_range()};
00343           index.set_local({i,j,k});
00344           index.set_global(index.get_local_id() + id_l_r * g.get_id());
00345           f(index);
00346         }
00347   }
00348 #else
00349   // In a sequential execution there is only one index processed at a time
00350   T_Item index { g.get_nd_range() };
00351   // To iterate on the local work-item
00352   id<Dimensions> local;
00353
00354   // Reconstruct the item from its group and local id
00355   auto reconstruct_item = [&] (id<Dimensions> l) {
00356     //local.display();
00357     //l.display();
00358     // Reconstruct the global item
00359     index.set_local(local);
00360     // \todo Some strength reduction here
00361     index.set_global(local + id<Dimensions>(g.get_local_range())*g.get_id());
00362     // Call the user kernel at last
00363     f(index);
00364   };
00365
00366   // Then iterate on all the work-items of the work-group
00367   parallel_for_iterate<Dimensions,
00368                        range<Dimensions>,
00369                        decltype(reconstruct_item),
00370                        id<Dimensions>> {
00371     g.get_local_range(),
00372     reconstruct_item,
00373     local };
00374 #endif
00375 }
00376
00377
00378 /** Implement a variation of parallel_for to take into account a
00379 nd_range<>
00380
00381 \todo Add an OpenMP implementation
00382
00383 \todo Deal with incomplete work-groups
```

```
00384
00385 \todo Implement with parallel_for_workgroup()/parallel_for_workitem()
00386 */
00387 template <int Dimensions = 1, typename ParallelForFunctor>
00388 void parallel_for(nd_range<Dimensions> r,
00389                   ParallelForFunctor f) {
00390   // To iterate on the work-group
00391   id<Dimensions> group;
00392   range<Dimensions> group_range = r.get_group_range();
00393
00394 #ifdef _OPENMP
00395
00396   auto iterate_in_work_group = [&] (id<Dimensions> g) {
00397     //group.display();
00398
00399     // Then iterate on the local work-groups
00400     trisycl::group<Dimensions> wg {g, r};
00401     parallel_for_workitem<Dimensions,
00402                           nd_item<Dimensions>,
00403                           decltype(f)>(wg, f);
00404   };
00405
00406 #else
00407
00408   // In a sequential execution there is only one index processed at a time
00409   nd_item<Dimensions> index { r };
00410
00411   // To iterate on the local work-item
00412   id<Dimensions> local;
00413   range<Dimensions> local_range = r.get_local_range();
00414
00415   // Reconstruct the item from its group and local id
00416   auto reconstruct_item = [&] (id<Dimensions> l) {
00417     //local.display();
00418     // Reconstruct the global item
00419     index.set_local(local);
00420     // Upgrade local_range to an id<> so that we can * with the group (an id<>)
00421     index.set_global(local + id<Dimensions>(local_range)*group);
00422     // Call the user kernel at last
00423     f(index);
00424   };
00425
00426   /* To recycle the parallel_for on range<>, wrap the ParallelForFunctor f
00427 into another functor that iterates inside the work-group and then
00428 calls f */
00429   auto iterate_in_work_group = [&] (id<Dimensions> g) {
00430     //group.display();
00431     // Then iterate on the local work-groups
00432     parallel_for_iterate<Dimensions,
00433                          range<Dimensions>,
00434                          decltype(reconstruct_item),
00435                          id<Dimensions>> { local_range,
00436                                            reconstruct_item,
00437                                            local };
00438   };
00439
00440 #endif
00441
00442   // First iterate on all the work-groups
00443   parallel_for_iterate<Dimensions,
00444                        range<Dimensions>,
00445                        decltype(iterate_in_work_group),
00446                        id<Dimensions>> { group_range,
00447                                          iterate_in_work_group,
00448                                          group };
00449 }
00450
00451
00452 /** Implement the loop on the work-items inside a work-group
00453 */
00454 template <int Dimensions, typename ParallelForFunctor>
00455 void parallel_for_workitem_in_group(const group<Dimensions> &g,
00456                                     ParallelForFunctor f) {
00457   parallel_for_workitem<Dimensions,
00458                         h_item<Dimensions>,
00459                         ParallelForFunctor>(g, f);
00460 }
00461
00462
00463 /// @} End the parallelism Doxygen group
00464
00465 }
00466
00467 /*
00468 # Some Emacs stuff:
00469 ### Local Variables:
00470 ### ispell-local-dictionary:  "american"
```

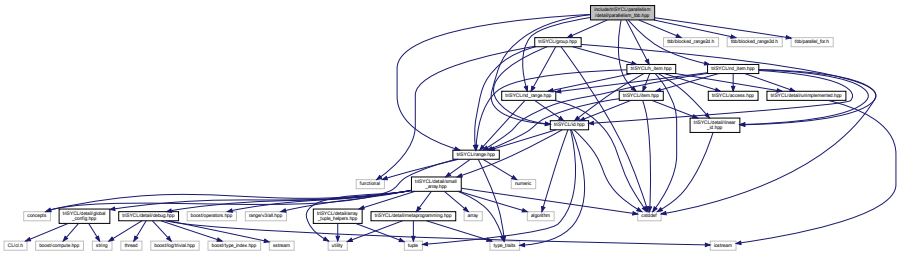
```
00471 ### eval:  (flyspell-prog-mode)
00472 ### End:
00473 */
00474
00475 #endif // TRISYCL_SYCL_PARALLELISM_DETAIL_PARALLELISM_HPP
```

13.153 include/triSYCL/parallelism/detail/parallelism_tbb.hpp File Reference

Implement the detail of the parallel constructions to launch kernels using Intel Threading Building Blocks (TBB).

```
#include "triSYCL/group.hpp"
#include "triSYCL/h_item.hpp"
#include "triSYCL/id.hpp"
#include "triSYCL/item.hpp"
#include "triSYCL/nd_item.hpp"
#include "triSYCL/nd_range.hpp"
#include "triSYCL/range.hpp"
#include <tbb/blocked_range2d.h>
#include <tbb/blocked_range3d.h>
#include <tbb/parallel_for.h>
```

Include dependency graph for parallelism_tbb.hpp:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Functions

- static auto trisycl::detail::to_tbb_range (const range< 1 > &r)
- static auto trisycl::detail::to_tbb_range (const range< 2 > &r)
- static auto trisycl::detail::to_tbb_range (const range< 3 > &r)
- static auto trisycl::detail::to_sycl_id (const tbb::blocked_range< size_t > &in)
- static auto trisycl::detail::to_sycl_id (const tbb::blocked_range2d< size_t > &in)
- static auto trisycl::detail::to_sycl_id (const tbb::blocked_range3d< size_t > &in)
- template<typename Range , typename ParallelForFunctor >
  void trisycl::detail::parallel_for_iterate (Range r, ParallelForFunctor &f)
- template<int Dimensions = 1, typename ParallelForFunctor , typename Id >
  void trisycl::detail::parallel_for (range< Dimensions > r, ParallelForFunctor f, Id)

Implementation of a data parallel computation with parallelism specified at launch time by a range<>.

- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for (range< Dimensions > r, ParallelForFunctor f, item< Dimensions >)

  Implementation of a data parallel computation with parallelism specified at launch time by a range<>.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for (range< Dimensions > r, ParallelForFunctor f)

  Calls the appropriate ternary parallel_for overload based on the index type of the kernel function object f.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for_global_offset (range< Dimensions > global_size, id< Dimensions > offset, ParallelForFunctor f)

  Implementation of parallel_for with a range<> and an offset.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for_workgroup (nd_range< Dimensions > r, ParallelForFunctor f)

  Implement the loop on the work-groups.
- template<int Dimensions, typename T_Item , typename ParallelForFunctor >
  void trisycl::detail::parallel_for_workitem (const group< Dimensions > &g, ParallelForFunctor f)

  Implement the loop on the work-items inside a work-group.
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for (nd_range< Dimensions > r, ParallelForFunctor f)

  Implement a variation of parallel_for to take into account a nd_range<>
- template<int Dimensions = 1, typename ParallelForFunctor >
  void trisycl::detail::parallel_for_workitem_in_group (const group< Dimensions > &g, ParallelForFunctor f)

  Implement the loop on the work-items inside a work-group.

13.153.1 Detailed Description

Implement the detail of the parallel constructions to launch kernels using Intel Threading Building Blocks (TBB).

This file gets conditionally included in "trisycl/parallelism.hpp" if TRISYCL_TBB is defined by the preprocessor.

jeffamstutz at gmail dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file parallelism_tbb.hpp.

13.154 parallelism_tbb.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_PARALLELISM_DETAIL_PARALLELISM_TBB_HPP
00002 #define TRISYCL_SYCL_PARALLELISM_DETAIL_PARALLELISM_TBB_HPP
00003
00004 /** \file
00005
00006 Implement the detail of the parallel constructions to launch kernels using
00007 Intel Threading Building Blocks (TBB). This file gets conditionally included
00008 in "trisycl/parallelism.hpp" if TRISYCL_TBB is defined by the preprocessor.
00009
00010 jeffamstutz at gmail dot com
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include "triSYCL/group.hpp"
00017 #include "triSYCL/h_item.hpp"
00018 #include "triSYCL/id.hpp"
00019 #include "triSYCL/item.hpp"
```

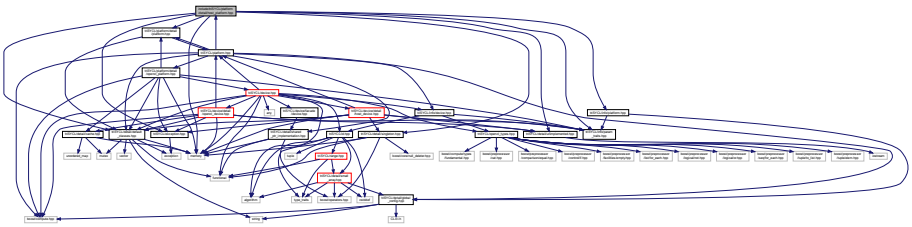
```
00020 #include "triSYCL/nd_item.hpp"
00021 #include "triSYCL/nd_range.hpp"
00022 #include "triSYCL/range.hpp"
00023
00024 #include <tbb/blocked_range2d.h>
00025 #include <tbb/blocked_range3d.h>
00026 #include <tbb/parallel_for.h>
00027
00028 /** \addtogroup parallelism
00029 @{
00030 */
00031
00032 namespace trisycl::detail {
00033
00034 static inline auto to_tbb_range(const range<1> &r)
00035 {
00036   return tbb::blocked_range<size_t>(0, r[0]);
00037 }
00038
00039 static inline auto to_tbb_range(const range<2> &r)
00040 {
00041   return tbb::blocked_range2d<size_t>(0, r[0], 0, r[1]);
00042 }
00043
00044 static inline auto to_tbb_range(const range<3> &r)
00045 {
00046   return tbb::blocked_range3d<size_t>(0, r[0], 0, r[1], 0, r[2]);
00047 }
00048
00049 static inline auto to_sycl_id(const tbb::blocked_range<size_t> &in)
00050 {
00051   return make_id(in.begin());
00052 }
00053
00054 static inline auto to_sycl_id(const tbb::blocked_range2d<size_t> &in)
00055 {
00056   const auto cols = in.cols().begin();
00057   const auto rows = in.rows().begin();
00058   return make_id(rows, cols);
00059 }
00060
00061 static inline auto to_sycl_id(const tbb::blocked_range3d<size_t> &in)
00062 {
00063   const auto pages = in.pages().begin();
00064   const auto cols = in.cols().begin();
00065   const auto rows = in.rows().begin();
00066   return make_id(rows, cols, pages);
00067 }
00068
00069 template <typename Range, typename ParallelForFunctor>
00070 void parallel_for_iterate(Range r, ParallelForFunctor &f)
00071 {
00072   tbb::parallel_for(to_tbb_range(r),
00073                     [&](auto idx) { f(to_sycl_id(idx)); },
00074                     tbb::simple_partitioner{});
00075 }
00076
00077 /** Implementation of a data parallel computation with parallelism
00078 specified at launch time by a range<>.  Kernel index is id or int.
00079 */
00080 template <int Dimensions = 1, typename ParallelForFunctor, typename Id>
00081 void parallel_for(range<Dimensions> r, ParallelForFunctor f, Id)
00082 {
00083   parallel_for_iterate(r, f);
00084 }
00085
00086 /** Implementation of a data parallel computation with parallelism
00087 specified at launch time by a range<>.  Kernel index is item.
00088 */
00089 template <int Dimensions = 1, typename ParallelForFunctor>
00090 void parallel_for(range<Dimensions> r,
00091                   ParallelForFunctor f,
00092                   item<Dimensions>)
00093 {
00094   auto reconstruct_item = [&](id<Dimensions> l) {
00095     item<Dimensions> index{r, l};
00096     f(index);
00097   };
00098
00099   parallel_for_iterate(r, reconstruct_item);
00100 }
00101
00102 /** Calls the appropriate ternary parallel_for overload based on the
00103 index type of the kernel function object f
00104 */
00105 template <int Dimensions = 1, typename ParallelForFunctor>
00106 void parallel_for(range<Dimensions> r, ParallelForFunctor f)
```

```
00107 {
00108   using mf_t = decltype(std::mem_fn(&ParallelForFunctor::operator()));
00109   using arg_t = typename mf_t::second_argument_type;
00110   parallel_for(r, f, arg_t{});
00111 }
00112
00113 /// Implementation of parallel_for with a range<> and an offset
00114 template <int Dimensions = 1, typename ParallelForFunctor>
00115 void parallel_for_global_offset(range<Dimensions> global_size,
00116                                 id<Dimensions> offset,
00117                                 ParallelForFunctor f)
00118 {
00119   auto reconstruct_item = [&](id<Dimensions> l) {
00120     item<Dimensions> index{global_size, l + offset, offset};
00121     f(index);
00122   };
00123
00124   parallel_for(global_size, reconstruct_item);
00125 }
00126
00127 /// Implement the loop on the work-groups
00128 template <int Dimensions = 1, typename ParallelForFunctor>
00129 void parallel_for_workgroup(nd_range<Dimensions> r, ParallelForFunctor f)
00130 {
00131   auto reconstruct_group = [&](id<Dimensions> l) {
00132     group<Dimensions> group{l, r};
00133     f(group);
00134   };
00135
00136   parallel_for_iterate(r.get_group_range(), reconstruct_group);
00137 }
00138
00139 /// Implement the loop on the work-items inside a work-group
00140 template <int Dimensions, typename T_Item, typename ParallelForFunctor>
00141 void parallel_for_workitem(const group<Dimensions> &g,
00142                            ParallelForFunctor f)
00143 {
00144   auto reconstruct_item = [&](id<Dimensions> local) {
00145     T_Item index{g.get_nd_range()};
00146     index.set_local(local);
00147     index.set_global(local +
00148                      id<Dimensions>(g.get_local_range()) * g.get_id());
00149     f(index);
00150   };
00151
00152   parallel_for_iterate(g.get_local_range(), reconstruct_item);
00153 }
00154
00155 /// Implement a variation of parallel_for to take into account a nd_range<>
00156 template <int Dimensions = 1, typename ParallelForFunctor>
00157 void parallel_for(nd_range<Dimensions> r, ParallelForFunctor f)
00158 {
00159   auto iterate_in_work_group = [&](id<Dimensions> g) {
00160     trisycl::group<Dimensions> wg{g, r};
00161     parallel_for_workitem<Dimensions, nd_item<Dimensions>, decltype(f)>(
00162         wg, f);
00163   };
00164
00165   parallel_for_iterate(r.get_group_range(), iterate_in_work_group);
00166 }
00167
00168 /// Implement the loop on the work-items inside a work-group
00169 template <int Dimensions, typename ParallelForFunctor>
00170 void parallel_for_workitem_in_group(const group<Dimensions> &g,
00171                                     ParallelForFunctor f)
00172 {
00173   parallel_for_workitem<Dimensions,
00174                         h_item<Dimensions>,
00175                         ParallelForFunctor>(g, f);
00176 }
00177
00178 /// @} End the parallelism Doxygen group
00179
00180 } // namespace trisycl::detail
00181
00182 /*
00183 # Some Emacs stuff:
00184 ### Local Variables:
00185 ### ispell-local-dictionary:  "american"
00186 ### eval:  (flyspell-prog-mode)
00187 ### End:
00188 */
00189
00190 #endif // TRISYCL_SYCL_PARALLELISM_DETAIL_PARALLELISM_TBB_HPP
```

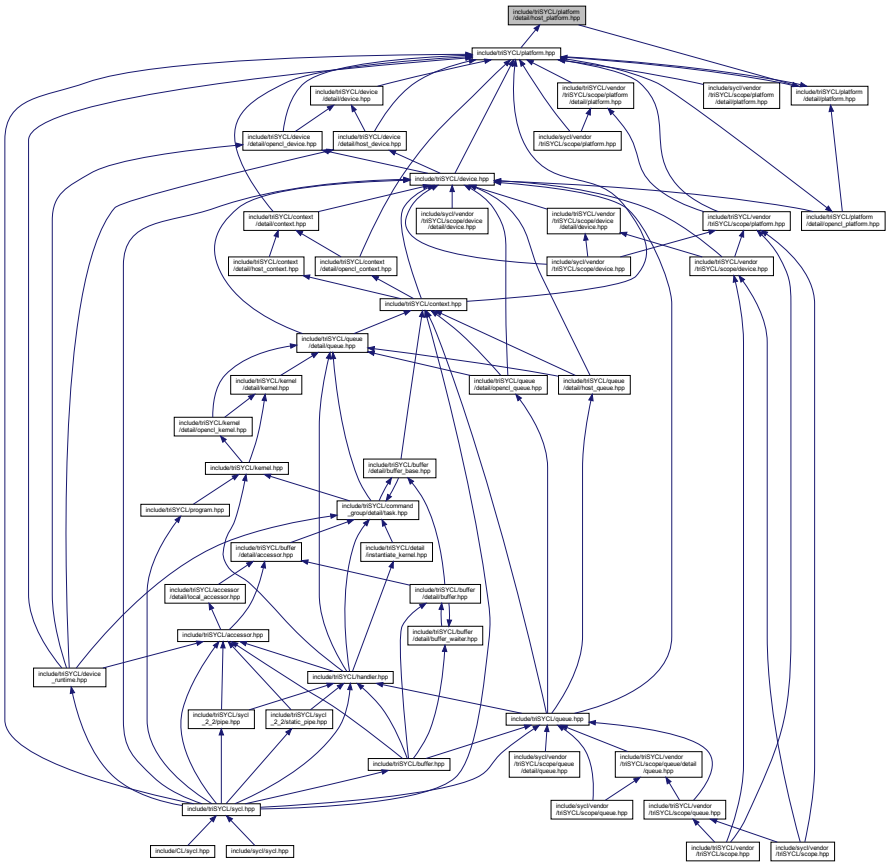
13.155 include/triSYCL/platform/detail/host_platform.hpp File Reference

```
#include <memory>
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/detail/singleton.hpp"
#include "triSYCL/detail/unimplemented.hpp"
#include "triSYCL/exception.hpp"
#include "triSYCL/info/param_traits.hpp"
#include "triSYCL/info/platform.hpp"
#include "triSYCL/platform/detail/platform.hpp"
```

Include dependency graph for host_platform.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::host_platform

  SYCL host platform. *More...*

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::extension
- namespace trisycl::extension::ce
- namespace trisycl::detail

  Some global triSYCL configuration.

13.156 host_platform.hpp

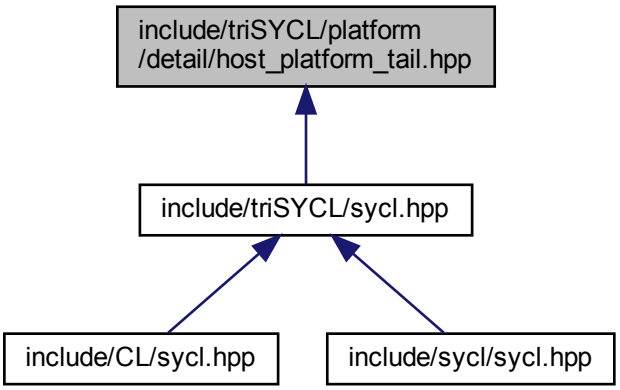
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_PLATFORM_DETAIL_HOST_PLATFORM_HPP
00002 #define TRISYCL_SYCL_PLATFORM_DETAIL_HOST_PLATFORM_HPP
00003
00004 /** \file The OpenCL triSYCL host platform implementation
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011 #include <memory>
00012
00013 #include "triSYCL/detail/default_classes.hpp"
00014
00015 #include "triSYCL/detail/singleton.hpp"
00016 #include "triSYCL/detail/unimplemented.hpp"
00017 #include "triSYCL/exception.hpp"
00018 #include "triSYCL/info/param_traits.hpp"
00019 #include "triSYCL/info/platform.hpp"
00020 #include "triSYCL/platform/detail/platform.hpp"
00021
00022 namespace trisycl {
00023
00024 namespace extension::ce {
00025   class platform;
00026 }
00027
00028 namespace detail {
00029
00030 /** \addtogroup execution Platforms, contexts, devices and queues
00031 @{
00032 */
00033
00034 /// SYCL host platform
00035 class host_platform :  public detail::platform,
00036                        public detail::singleton<host_platform> {
00037
00038 friend extension::ce::platform;
00039
00040 // \todo Have this compatible with has_extension
00041 vector_class<string_class> platform_extensions { "Xilinx_blocking_pipes" };
00042
00043 public:
00044
00045 #ifdef TRISYCL_OPENCL
00046 /** Return the cl_platform_id of the underlying OpenCL platform
00047
00048 This throws an error since there is no OpenCL platform associated
00049 to the host platform.
00050 */
00051   cl_platform_id get()const override {
00052     throw non_cl_error("The host platform has no OpenCL platform");
00053   }
00054
00055
00056 /** Return the underlying Boost.Compute platform
00057
00058 This throws an error since there is no Boost Compute platform associated
00059 to the host platform.
00060 */
00061   boost::compute::platform &get_boost_compute()const override {
00062     throw
00063       non_cl_error("The host device has no underlying Boost Compute platform");
00064   }
00065 #endif
```

```
00066
00067
00068 /// Return true since this platform is the SYCL host platform
00069   bool is_host()const override {
00070     return true;
00071   }
00072
00073
00074 /** Returning the information parameters for the host platform
00075 implementation
00076 */
00077   string_class get_info_string(info::platform param)const override {
00078     switch (param) {
00079     case info::platform::profile:
00080       /* Well...  Is the host platform really a full profile whereas it
00081 is not really OpenCL? */
00082       return "FULL_PROFILE";
00083
00084     case info::platform::version:
00085       // \todo I guess it should include the software version too...
00086       return "2.2";
00087
00088     case info::platform::name:
00089       return "triSYCL host platform";
00090
00091     case info::platform::vendor:
00092       return "triSYCL Open Source project";
00093
00094     default:
00095       // \todo Define some SYCL exception type for this type of errors
00096       throw std::invalid_argument {
00097         "Unknown parameter value for SYCL platform information" };
00098     }
00099   }
00100
00101   vector_class<string_class> get_extension_strings(void)const override {
00102     return platform_extensions;
00103   }
00104
00105
00106 /** Specify whether a specific extension is supported on the platform
00107
00108 \todo To be implemented
00109 */
00110   bool has_extension(const string_class &extension)const override {
00111     return std::find(platform_extensions.begin(), platform_extensions.end(), extension) !=
00112            platform_extensions.end();
00113   }
00114
00115 /** Get all the available devices for the host platform
00116
00117 \param[in] device_type is the device type to filter the selection
00118 or \c info::device_type::all by default to return all the
00119 devices
00120
00121 \return the device list
00122 */
00123   vector_class<trisycl::device>
00124   get_devices(const device_selector &device_selector) const override;
00125
00126 };
00127
00128 /// @} to end the execution Doxygen group
00129
00130 }
00131 }
00132
00133 /*
00134 # Some Emacs stuff:
00135 ### Local Variables:
00136 ### ispell-local-dictionary:  "american"
00137 ### eval:  (flyspell-prog-mode)
00138 ### End:
00139 */
00140
00141 #endif // TRISYCL_SYCL_PLATFORM_DETAIL_HOST_PLATFORM_HPP
```

13.157 include/triSYCL/platform/detail/host_platform_tail.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.158 host_platform_tail.hpp

Go to the documentation of this file.

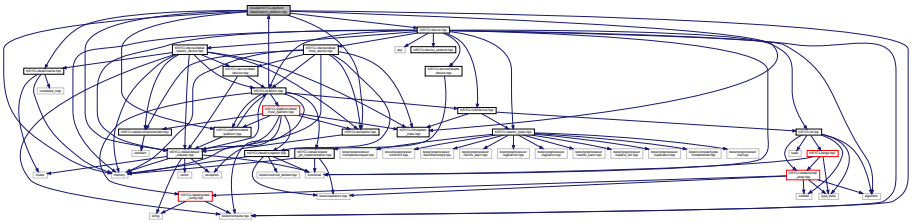
```
00001 #ifndef TRISYCL_SYCL_PLATFORM_DETAIL_HOST_PLATFORM_TAIL_HPP
00002 #define TRISYCL_SYCL_PLATFORM_DETAIL_HOST_PLATFORM_TAIL_HPP
00003
00004 /** \file The ending part of the SYCL host platform
00005
00006 This is here to break a dependency between platform and device
00007
00008 a-doumoulakis at gmail dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 namespace trisycl::detail {
00015
00016 /** \addtogroup execution Platforms, contexts, devices and queues
00017 @{
00018 */
00019
00020 /** Get all the available devices for this platform
00021
00022 \param[in] device_type is the device type to filter the selection
00023 or \c info::device_type::all by default to return all the
00024 devices
00025
```

```
00026 \return the device list
00027 */
00028 vector_class<::trisycl::device>
00029 inline host_platform::get_devices(const device_selector &device_selector)const {
00030 /** If \c get_devices is called with the host platform
00031 and the right device type, returns the host_device.
00032 */
00033   if (device_selector(::trisycl::device {}) > 0)
00034     // Return 1 default device, i.e.  the host device
00035     return { {} };
00036   else
00037     // No matching device
00038     return {};
00039 }
00040
00041 /// @} to end the Doxygen group
00042
00043 }
00044
00045 /*
00046 # Some Emacs stuff:
00047 ### Local Variables:
00048 ### ispell-local-dictionary:  "american"
00049 ### eval:  (flyspell-prog-mode)
00050 ### End:
00051 */
00052
00053 #endif // TRISYCL_SYCL_PLATFORM_DETAIL_HOST_PLATFORM_TAIL_HPP
```

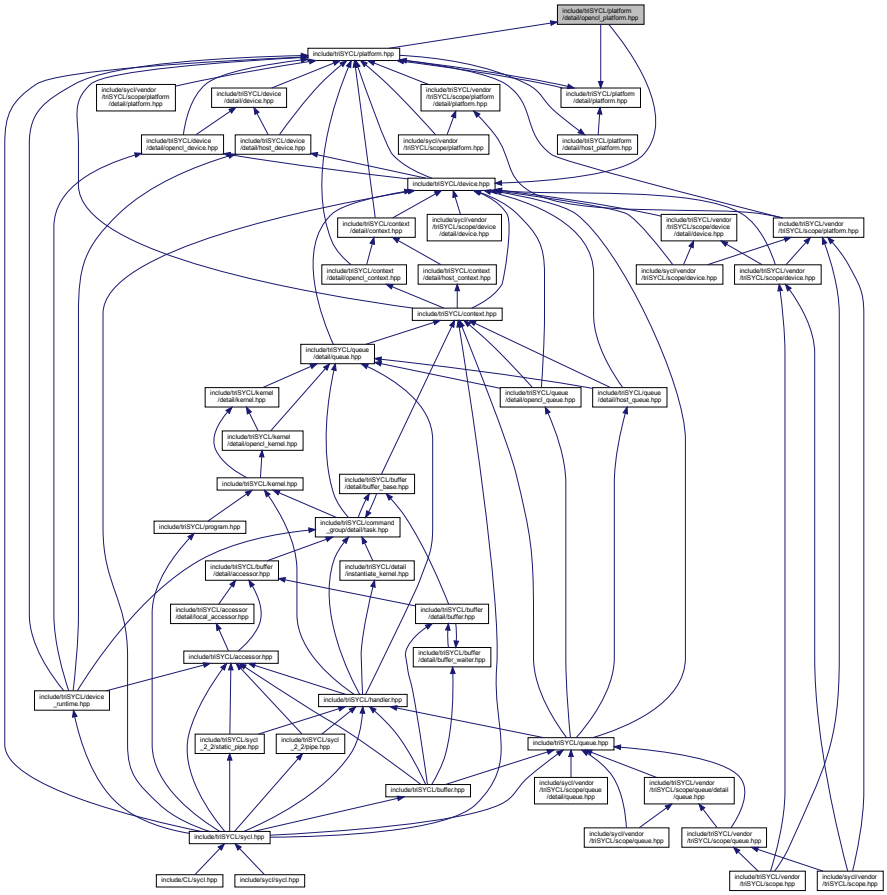
13.159 include/triSYCL/platform/detail/opencl_platform.hpp File Reference

```
#include <memory>
#include <boost/compute.hpp>
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/detail/cache.hpp"
#include "triSYCL/device.hpp"
#include "triSYCL/exception.hpp"
#include "triSYCL/info/param_traits.hpp"
#include "triSYCL/platform/detail/platform.hpp"
```

Include dependency graph for opencl_platform.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::opencl_platform

  SYCL OpenCL platform. *More...*

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.160 opencl_platform.hpp

Go to the documentation of this file.

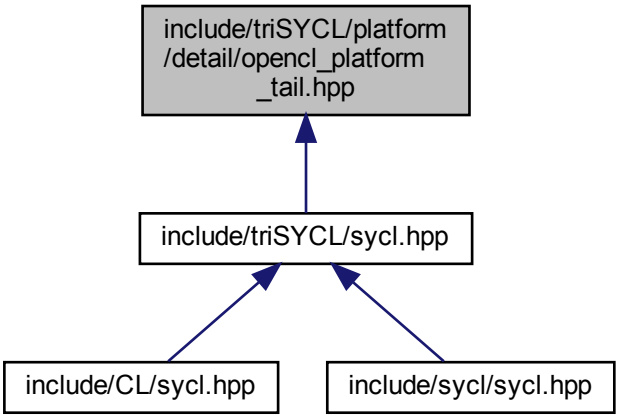
```
00001 #ifndef TRISYCL_SYCL_PLATFORM_DETAIL_OPENCL_PLATFORM_HPP
00002 #define TRISYCL_SYCL_PLATFORM_DETAIL_OPENCL_PLATFORM_HPP
00003
00004 /** \file The OpenCL triSYCL OpenCL platform implementation
00005
00006 Ronan at Keryell point FR
```

```
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011 #include <memory>
00012
00013 #include <boost/compute.hpp>
00014
00015 #include "triSYCL/detail/default_classes.hpp"
00016
00017 #include "triSYCL/detail/cache.hpp"
00018 #include "triSYCL/device.hpp"
00019 #include "triSYCL/exception.hpp"
00020 #include "triSYCL/info/param_traits.hpp"
00021 #include "triSYCL/platform/detail/platform.hpp"
00022
00023 namespace trisycl {
00024
00025 class device;
00026
00027 namespace detail {
00028
00029 /** \addtogroup execution Platforms, contexts, devices and queues
00030 @{
00031 */
00032
00033 /// SYCL OpenCL platform
00034 class opencl_platform :  public detail::platform {
00035
00036 /// Use the Boost Compute abstraction of the OpenCL platform
00037   boost::compute::platform p;
00038
00039 /** A cache to always return the same live platform for a given OpenCL
00040 platform
00041
00042 C++11 guaranties the static construction is thread-safe
00043 */
00044   static detail::cache<cl_platform_id, detail::opencl_platform> cache;
00045
00046 public:
00047
00048 /// Return the cl_platform_id of the underlying OpenCL platform
00049   cl_platform_id get()const override {
00050     return p.id();
00051   }
00052
00053
00054 /// Return the underlying Boost.Compute platform
00055   const boost::compute::platform &get_boost_compute()const override {
00056     return p;
00057   }
00058
00059
00060 /// Return false since an OpenCL platform is not the SYCL host platform
00061   bool is_host()const override {
00062     return false;
00063   }
00064
00065
00066 /// Returning the information string parameters for the OpenCL platform
00067   string_class get_info_string(info::platform param)const override {
00068     /* Use the fact that the triSYCL info values are the same as the
00069 OpenCL ones used in Boost.Compute to just cast the enum class
00070 to the int value */
00071     return p.get_info<std::string>(static_cast<cl_platform_info>(param));
00072   }
00073
00074   vector_class<string_class> get_extension_strings()const override {
00075     return p.extensions();
00076   }
00077
00078
00079 /// Specify whether a specific extension is supported on the platform
00080   bool has_extension(const string_class &extension)const override {
00081     return p.supports_extension(extension);
00082   }
00083
00084
00085 ///// Get a singleton instance of the opencl_platform
00086   static std::shared_ptr<opencl_platform>
00087   instance(const boost::compute::platform &p) {
00088     return cache.get_or_register(p.id(),
00089                                  [&] { return new opencl_platform { p }; });
00090   }
00091
00092
00093 /** Get all the available devices for this OpenCL platform
```

```
00094
00095 \param[in] device_type is the device type to filter the selection
00096 or \c info::device_type::all by default to return all the
00097 devices
00098
00099 \return the device list
00100 */
00101   vector_class<trisycl::device>
00102   get_devices(const device_selector &device_selector) const override;
00103
00104 private:
00105
00106 /// Only the instance factory can built it
00107   opencl_platform(const boost::compute::platform &p) :  p { p } {}
00108
00109 public:
00110
00111 /// Unregister from the cache on destruction
00112   ~opencl_platform()override {
00113     cache.remove(p.id());
00114   }
00115
00116 };
00117
00118 /* Allocate the cache here but since this is a pure-header library,
00119 use a weak symbol so that only one remains when SYCL headers are
00120 used in different compilation units of a program
00121 */
00122 TRISYCL_WEAK_ATTRIB_PREFIX
00123 detail::cache<cl_platform_id, detail::opencl_platform> opencl_platform::cache
00124 TRISYCL_WEAK_ATTRIB_SUFFIX;
00125
00126 /// @} to end the execution Doxygen group
00127
00128 }
00129 }
00130
00131 /*
00132 # Some Emacs stuff:
00133 ### Local Variables:
00134 ### ispell-local-dictionary:  "american"
00135 ### eval:  (flyspell-prog-mode)
00136 ### End:
00137 */
00138
00139 #endif // TRISYCL_SYCL_PLATFORM_DETAIL_HOST_PLATFORM_HPP
```

13.161 include/triSYCL/platform/detail/opencl_platform_tail.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.162 opencl_platform_tail.hpp

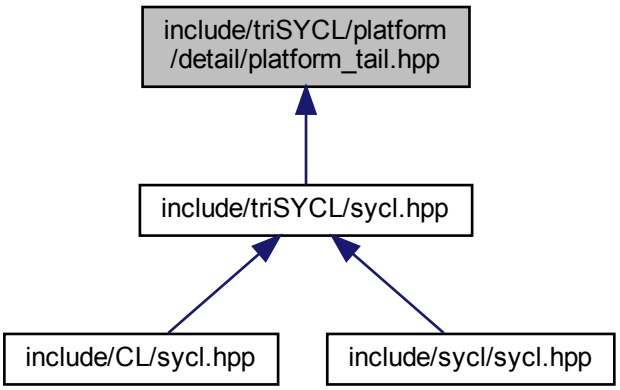
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_PLATFORM_DETAIL_OPENCL_PLATFORM_TAIL_HPP
00002 #define TRISYCL_SYCL_PLATFORM_DETAIL_OPENCL_PLATFORM_TAIL_HPP
00003
00004 /** \file The ending part of the SYCL host platform
00005
00006 This is here to break a dependency between platform and device
00007
00008 a-doumoulakis at gmail dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 namespace trisycl::detail {
00015
00016 /** \addtogroup execution Platforms, contexts, devices and queues
00017 @{
00018 */
00019
00020 /** Returns a vector class containing all SYCL devices associated with
00021 this OpenCL platform
00022
00023 \param[in] device_type is the device type to filter the selection
00024 or \c info::device_type::all by default to return all the
```

```
00025 devices
00026
00027 \return the device list
00028 */
00029 vector_class<::trisycl::device> inline
00030 opencl_platform::get_devices(const device_selector &device_selector)const {
00031   vector_class<::trisycl::device> devices;
00032   // Add the desired OpenCL devices
00033   for (const auto &d :  get_boost_compute().devices()) {
00034     // Get the SYCL device from the Boost Compute device
00035     ::trisycl::device sycl_dev { d };
00036     /* Return the devices with the good criterion according to the selector.
00037 By calling devices on the \c boost::compute::platform we know that
00038 we iterate only over the device belonging to the current platform,
00039 */
00040     if (device_selector(sycl_dev) > 0)
00041       devices.push_back(sycl_dev);
00042   }
00043
00044   return devices;
00045 }
00046
00047 /// @} to end the Doxygen group
00048
00049 }
00050
00051 /*
00052 # Some Emacs stuff:
00053 ### Local Variables:
00054 ### ispell-local-dictionary:  "american"
00055 ### eval:  (flyspell-prog-mode)
00056 ### End:
00057 */
00058
00059 #endif // TRISYCL_SYCL_PLATFORM_DETAIL_OPENCL_PLATFORM_TAIL_HPP
```

13.163 include/triSYCL/platform/detail/platform_tail.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

13.164 platform_tail.hpp

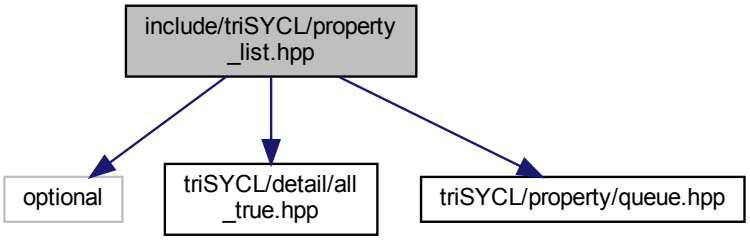
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_PLATFORM_TAIL_HPP
00002 #define TRISYCL_SYCL_PLATFORM_TAIL_HPP
00003
00004 /** \file The ending part of the OpenCL SYCL platform
00005
00006 This is here to break a dependence between platform and device_selector.
00007 This file is distributed under the University of Illinois Open Source
00008 License.  See LICENSE.TXT for details.
00009 */
00010
00011 namespace trisycl {
00012
00013 inline
00014 platform::platform(const device_selector &device_selector) {
00015   platform host_platform = {};
00016 #ifdef TRISYCL_OPENCL
00017   if (host_platform.implementation->get_devices(device_selector).empty()) {
00018     for (const auto &d :  boost::compute::system::platforms()) {
00019       auto clplatform = ::trisycl::platform { d };
00020       auto devices = clplatform.implementation->get_devices(device_selector);
00021       if (!devices.empty()) {
00022         *this = std::move(clplatform);
00023         return;
00024       }
00025     }
00026   }
00027 #endif
00028   *this = std::move(host_platform);
00029 }
00030
00031 inline vector_class<device>
00032 platform::get_devices(info::device_type device_type)const {
00033   return implementation->get_devices(device_type_selector { device_type });
00034 }
00035
00036 }
00037
00038 #endif
```

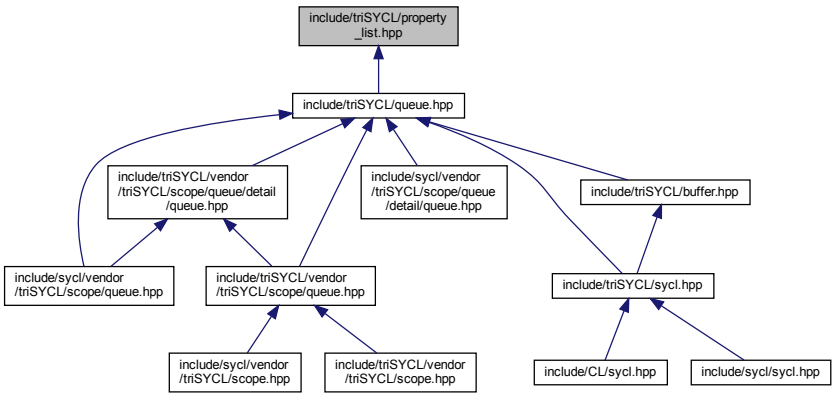
13.165 include/triSYCL/property_list.hpp File Reference

```
#include <optional>
#include "triSYCL/detail/all_true.hpp"
#include "triSYCL/property/queue.hpp"
```

Include dependency graph for property_list.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::property_list

  SYCL property list.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

Macros

- #define TRISYCL_PROPERTY_CREATE(type, prop_name)
- #define TRISYCL_PROPERTY_HAS_GET(type, prop_name)

13.165.1 Macro Definition Documentation

13.165.1.1 TRISYCL_PROPERTY_CREATE

```
#define TRISYCL_PROPERTY_CREATE(
              type,
              prop_name )
```

Value:

```
  std::optional<property::type::prop_name> prop_name;                  \
  void addproperty(property::type::prop_name prop) { prop_name = prop; }
```

Definition at line 11 of file property_list.hpp.

13.165.1.2 TRISYCL_PROPERTY_HAS_GET

```
#define TRISYCL_PROPERTY_HAS_GET(
              type,
              prop_name )
```

Value:

```
  template<>                                                                \
  inline bool                                                               \
  property_list::has_property<property::type::prop_name>()const {      \
    return prop_name.has_value();                                           \
  }                                                                         \
  template<>                                                                \
  inline trisycl::property::type::prop_name                                 \
  property_list::get_property<property::type::prop_name>()const {      \
    return prop_name.value();                                               \
  }
```

Definition at line 53 of file property_list.hpp.

13.166 property_list.hpp

Go to the documentation of this file.

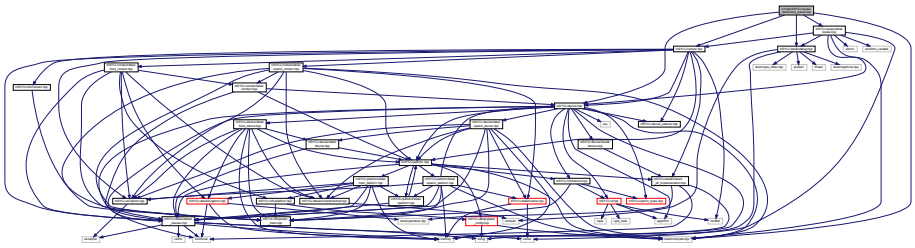
```
00001 #ifndef TRISYCL_SYCL_PROPERTY_LIST_HPP
00002 #define TRISYCL_SYCL_PROPERTY_LIST_HPP
00003
00004 #include <optional>
00005
00006 #include "triSYCL/detail/all_true.hpp"
00007 #include "triSYCL/property/queue.hpp"
00008
00009 namespace trisycl {
00010
00011 #define TRISYCL_PROPERTY_CREATE(type, prop_name)                        \
00012 std::optional<property::type::prop_name> prop_name;                   \
00013 void addproperty(property::type::prop_name prop) { prop_name = prop; }
00014
00015 /// SYCL property list
00016 class property_list {
00017   /*
00018 * The properties are stored as a std::optional per property,
00019 * and the addproperty methods init the correct one for each known
00020 * property, this method is recursive to deal with the pack parameter.
00021 */
00022   TRISYCL_PROPERTY_CREATE(queue, enable_profiling);
00023
00024 protected:
00025   template <typename propertyT>
00026   inline bool has_property() const;
00027
00028   template<typename propertyT>
00029   inline propertyT get_property() const;
00030
00031 private:
00032   /* we need an iterative function to deal with the pack parameter */
00033   template<typename T, typename...  propsT,
00034            typename = std::enable_if_t<detail::all_true<std::is_convertible<propsT,
      detail::property>::value ...>::value»
00035   void addproperty(T first, propsT...  next) {
00036     addproperty(next...);
00037   }
00038 public:
00039 /** Construct a property list from a list of classes derived from the detail::property.
00040
00041 The detail::property and all_true are required to disambiguate queue/buffer constructors, so
00042 that property_lists are only created from property classes.
00043 */
00044   template<typename...  propsT,
00045            typename = std::enable_if_t<detail::all_true<std::is_convertible<propsT,
      detail::property>::value ...>::value»
00046   property_list(propsT...  props) {
00047     addproperty(props...);
00048   }
00049
00050   property_list() {}
00051 };
```

```
00052
00053 #define TRISYCL_PROPERTY_HAS_GET(type, prop_name)                       \
00054 template<>                                                              \
00055 inline bool                                                             \
00056 property_list::has_property<property::type::prop_name>() const {       \
00057 return prop_name.has_value();                                          \
00058 }                                                                       \
00059 template<>                                                              \
00060 inline trisycl::property::type::prop_name                               \
00061 property_list::get_property<property::type::prop_name>() const {       \
00062 return prop_name.value();                                              \
00063 }
00064
00065 TRISYCL_PROPERTY_HAS_GET(queue, enable_profiling)
00066
00067 #undef TRISYCL_PROPERTY_CREATE
00068 #undef TRISYCL_PROPERTY_HAS_GET
00069
00070 }
00071 #endif
```

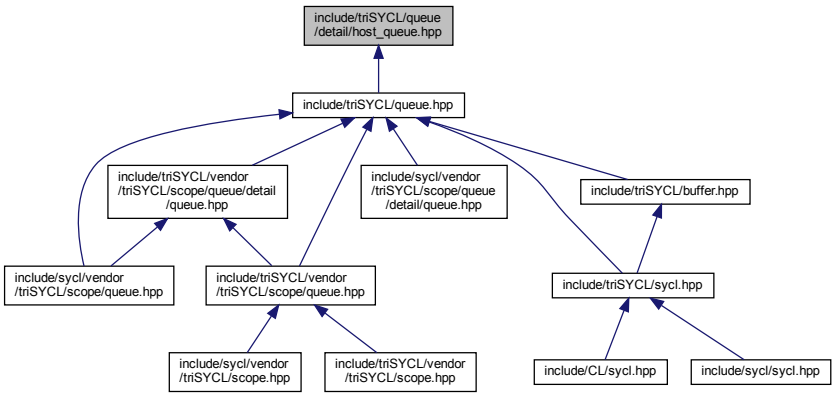
13.167 include/triSYCL/queue/detail/host_queue.hpp File Reference

```
#include <boost/compute.hpp>
#include "triSYCL/context.hpp"
#include "triSYCL/detail/debug.hpp"
#include "triSYCL/device.hpp"
#include "triSYCL/queue/detail/queue.hpp"
```

Include dependency graph for host_queue.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::host_queue

  Some implementation details about the SYCL queue.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.168 host_queue.hpp

Go to the documentation of this file.

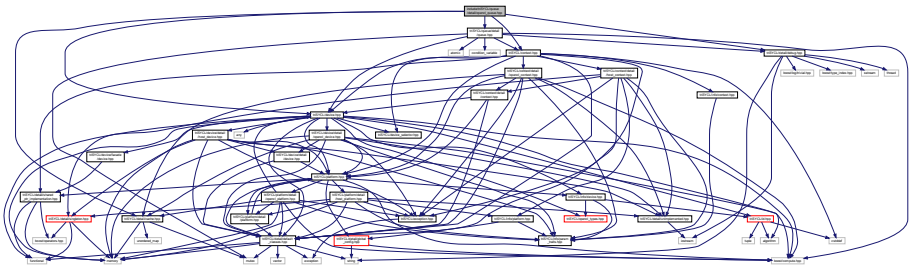
```
00001 #ifndef TRISYCL_SYCL_QUEUE_DETAIL_HOST_QUEUE_HPP
00002 #define TRISYCL_SYCL_QUEUE_DETAIL_HOST_QUEUE_HPP
00003
00004 /** \file Some implementation details of the host queue
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #ifdef TRISYCL_OPENCL
00013 #include <boost/compute.hpp>
00014 #endif
00015
00016 #include "triSYCL/context.hpp"
00017 #include "triSYCL/detail/debug.hpp"
00018 #include "triSYCL/device.hpp"
00019 #include "triSYCL/queue/detail/queue.hpp"
00020
00021 namespace trisycl::detail {
00022
00023 /** Some implementation details about the SYCL queue
00024
00025 Note that a host queue is not a singleton, compared to host
00026 device or host platform, for example.
00027 */
00028 class host_queue :  public detail::queue,
00029                     detail::debug<host_queue> {
00030
00031 #ifdef TRISYCL_OPENCL
00032 /** Return the cl_command_queue of the underlying OpenCL queue
00033
00034 This throws an error since there is no OpenCL queue associated
00035 to the host queue.
00036 */
00037   cl_command_queue get()const override {
00038     throw non_cl_error("The host queue has no OpenCL command queue");
00039   }
00040
00041
00042 /** Return the underlying Boost.Compute command queue
00043
00044 This throws an error since there is no OpenCL queue associated
00045 to the host queue.
00046 */
00047   boost::compute::command_queue &get_boost_compute()override {
00048     throw non_cl_error("The host queue has no OpenCL command queue");
00049   }
00050 #endif
00051
00052
00053 /// Return the SYCL host queue's host context
00054   trisycl::context get_context()const override {
00055     // Return the default context for the default device
00056     // which is the host context
00057     return { get_device() };
00058   }
```

```
00059
00060
00061 /// Return the SYCL host device the host queue is associated with
00062   trisycl::device get_device()const override {
00063     // Return the default device which is the host device
00064     return {};
00065   }
00066
00067
00068 /// Claim proudly that the queue is executing on the SYCL host device
00069   bool is_host()const override {
00070     return true;
00071   }
00072
00073
00074 };
00075
00076 }
00077
00078 /*
00079 # Some Emacs stuff:
00080 ### Local Variables:
00081 ### ispell-local-dictionary:  "american"
00082 ### eval:  (flyspell-prog-mode)
00083 ### End:
00084 */
00085
00086 #endif // TRISYCL_SYCL_QUEUE_DETAIL_HOST_QUEUE_HPP
```

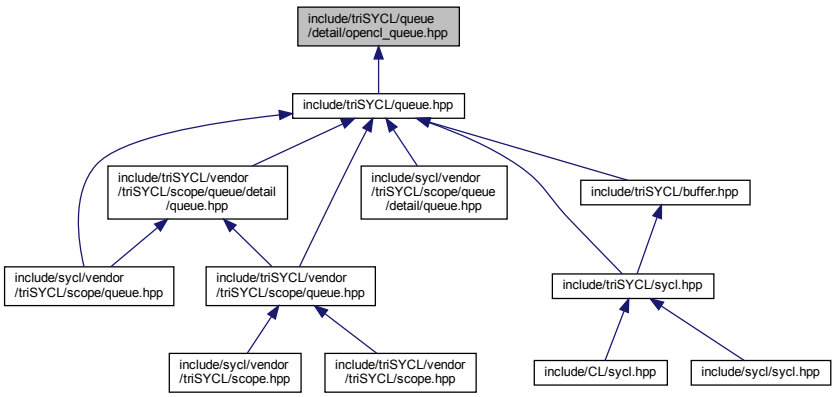
13.169 include/triSYCL/queue/detail/opencl_queue.hpp File Reference

```
#include "triSYCL/context.hpp"
#include "triSYCL/detail/cache.hpp"
#include "triSYCL/detail/debug.hpp"
#include "triSYCL/device.hpp"
#include "triSYCL/queue/detail/queue.hpp"
```

Include dependency graph for opencl_queue.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::opencl_queue

  Some implementation details about the SYCL queue.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.170 opencl_queue.hpp

Go to the documentation of this file.

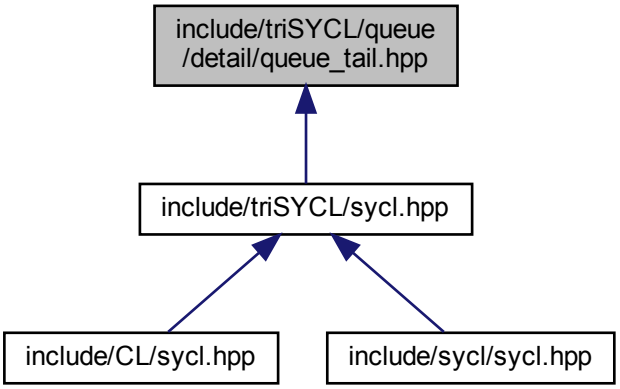
```
00001 #ifndef TRISYCL_SYCL_QUEUE_DETAIL_OPENCL_QUEUE_HPP
00002 #define TRISYCL_SYCL_QUEUE_DETAIL_OPENCL_QUEUE_HPP
00003
00004 /** \file Some implementation details of the OpenCL queue
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include "triSYCL/context.hpp"
00013 #include "triSYCL/detail/cache.hpp"
00014 #include "triSYCL/detail/debug.hpp"
00015 #include "triSYCL/device.hpp"
00016 #include "triSYCL/queue/detail/queue.hpp"
00017
00018 namespace trisycl::detail {
00019
00020 /// Some implementation details about the SYCL queue
00021 class opencl_queue :  public detail::queue,
00022                       detail::debug<opencl_queue> {
00023 /// Use the Boost Compute abstraction of the OpenCL command queue
00024   boost::compute::command_queue q;
00025
00026 /** A cache to always return the same alive queue for a given OpenCL
00027 command queue
```

```
00028
00029 C++11 guaranties the static construction is thread-safe
00030 */
00031   static detail::cache<cl_command_queue, detail::opencl_queue> cache;
00032
00033 /// Return the cl_command_queue of the underlying OpenCL queue
00034   cl_command_queue get()const override {
00035     return q.get();
00036   }
00037
00038
00039 /// Return the underlying Boost.Compute command queue
00040   boost::compute::command_queue &get_boost_compute()override {
00041     return q;
00042   }
00043
00044
00045 /// Return the SYCL context associated to the queue
00046   trisycl::context get_context()const override {
00047     return q.get_context();
00048   }
00049
00050
00051 /// Return the SYCL device associated to the queue
00052   trisycl::device get_device()const override {
00053     return q.get_device();
00054   }
00055
00056
00057 /// Claim proudly that an OpenCL queue cannot be the SYCL host queue
00058   bool is_host()const override {
00059     return false;
00060   }
00061
00062 private:
00063
00064 /// Only the instance factory can built it
00065   opencl_queue(const boost::compute::command_queue &q) :  q { q } {}
00066
00067 public:
00068
00069 /// Get a singleton instance of the opencl_queue
00070   static std::shared_ptr<opencl_queue>
00071   instance(const boost::compute::command_queue &q) {
00072     return cache.get_or_register(q.get(),
00073                                  [&] { return new opencl_queue { q }; });
00074   }
00075
00076
00077 /** Create a new queue associated to this device
00078
00079 \todo Check with SYCL committee what is the expected behaviour
00080 here about the context.  Is this a new context everytime, or
00081 always the same for a given device?
00082 */
00083   static std::shared_ptr<detail::queue>
00084   instance(const trisycl::device &d) {
00085     return instance (boost::compute::command_queue {
00086         // For now, create a new context every time
00087         boost::compute::context { d.get_boost_compute() },
00088         d.get_boost_compute()
00089           });
00090   }
00091
00092
00093 /// Unregister from the cache on destruction
00094   ~opencl_queue()override {
00095     cache.remove(q.get());
00096   }
00097
00098 };
00099
00100 /* Allocate the cache here but since this is a pure-header library,
00101 use a weak symbol so that only one remains when SYCL headers are
00102 used in different compilation units of a program
00103 */
00104 TRISYCL_WEAK_ATTRIB_PREFIX
00105 detail::cache<cl_command_queue, detail::opencl_queue> opencl_queue::cache
00106 TRISYCL_WEAK_ATTRIB_SUFFIX;
00107
00108 }
00109
00110 /*
00111 # Some Emacs stuff:
00112 ### Local Variables:
00113 ### ispell-local-dictionary:  "american"
00114 ### eval:  (flyspell-prog-mode)
```

```
00115 ### End:
00116 */
00117
00118 #endif // TRISYCL_SYCL_QUEUE_DETAIL_OPENCL_QUEUE_HPP
```

13.171 include/triSYCL/queue/detail/queue_tail.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

Macros

- #define TRISYCL_DEFAULT_QUEUE_SELECTOR host_selector

13.171.1 Macro Definition Documentation

13.171.1.1 TRISYCL_DEFAULT_QUEUE_SELECTOR

```
#define TRISYCL_DEFAULT_QUEUE_SELECTOR host_selector
```

Definition at line 19 of file queue_tail.hpp.

13.172 queue_tail.hpp

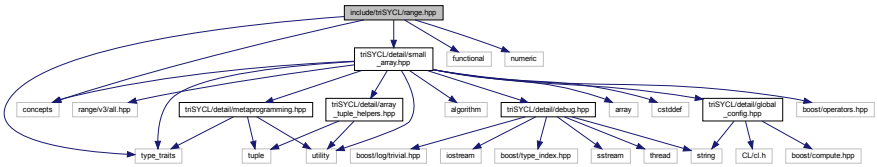
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_QUEUE_DETAIL_QUEUE_TAIL_HPP
00002 #define TRISYCL_SYCL_QUEUE_DETAIL_QUEUE_TAIL_HPP
00003
00004 /** \file The ending part of of the OpenCL SYCL context
00005
00006 This is here to break a dependence between context and device_selector
00007
00008 Dave Airlie
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013 /*
00014 * to workaround the fact we don't have a full device compiler yet, but we want
00015 * to play around with CTS tests without breaking our local development, this
00016 * default to host selector even though this is against the spec.
00017 */
00018 #ifndef TRISYCL_DEFAULT_QUEUE_SELECTOR
00019 #define TRISYCL_DEFAULT_QUEUE_SELECTOR host_selector
00020 #endif
00021
00022 namespace trisycl {
00023
00024 /* has to be inline as it references default_selector */
00025 /* this should be default selector but until we have device compiler
00026 in better shape and better queue let's just default to host selector for now */
00027 inline queue::queue(const property_list &propList) :
00028   queue { TRISYCL_DEFAULT_QUEUE_SELECTOR {}, propList } {}
00029
00030 inline queue::queue(const async_handler &asyncHandler, const property_list &propList) :
00031   queue { TRISYCL_DEFAULT_QUEUE_SELECTOR {}, asyncHandler, propList } {}
00032
00033 }
00034
00035 #endif // TRISYCL_SYCL_QUEUE_DETAIL_QUEUE_TAIL_HPP
```

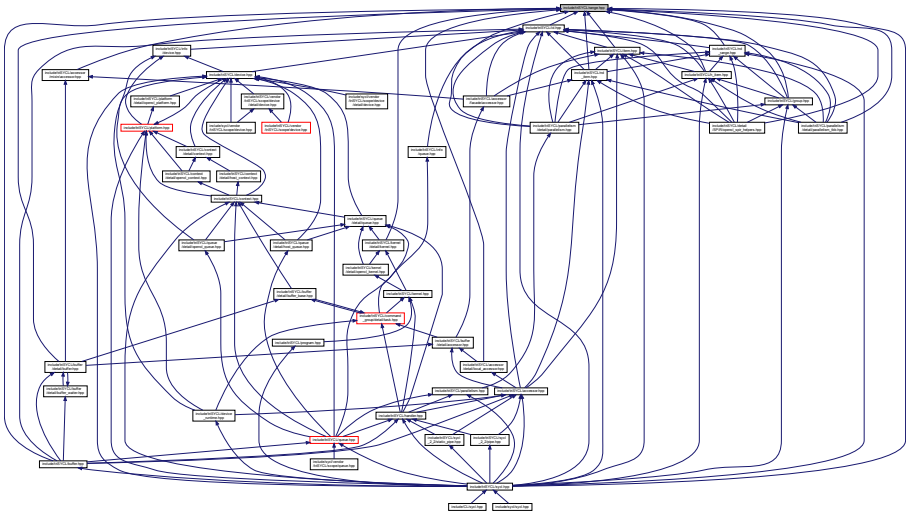
13.173 include/triSYCL/range.hpp File Reference

```
#include <concepts>
#include <functional>
#include <numeric>
#include <type_traits>
#include "triSYCL/detail/small_array.hpp"
```

Include dependency graph for range.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::range< Dimensions >

  A SYCL range defines a multi-dimensional index range that can be used to define launch parallel computation extent or buffer sizes. More...
- struct trisycl::detail::is_range< T >

  A type trait to check if a type it a sycl::range or not.
- struct trisycl::detail::is_range< range< Dimensions > >

  Return true if template instantiating match a sycl::range.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Functions

- template<typename... BasicType>
  trisycl::range (BasicType... Args) -> range< sizeof...(Args)>

  User-defined deduction guides to deduce the rank of a range from the number of arguments.
- auto trisycl::detail::rangeify (auto r)

  Make an explicit range<1> when an integral is provided or pass the value unchanged.

Variables

- template<typename T >
  constexpr auto trisycl::detail::is_range_v = is_range<T>::value

  A variable to check if a type is a sycl::range or not.

13.174 range.hpp

Go to the documentation of this file.

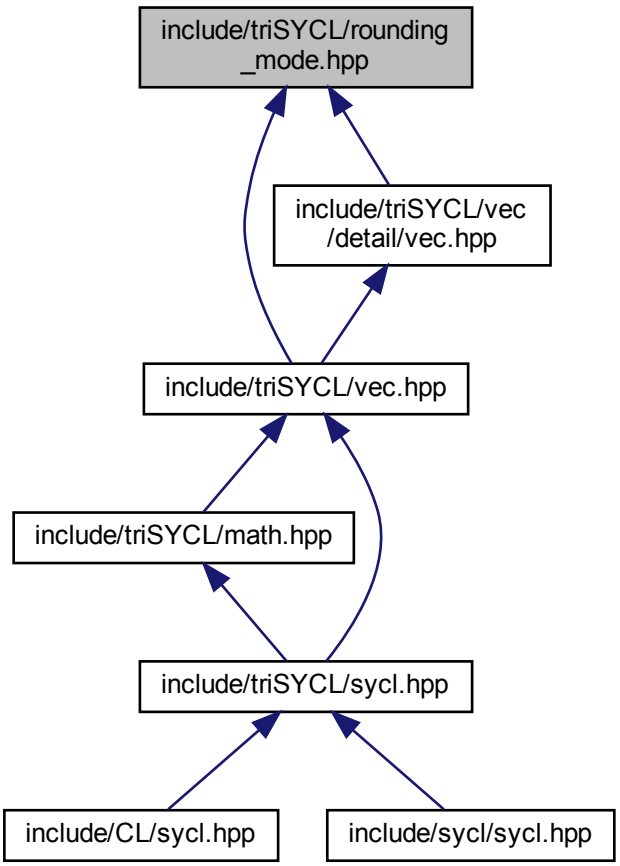
```
00001 #ifndef TRISYCL_SYCL_RANGE_HPP
00002 #define TRISYCL_SYCL_RANGE_HPP
00003
00004 /** \file The OpenCL SYCL range<>
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <concepts>
00013 #include <functional>
00014 #include <numeric>
00015 #include <type_traits>
00016 #include "triSYCL/detail/small_array.hpp"
00017
00018 namespace trisycl {
00019
00020 /** \addtogroup parallelism Expressing parallelism through kernels
00021 @{
00022 */
00023
00024 /** A SYCL range defines a multi-dimensional index range that can be used
00025 to define launch parallel computation extent or buffer sizes.
00026
00027 \todo use std::size_t Dimensions instead of int Dimensions in the
00028 specification?
00029
00030 \todo add to the specification this default parameter value?
00031
00032 \todo add to the specification some way to specify an offset?
00033 */
00034 template <int Dimensions = 1>
00035 class range
00036     :  public detail::small_array_sycl<std::size_t, range<Dimensions>,
00037                                        Dimensions> {
00038
00039  public:
00040
00041   // Inherit of all the constructors
00042   using detail::small_array_sycl<std::size_t,
00043                                  range<Dimensions>,
00044                                  Dimensions>::small_array_sycl;
00045
00046   range() = default;
00047
00048 /// The number of dimensions of the range
00049   static auto constexpr rank() { return Dimensions; }
00050
00051 /** Return the number of elements in the range
00052 */
00053   size_t size()const {
00054     // Return the product of the sizes in each dimension
00055     return std::accumulate(this->cbegin(),
00056                            this->cend(),
00057                            1,
00058                            std::multiplies<size_t> {});
00059   }
00060 };
00061
00062 /**  User-defined deduction guides to deduce the rank of a range from
00063 the number of arguments
00064
00065 For example range { 4, 5, 7 } will define an range<3> { 4, 5, 7 }
00066 */
00067 template <typename...  BasicType>
00068 range(BasicType...  Args) -> range<sizeof...(Args)>;
00069
00070 /// @} End the parallelism Doxygen group
00071
00072 namespace detail {
00073 /** Make an explicit range<1> when an integral is provided or pass
00074 the value unchanged */
00075     auto rangeify(auto r) {
00076       if constexpr (std::integral<decltype(r)>)
00077         return range { r };
00078       else
00079         return r;
00080     };
00081
00082 /// A type trait to check if a type it a sycl::range or not
```

```
00083 template <typename T> struct is_range :  std::false_type {};
00084
00085 /// Return true if template instantiating match a sycl::range
00086 template <int Dimensions>
00087 struct is_range<range<Dimensions>> :  std::true_type {};
00088
00089 /// A variable to check if a type is a sycl::range or not
00090 template <typename T> constexpr auto is_range_v = is_range<T>::value;
00091
00092 } // namespace detail
00093
00094 } // namespace trisycl
00095
00096 /*
00097 # Some Emacs stuff:
00098 ### Local Variables:
00099 ### ispell-local-dictionary:  "american"
00100 ### eval:  (flyspell-prog-mode)
00101 ### End:
00102 */
00103
00104 #endif // TRISYCL_SYCL_RANGE_HPP
```

13.175 include/triSYCL/rounding_mode.hpp File Reference

Define rounding modes used in SYCL.

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

Enumerations

- enum class trisycl::rounding_mode { trisycl::automatic , trisycl::rte , trisycl::rtz , trisycl::rtp , trisycl::rtn }

  Rounding mode for vector conversions.

13.175.1 Detailed Description

Define rounding modes used in SYCL.

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file rounding_mode.hpp.

13.176 rounding_mode.hpp

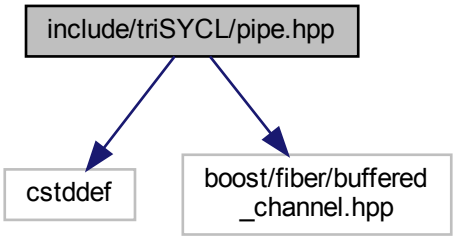
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_ROUNDING_MODE_HPP
00002 #define TRISYCL_SYCL_ROUNDING_MODE_HPP
00003
00004 /** \file
00005
00006 Define rounding modes used in SYCL
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 namespace trisycl {
00013
00014 /// Rounding mode for vector conversions
00015 enum class rounding_mode {
00016   automatic, /// Default rounding mode, rtz for integers, rte for floating-point
00017   rte,       /// Round to nearest even
00018   rtz,       /// Round towards zero
00019   rtp,       /// Round towards positive infinity
00020   rtn        /// Round towards negative infinity
00021 };
00022
00023 }
00024
00025 /*
00026 # Some Emacs stuff:
00027 ### Local Variables:
00028 ### ispell-local-dictionary:  "american"
00029 ### eval:  (flyspell-prog-mode)
00030 ### End:
00031 */
00032
00033 #endif // TRISYCL_SYCL_ROUNDING_MODE_HPP
```

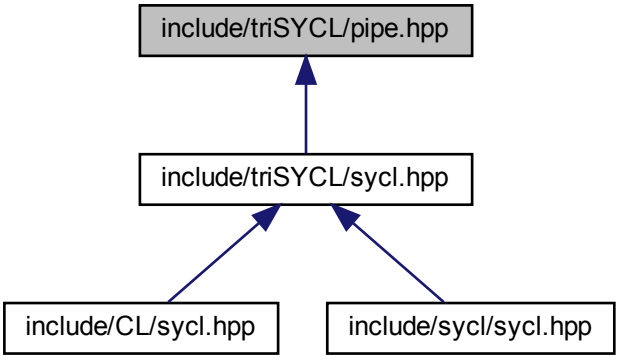
13.177 include/triSYCL/pipe.hpp File Reference

```
#include <cstddef>
#include <boost/fiber/buffered_channel.hpp>
```

Include dependency graph for pipe.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::pipe< Name, T, MinCapacity, SizeT >

  SYCL pipe object. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

13.178 pipe.hpp

Go to the documentation of this file.

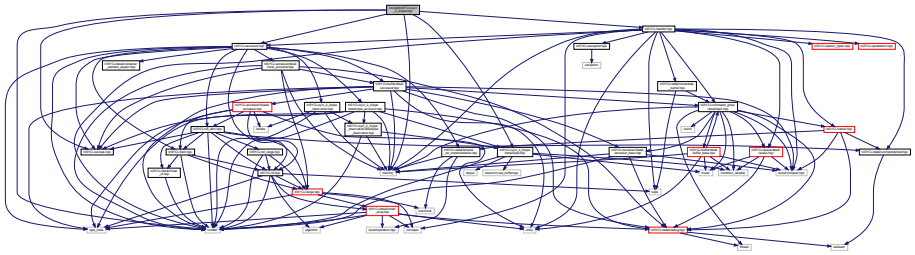
```
00001 #ifndef TRISYCL_SYCL_PIPE_HPP
00002 #define TRISYCL_SYCL_PIPE_HPP
00003
00004 /** \file The SYCL data-flow pipe extension
00005
00006 Ronan dot Keryell arobase Xilinx point com
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <cstddef>
00013
00014 #include <boost/fiber/buffered_channel.hpp>
00015
00016 namespace trisycl {
00017
00018 /** \addtogroup data Data access and storage in SYCL
00019 @{
00020 */
00021
00022 /** SYCL pipe object
00023
00024 The member functions can throw for now to help debugging or
00025 implementation on CPU.
00026
00027 This is mainly a CPU implementation of the Intel extension
00028 described in
00029
00030 - https://github.com/intel/llvm/pull/292
00031
00032 - https://github.com/intel/llvm/pull/635
00033
00034 - https://github.com/intel/llvm/issues/832
00035
00036 \todo Bike-shed the naming & API according to current C++ proposals
00037 or libraries about static sized FIFO
00038
00039 This could be implemented on top of
00040 https://xilinx.github.io/XRT/master/html/streaming_data_support.html
00041 and/or expose some features from XRT
00042 */
00043 template <typename Name,
00044           typename T,
00045           std::size_t MinCapacity = 0,
00046           typename SizeT = std::size_t>
00047 class pipe {
00048
00049 /** Finds the smallest integral power of two not less than the given
00050 value but at least 2 to fit the requirement from
00051 boost::fibers::buffered_channel
00052
00053 \todo At some point use C++20 std::bit_ceil instead of this
00054 DIY function */
00055   static constexpr std::size_t bit_ceil_at_least_2(std::size_t value) {
00056     std::size_t result = 2;
00057     while (value > result)
00058       result *= 2;
00059     return result;
00060   }
00061
00062 public:
00063
00064   // The data type transmitted by the pipe
00065   using value_type = T;
00066
00067 /** The minimum capacity in value_type of this pipe.
00068
00069 The underlying implementation requires a power of 2 */
00070   static constexpr SizeT min_capacity = bit_ceil_at_least_2(MinCapacity);
00071
00072   // The data type used to represent the capacity itself
00073   using size_type = SizeT;
00074
00075 private:
00076
00077 /// A FIFO used to implement the pipe
00078   static inline boost::fibers::buffered_channel<value_type>
00079   channel { min_capacity };
00080
00081 public:
00082
```

```
00083 /** Blocking read from the pipe
00084
00085 \return the value
00086 */
00087   static value_type read() {
00088     return channel.value_pop();
00089   }
00090
00091
00092 /** Blocking write to the pipe
00093
00094 \param[in] value is the data to send to the pipe
00095 */
00096   static void write(const value_type &value) {
00097     channel.push(value);
00098   }
00099
00100
00101 /** Non-blocking read from the pipe
00102
00103 \param[out] success_code is true iff the read was successful
00104
00105 \return the value read from the pipe or the default one
00106 */
00107   static value_type read(bool &success_code) {
00108     value_type value;
00109     success_code =
00110       channel.try_pop(value) == boost::fibers::channel_op_status::success;
00111     /* The current API might be inefficient if we copy always value back.
00112 But we should have a copy elision...  */
00113     return value;
00114   }
00115
00116
00117 /** Non-blocking write to the pipe
00118
00119 \param[out] success_code is true iff the read was successful
00120
00121 \return the value read from the pipe or the default one
00122 */
00123   static void write(const value_type &data, bool &success_code) {
00124       success_code =
00125         channel.try_push(data) != boost::fibers::channel_op_status::full;
00126   }
00127
00128 };
00129
00130
00131 /// @} End the execution Doxygen group
00132
00133 }
00134
00135 /*
00136 # Some Emacs stuff:
00137 ### Local Variables:
00138 ### ispell-local-dictionary:  "american"
00139 ### eval:  (flyspell-prog-mode)
00140 ### End:
00141 */
00142
00143 #endif // TRISYCL_SYCL_PIPE_HPP
```

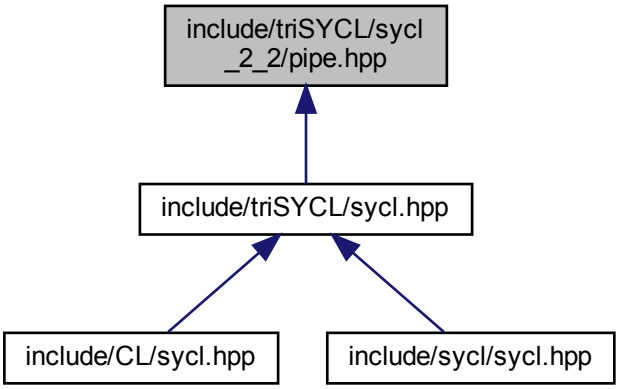
13.179 include/triSYCL/sycl_2_2/pipe.hpp File Reference

```
#include <cstddef>
#include <memory>
#include "triSYCL/access.hpp"
#include "triSYCL/accessor.hpp"
#include "triSYCL/handler.hpp"
#include "triSYCL/sycl_2_2/pipe/detail/pipe.hpp"
```

Include dependency graph for pipe.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::sycl_2_2::pipe< T >

  A SYCL pipe. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::sycl_2_2

13.180 pipe.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_SYCL_2_2_PIPE_HPP
00002 #define TRISYCL_SYCL_SYCL_2_2_PIPE_HPP
00003
00004 /** \file The SYCL 2.2 dataflow pipe<>
```

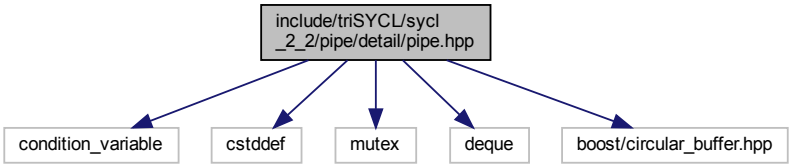
```
00005
00006 This is a proposal for the now abandoned SYCL 2.2 provisional specification.
00007 This is still here for historical reasons.
00008
00009 Ronan at Keryell point FR
00010
00011 This file is distributed under the University of Illinois Open Source
00012 License.  See LICENSE.TXT for details.
00013 */
00014
00015 #include <cstddef>
00016 #include <memory>
00017
00018 #include "triSYCL/access.hpp"
00019 #include "triSYCL/accessor.hpp"
00020 #include "triSYCL/handler.hpp"
00021 #include "triSYCL/sycl_2_2/pipe/detail/pipe.hpp"
00022
00023 namespace trisycl::sycl_2_2 {
00024
00025 /** \addtogroup old_data Data access and storage in old version of SYCL
00026 @{
00027 */
00028
00029 /** A SYCL pipe
00030
00031 This is a proposal for the now abandoned SYCL 2.2 provisional specification.
00032 This is still here for historical reasons.
00033
00034 Implement a FIFO-style object that can be used through accessors
00035 to send some objects T from the input to the output
00036 */
00037 template <typename T>
00038 class pipe
00039     /* Use the underlying pipe implementation that can be shared in
00040 the SYCL model */
00041   :  public detail::shared_ptr_implementation<pipe<T>,
00042                                               detail::sycl_2_2::pipe<T»,
00043     detail::debug<pipe<T» {
00044
00045   // The type encapsulating the implementation
00046   using implementation_t = typename pipe::shared_ptr_implementation;
00047
00048   // Allows the comparison operation to access the implementation
00049   friend implementation_t;
00050
00051 public:
00052
00053   // Make the implementation member directly accessible in this class
00054   using implementation_t::implementation;
00055
00056 /// The STL-like types
00057   /* Since a pipe element cannot be directly addressed without
00058 accessor, only define value_type here */
00059   using value_type = T;
00060
00061
00062 /// Construct a pipe able to store up to capacity T objects
00063   pipe(std::size_t capacity)
00064     :  implementation_t { new detail::sycl_2_2::pipe<T> { capacity } } { }
00065
00066
00067 /** Get an accessor to the pipe with the required mode
00068
00069 \param Mode is the requested access mode
00070
00071 \param Target is the type of pipe access required
00072
00073 \param[in] command_group_handler is the command group handler in
00074 which the kernel is to be executed
00075 */
00076   template <access::mode Mode,
00077             access::target Target = access::target::pipe>
00078   accessor<value_type, 1, Mode, Target>
00079   get_access(handler &command_group_handler) {
00080     static_assert(Target == access::target::pipe
00081                   || Target == access::target::blocking_pipe,
00082                   "get_access(handler) with pipes can only deal with "
00083                   "access::pipe or access::blocking_pipe");
00084     return { implementation, command_group_handler };
00085   }
00086
00087
00088 /// Return the maximum number of elements that can fit in the pipe
00089   std::size_t capacity()const {
00090     return implementation->capacity();
00091     }
```

```
00092
00093 };
00094
00095 /// @} End the execution Doxygen group
00096
00097 }
00098
00099 /*
00100 # Some Emacs stuff:
00101 ### Local Variables:
00102 ### ispell-local-dictionary:  "american"
00103 ### eval:  (flyspell-prog-mode)
00104 ### End:
00105 */
00106
00107 #endif // TRISYCL_SYCL_SYCL_2_2_PIPE_HPP
```

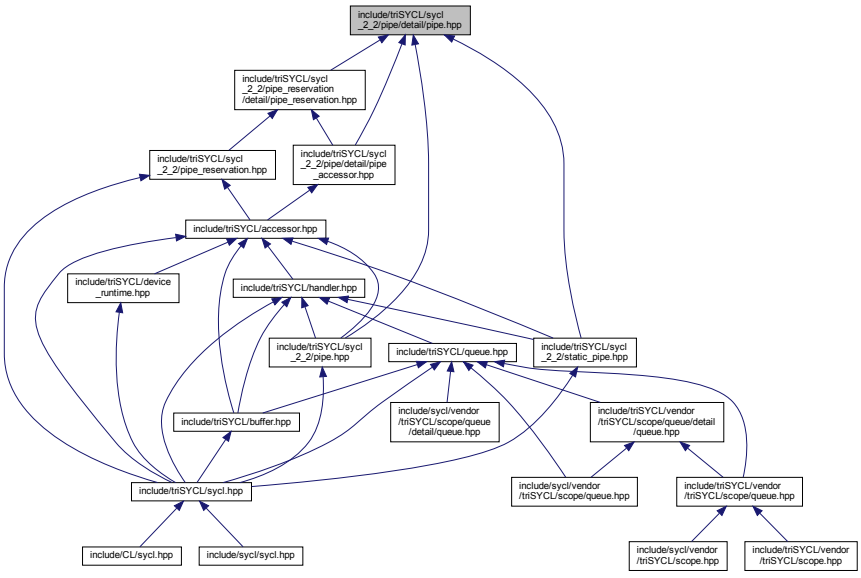
13.181 include/triSYCL/sycl_2_2/pipe/detail/pipe.hpp File Reference

```
#include <condition_variable>
#include <cstddef>
#include <mutex>
#include <deque>
#include <boost/circular_buffer.hpp>
```

Include dependency graph for pipe.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::detail::sycl_2_2::reserve_id< T >

  A private description of a reservation station. More...
- class trisycl::detail::sycl_2_2::pipe< T >

  Implement a pipe object. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.
- namespace trisycl::detail::sycl_2_2

13.182 pipe.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_SYCL_2_2_PIPE_DETAIL_PIPE_HPP
00002 #define TRISYCL_SYCL_SYCL_2_2_PIPE_DETAIL_PIPE_HPP
00003
00004 /** \file The SYCL pipe<> details
00005
00006 This is a proposal for the now abandoned SYCL 2.2 provisional specification.
00007 This is still here for historical reasons.
00008
00009 Ronan at Keryell point FR
00010
00011 This file is distributed under the University of Illinois Open Source
00012 License.  See LICENSE.TXT for details.
00013 */
00014
00015 #include <condition_variable>
00016 #include <cstddef>
00017 #include <mutex>
00018 #include <deque>
00019
00020 #ifdef TRISYCL_MAKE_BOOST_CIRCULARBUFFER_THREAD_SAFE
00021 /* The debug mode of boost/circular_buffer.hpp has a nasty side effect
00022 in multithread applications using several iterators at the same
00023 time even in read-only mode because the library tracks them for
00024 debugging purpose in a...  non-thread safe way
00025
00026 This is described in https://svn.boost.org/trac/boost/ticket/6277
00027 and fixed with https://github.com/boostorg/circular_buffer/pull/9
00028 */
00029 #define BOOST_CB_DISABLE_DEBUG
00030 #endif
00031 #include <boost/circular_buffer.hpp>
00032
00033 namespace trisycl::detail::sycl_2_2 {
00034
00035 /** \addtogroup old_data Data access and storage in old version of SYCL
00036 @{
00037 */
00038
00039 /// A private description of a reservation station
00040 template <typename T>
00041 struct reserve_id {
00042 /// Start of the reservation in the pipe storage
00043   typename boost::circular_buffer<T>::iterator start;
00044
00045 /// Number of elements in the reservation
00046   std::size_t size;
00047
00048   /* True when the reservation has been committed and is ready to be
00049 released */
00050   bool ready = false;
00051
00052 /** Track a reservation not committed yet
00053
```

```
00054 \param[in] start point to the start of the reservation in the
00055 pipe storage
00056
00057 \param[in] size is the number of elements in the reservation
00058 */
00059   reserve_id(typename boost::circular_buffer<T>::iterator start,
00060              std::size_t size) :  start { start }, size { size } {}
00061
00062 };
00063
00064
00065 /** Implement a pipe object
00066
00067 Use some mutable members so that the pipe object can be changed even
00068 when the accessors are captured in a lambda.
00069 */
00070 template <typename T>
00071 class pipe :  public detail::debug<pipe<T>> {
00072
00073 public:
00074
00075   using value_type = T;
00076
00077 /// Implement the pipe with a circular buffer
00078   using implementation_t = boost::circular_buffer<value_type>;
00079
00080 private:
00081
00082 /// The circular buffer to store the elements
00083   boost::circular_buffer<value_type> cb;
00084
00085 /** To protect the access to the circular buffer.
00086
00087 In case the object is capture in a lambda per copy, make it
00088 mutable.  */
00089   mutable std::mutex cb_mutex;
00090
00091 /// The queue of pending write reservations
00092   std::deque<reserve_id<value_type>> w_rid_q;
00093
00094 public:
00095 #ifndef _MSC_VER
00096   using rid_iterator = typename decltype(w_rid_q)::iterator;
00097 #else
00098   using rid_iterator = typename std::deque<reserve_id<value_type>>::iterator;
00099 #endif
00100
00101 private:
00102
00103 /// The queue of pending read reservations
00104   std::deque<reserve_id<value_type>> r_rid_q;
00105
00106 /// Track the number of frozen elements related to read reservations
00107   std::size_t read_reserved_frozen;
00108
00109 /// To signal that a read has been successful
00110   std::condition_variable read_done;
00111
00112 /// To signal that a write has been successful
00113   std::condition_variable write_done;
00114
00115 /// To control the debug mode, disabled by default
00116   bool debug_mode = false;
00117
00118 public:
00119
00120 /// True when the pipe is currently used for reading
00121   bool used_for_reading = false;
00122
00123 /// True when the pipe is currently used for writing
00124   bool used_for_writing = false;
00125
00126 /// Create a pipe as a circular buffer of the required capacity
00127   pipe(std::size_t capacity) :  cb { capacity }, read_reserved_frozen { 0 } { }
00128
00129
00130 /** Return the maximum number of elements that can fit in the pipe
00131 */
00132   std::size_t capacity()const {
00133     // No lock required since it is fixed and set at construction time
00134     return cb.capacity();
00135   }
00136
00137 private:
00138
00139 /** Get the current number of elements in the pipe that can be read
00140
```

```
00141 This is obviously a volatile value which is constrained by the
00142 theory of restricted relativity.
00143
00144 Note that on some devices it may be costly to implement (for
00145 example on FPGA).
00146 */
00147   std::size_t size()const {
00148     TRISYCL_DUMP_T("size() cb.size() = " << cb.size()
00149                    << " cb.end() = " << (void *)&*cb.end()
00150                    << " reserved_for_reading() = " << reserved_for_reading()
00151                    << " reserved_for_writing() = " << reserved_for_writing());
00152     /* The actual number of available elements depends from the
00153 elements blocked by some reservations.
00154 This prevents a consumer to read into reserved area.  */
00155     return cb.size() - reserved_for_reading() - reserved_for_writing();
00156   }
00157
00158
00159 /** Test if the pipe is empty
00160
00161 This is obviously a volatile value which is constrained by
00162 restricted relativity.
00163
00164 Note that on some devices it may be costly to implement on the
00165 write side (for example on FPGA).
00166 */
00167   bool empty()const {
00168     TRISYCL_DUMP_T("empty() cb.size() = " << cb.size()
00169                    << " size() = " << size());
00170     // It is empty when the size is zero, taking into account reservations
00171     return size() ==  0;
00172   }
00173
00174
00175 /** Test if the pipe is full
00176
00177 This is obviously a volatile value which is constrained by
00178 restricted relativity.
00179
00180 Note that on some devices it may be costly to implement on the
00181 read side (for example on FPGA).
00182 */
00183   bool full()const {
00184     return cb.full();
00185   }
00186
00187
00188 public:
00189
00190 /// The size() method used outside needs to lock the datastructure
00191   std::size_t size_with_lock()const {
00192     std::lock_guard<std::mutex> lg { cb_mutex };
00193     return size();
00194   }
00195
00196
00197 /// The empty() method used outside needs to lock the datastructure
00198   bool empty_with_lock()const {
00199     std::lock_guard<std::mutex> lg { cb_mutex };
00200     return empty();
00201   }
00202
00203
00204   // The full() method used outside needs to lock the datastructure
00205   bool full_with_lock()const {
00206     std::lock_guard<std::mutex> lg { cb_mutex };
00207     return full();
00208   }
00209
00210
00211 /** Try to write a value to the pipe
00212
00213 \param[in] value is what we want to write
00214
00215 \param[in] blocking specify if the call wait for the operation
00216 to succeed
00217
00218 \return true on success
00219
00220 \todo provide a && version
00221 */
00222   bool write(const T &value, bool blocking = false) {
00223     // Lock the pipe to avoid being disturbed
00224     std::unique_lock<std::mutex> ul { cb_mutex };
00225     TRISYCL_DUMP_T("Write pipe full = " << full()
00226                    << " value = " << value);
00227
```

```
00228     if (blocking)
00229       /* If in blocking mode, wait for the not full condition, that
00230 may be changed when a read is done */
00231       read_done.wait(ul, [&] { return !full(); });
00232     else if (full())
00233       return false;
00234
00235     cb.push_back(value);
00236     TRISYCL_DUMP_T("Write pipe front = " << cb.front()
00237                    << " back = " << cb.back()
00238                    << " cb.begin() = " << (void *)&*cb.begin()
00239                    << " cb.size() = " << cb.size()
00240                    << " cb.end() = " << (void *)&*cb.end()
00241                    << " reserved_for_reading() = " << reserved_for_reading()
00242                    << " reserved_for_writing() = " << reserved_for_writing());
00243     // Micro-optimization:  unlock before the notification
00244     // https://en.cppreference.com/w/cpp/thread/condition_variable/notify_all
00245     ul.unlock();
00246     // Notify the clients waiting to read something from the pipe
00247     write_done.notify_all();
00248     return true;
00249   }
00250
00251
00252 /** Try to read a value from the pipe
00253
00254 \param[out] value is the reference to where to store what is
00255 read
00256
00257 \param[in] blocking specify if the call wait for the operation
00258 to succeed
00259
00260 \return true on success
00261 */
00262   bool read(T &value, bool blocking = false) {
00263     // Lock the pipe to avoid being disturbed
00264     std::unique_lock<std::mutex> ul { cb_mutex };
00265     TRISYCL_DUMP_T("Read pipe empty = " << empty());
00266
00267     if (blocking)
00268       /* If in blocking mode, wait for the not empty condition, that
00269 may be changed when a write is done */
00270       write_done.wait(ul, [&] { return !empty(); });
00271     else if (empty())
00272       return false;
00273
00274     TRISYCL_DUMP_T("Read pipe front = " << cb.front()
00275                    << " back = " << cb.back()
00276                    << " reserved_for_reading() = " << reserved_for_reading());
00277     if (read_reserved_frozen)
00278 /** If there is a pending reservation, read the next element to
00279 be read and update the number of reserved elements */
00280       value = cb.begin()[read_reserved_frozen++];
00281     else {
00282       /* There is no pending read reservation, so pop the read value
00283 from the pipe */
00284       value = cb.front();
00285       cb.pop_front();
00286     }
00287
00288     TRISYCL_DUMP_T("Read pipe value = " << value);
00289     // Micro-optimization:  unlock before the notification
00290     // https://en.cppreference.com/w/cpp/thread/condition_variable/notify_all
00291     ul.unlock();
00292     // Notify the clients waiting for some room to write in the pipe
00293     read_done.notify_all();
00294     return true;
00295   }
00296
00297
00298 /** Compute the amount of elements blocked by read reservations, not yet
00299 committed
00300
00301 This includes some normal reads to pipes between/after
00302 un-committed reservations
00303
00304 This function assumes that the data structure is locked
00305 */
00306   std::size_t reserved_for_reading()const {
00307     return read_reserved_frozen;
00308   }
00309
00310
00311 /** Compute the amount of elements blocked by write reservations, not yet
00312 committed
00313
00314 This includes some normal writes to pipes between/after
```

```
00315 un-committed reservations
00316
00317 This function assumes that the data structure is locked
00318 */
00319   std::size_t reserved_for_writing()const {
00320     if (w_rid_q.empty())
00321       // No on-going reservation
00322       return 0;
00323     else
00324       /* The reserved size is from the first element of the first
00325 on-going reservation up to the end of the pipe content */
00326       return cb.end() - w_rid_q.front().start;
00327   }
00328
00329
00330 /** Reserve some part of the pipe for reading
00331
00332 \param[in] s is the number of element to reserve
00333
00334 \param[out] rid is an iterator to a description of the
00335 reservation that has been done if successful
00336
00337 \param[in] blocking specify if the call wait for the operation
00338 to succeed
00339
00340 \return true if the reservation was successful
00341 */
00342   bool reserve_read(std::size_t s,
00343                     rid_iterator &rid,
00344                     bool blocking = false)  {
00345     // Lock the pipe to avoid being disturbed
00346     std::unique_lock<std::mutex> ul { cb_mutex };
00347
00348     TRISYCL_DUMP_T("Before read reservation cb.size() = " « cb.size()
00349                    « " size() = " « size());
00350     if (s == 0)
00351       // Empty reservation requested, so nothing to do
00352       return false;
00353
00354     if (blocking)
00355       /* If in blocking mode, wait for enough elements to read in the
00356 pipe for the reservation.  This condition can change when a
00357 write is done */
00358       write_done.wait(ul, [&] { return s <= size(); });
00359     else if (s > size())
00360       // Not enough elements to read in the pipe for the reservation
00361       return false;
00362
00363     // Compute the location of the first element of the reservation
00364     auto first = cb.begin() + read_reserved_frozen;
00365     // Increment the number of frozen elements
00366     read_reserved_frozen += s;
00367     /* Add a description of the reservation at the end of the
00368 reservation queue */
00369     r_rid_q.emplace_back(first, s);
00370     // Return the iterator to the last reservation descriptor
00371     rid = r_rid_q.end() - 1;
00372     TRISYCL_DUMP_T("After reservation cb.size() = " « cb.size()
00373                    « " size() = " « size());
00374     return true;
00375   }
00376
00377
00378 /** Reserve some part of the pipe for writing
00379
00380 \param[in] s is the number of element to reserve
00381
00382 \param[out] rid is an iterator to a description of the
00383 reservation that has been done if successful
00384
00385 \param[in] blocking specify if the call wait for the operation
00386 to succeed
00387
00388 \return true if the reservation was successful
00389 */
00390   bool reserve_write(std::size_t s,
00391                      rid_iterator &rid,
00392                      bool blocking = false)  {
00393     // Lock the pipe to avoid being disturbed
00394     std::unique_lock<std::mutex> ul { cb_mutex };
00395
00396     TRISYCL_DUMP_T("Before write reservation cb.size() = " « cb.size()
00397                    « " size() = " « size());
00398     if (s == 0)
00399       // Empty reservation requested, so nothing to do
00400       return false;
00401
```

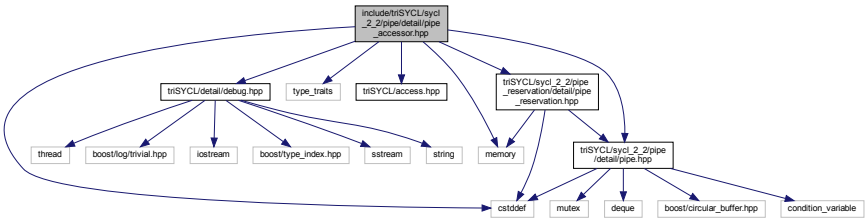
```
00402     if (blocking)
00403       /* If in blocking mode, wait for enough room in the pipe, that
00404 may be changed when a read is done.  Do not use a difference
00405 here because it is only about unsigned values */
00406       read_done.wait(ul, [&] { return cb.size() + s <= capacity(); });
00407     else if (cb.size() + s > capacity())
00408       // Not enough room in the pipe for the reservation
00409       return false;
00410
00411     /* If there is enough room in the pipe, just create default values
00412 in it to do the reservation */
00413     for (std::size_t i = 0; i != s; ++i)
00414       cb.push_back();
00415     /* Compute the location of the first element a posteriori since it
00416 may not exist a priori if cb was empty before */
00417     auto first = cb.end() - s;
00418     /* Add a description of the reservation at the end of the
00419 reservation queue */
00420     w_rid_q.emplace_back(first, s);
00421     // Return the iterator to the last reservation descriptor
00422     rid = w_rid_q.end() - 1;
00423     TRISYCL_DUMP_T("After reservation cb.size() = " « cb.size()
00424                    « " size() = " « size());
00425     return true;
00426   }
00427
00428
00429 /** Process the read reservations that are ready to be released in the
00430 reservation queue
00431 */
00432   void move_read_reservation_forward() {
00433     // Lock the pipe to avoid nuisance
00434     std::unique_lock<std::mutex> lock { cb_mutex };
00435
00436     for (;;) {
00437       if (r_rid_q.empty())
00438         // No pending reservation, so nothing to do
00439         break;
00440       if (!r_rid_q.front().ready)
00441         /* If the first reservation is not ready to be released, stop
00442 because it is blocking all the following in the queue
00443 anyway */
00444         break;
00445       // Remove the reservation to be released from the queue
00446       r_rid_q.pop_front();
00447       std::size_t n_to_pop;
00448       if (r_rid_q.empty())
00449         // If it was the last one, remove all the reservation
00450         n_to_pop = read_reserved_frozen;
00451       else
00452         // Else remove everything up to the next reservation
00453         n_to_pop =  r_rid_q.front().start - cb.begin();
00454       // No longer take into account these reserved slots
00455       read_reserved_frozen -= n_to_pop;
00456       // Release the elements from the FIFO
00457       while (n_to_pop--)
00458         cb.pop_front();
00459       // Notify the clients waiting for some room to write in the pipe
00460       read_done.notify_all();
00461       /* ...and process the next reservation to see if it is ready to
00462 be released too */
00463     }
00464   }
00465
00466
00467 /** Process the write reservations that are ready to be released in the
00468 reservation queue
00469 */
00470   void move_write_reservation_forward() {
00471     // Lock the pipe to avoid nuisance
00472     std::lock_guard<std::mutex> lg { cb_mutex };
00473
00474     for (;;) {
00475       if (w_rid_q.empty())
00476         // No pending reservation, so nothing to do
00477         break;
00478       // Get the first reservation
00479       const auto &rid = w_rid_q.front();
00480       if (!rid.ready)
00481         /* If the reservation is not ready to be released, stop
00482 because it is blocking all the following in the queue
00483 anyway */
00484         break;
00485       // Remove the reservation to be released from the queue
00486       w_rid_q.pop_front();
00487       // Notify the clients waiting to read something from the pipe
00488       write_done.notify_all();
```

```
00489       /* ...and process the next reservation to see if it is ready to
00490 be released too */
00491     }
00492   }
00493
00494 };
00495
00496 /// @} End the execution Doxygen group
00497
00498 }
00499
00500 /*
00501 # Some Emacs stuff:
00502 ### Local Variables:
00503 ### ispell-local-dictionary:  "american"
00504 ### eval:  (flyspell-prog-mode)
00505 ### End:
00506 */
00507
00508 #endif // TRISYCL_SYCL_SYCL_2_2_PIPE_DETAIL_PIPE_HPP
```

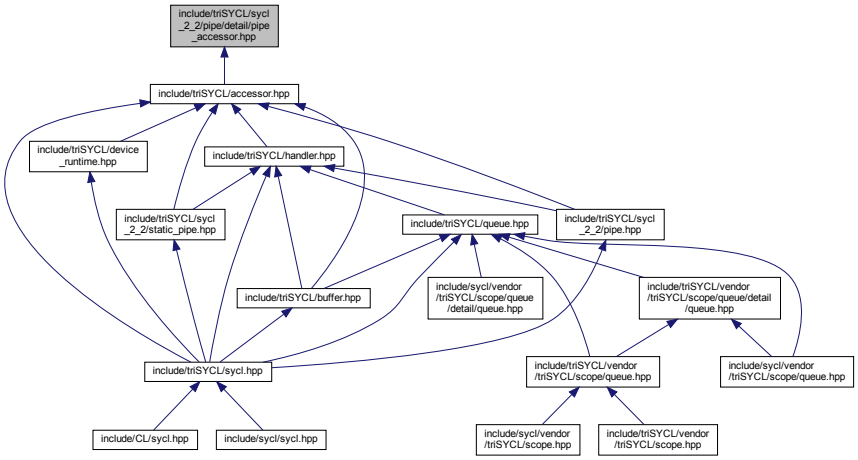
13.183 include/triSYCL/sycl_2_2/pipe/detail/pipe_accessor.hpp File Reference

```
#include <cstddef>
#include <memory>
#include <type_traits>
#include "triSYCL/access.hpp"
#include "triSYCL/detail/debug.hpp"
#include "triSYCL/sycl_2_2/pipe/detail/pipe.hpp"
#include "triSYCL/sycl_2_2/pipe_reservation/detail/pipe_reservation.hpp"
```

Include dependency graph for pipe_accessor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >

  The accessor abstracts the way pipe data are accessed inside a kernel. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.
- namespace trisycl::detail::sycl_2_2

13.184 pipe_accessor.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_SYCL_2_2_PIPE_DETAIL_PIPE_ACCESSOR_HPP
00002 #define TRISYCL_SYCL_SYCL_2_2_PIPE_DETAIL_PIPE_ACCESSOR_HPP
00003
00004 /** \file The SYCL pipe accessor<> detail behind the scene
00005
00006 This is a proposal for the now abandoned SYCL 2.2 provisional specification.
00007 This is still here for historical reasons.
00008
00009 Ronan at Keryell point FR
00010
00011 This file is distributed under the University of Illinois Open Source
00012 License.  See LICENSE.TXT for details.
00013 */
00014
00015 #include <cstddef>
00016 #include <memory>
00017 #include <type_traits>
00018
00019 #include "triSYCL/access.hpp"
00020 #include "triSYCL/detail/debug.hpp"
00021 #include "triSYCL/sycl_2_2/pipe/detail/pipe.hpp"
00022 #include "triSYCL/sycl_2_2/pipe_reservation/detail/pipe_reservation.hpp"
00023
```

```
00024 namespace trisycl {
00025
00026 class handler;
00027
00028 namespace detail {
00029
00030 // Forward declaration of detail::accessor to declare the specialization
00031 template <typename T,
00032           int Dimensions,
00033           access::mode Mode,
00034           access::target Target>
00035 class accessor;
00036
00037 namespace sycl_2_2 {
00038
00039 /** \addtogroup old_data Data access and storage in old version of SYCL
00040 @{
00041 */
00042
00043 /** The accessor abstracts the way pipe data are accessed inside a
00044 kernel
00045 */
00046 template <typename T,
00047           access::mode AccessMode,
00048           access::target Target>
00049 class pipe_accessor :
00050     public detail::debug<detail::sycl_2_2::pipe_accessor<T, AccessMode, Target»
00051 {
00052 public:
00053
00054   static constexpr auto rank = 1;
00055   static constexpr auto mode = AccessMode;
00056   static constexpr auto target = Target;
00057
00058   static constexpr bool blocking =
00059     (target == trisycl::access::target::blocking_pipe);
00060
00061 /// The STL-like types
00062   using value_type = T;
00063   using reference = value_type&;
00064   using const_reference = const value_type&;
00065
00066 private:
00067
00068 /// The real pipe implementation behind the hood
00069   std::shared_ptr<detail::sycl_2_2::pipe<T» implementation;
00070
00071 /** Store the success status of last pipe operation
00072
00073 It is not impacted by reservation success.
00074
00075 It does exist even if the pipe accessor is not evaluated in a
00076 boolean context for, but a use-def analysis can optimise it out
00077 in that case and not use some storage
00078
00079 Use a mutable state here so that it can work with a [=] lambda
00080 capture without having to declare the whole lambda as mutable
00081 */
00082   bool mutable ok = false;
00083
00084 public:
00085
00086 /** Construct a pipe accessor from an existing pipe
00087 */
00088   pipe_accessor(const std::shared_ptr<detail::sycl_2_2::pipe<T>> &p)
00089     :  implementation { p } {
00090     //    TRISYCL_DUMP_T("Create a kernel pipe accessor write = "
00091     //                   « is_write_access());
00092     // Verify that the pipe is not already used in the requested mode
00093     if (mode == access::mode::write)
00094       if (implementation->used_for_writing)
00095 /// \todo Use pipe_exception instead
00096         throw std::logic_error { "The pipe is already used for writing." };
00097       else
00098         implementation->used_for_writing = true;
00099     else
00100       if (implementation->used_for_reading)
00101         throw std::logic_error { "The pipe is already used for reading." };
00102       else
00103         implementation->used_for_reading = true;
00104   }
00105
00106
00107 /** Construct a pipe accessor from an existing pipe
00108
00109 For now the handler is not used.
00110 */
```

```
00111   pipe_accessor(const std::shared_ptr<detail::sycl_2_2::pipe<T>> &p,
00112                 handler &command_group_handler)
00113     :  pipe_accessor(p) {}
00114
00115
00116   pipe_accessor() = default;
00117
00118
00119 /// Return the maximum number of elements that can fit in the pipe
00120   std::size_t capacity()const {
00121     return implementation->capacity();
00122   }
00123
00124 /** Get the current number of elements in the pipe
00125
00126 This is obviously a volatile value which is constrained by
00127 restricted relativity.
00128
00129 Note that on some devices it may be costly to implement (for
00130 example on FPGA).
00131 */
00132   std::size_t size()const {
00133     return implementation->size_with_lock();
00134   }
00135
00136
00137 /** Test if the pipe is empty
00138
00139 This is obviously a volatile value which is constrained by
00140 restricted relativity.
00141
00142 Note that on some devices it may be costly to implement on the
00143 write side (for example on FPGA).
00144 */
00145   bool empty()const {
00146     return implementation->empty_with_lock();
00147   }
00148
00149
00150 /** Test if the pipe is full
00151
00152 This is obviously a volatile value which is constrained by
00153 restricted relativity.
00154
00155 Note that on some devices it may be costly to implement on the
00156 read side (for example on FPGA).
00157 */
00158   bool full()const {
00159     return implementation->full_with_lock();
00160   }
00161
00162
00163 /** In an explicit bool context, the accessor gives the success
00164 status of the last access
00165
00166 It is not impacted by reservation success.
00167
00168 The explicitness is related to avoid \code some_pipe «
00169 some_value \endcode to be interpreted as \code some_bool «
00170 some_value \endcode when the type of \code some_value \endcode
00171 is not the same type as the pipe type.
00172
00173 \return true on success of the previous read or write operation
00174 */
00175   explicit operator bool()const {
00176     return ok;
00177   }
00178
00179
00180 /** Try to write a value to the pipe
00181
00182 \param[in] value is what we want to write
00183
00184 \return this so we can apply a sequence of write for example
00185 (but do not do this on a non blocking pipe...)
00186
00187 \todo provide a && version
00188
00189 This function is const so it can work when the accessor is
00190 passed by copy in the [=] kernel lambda, which is not mutable by
00191 default
00192 */
00193   const pipe_accessor &write(const value_type &value)const {
00194     static_assert(mode == access::mode::write,
00195                   "'.write(const value_type &value)' method on a pipe accessor"
00196                   " is only possible with write access mode");
00197     ok = implementation->write(value, blocking);
```

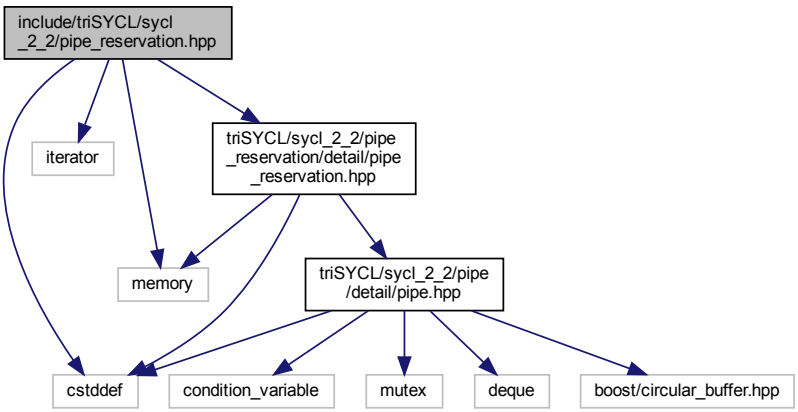
```
00198     // Return a reference to *this so we can apply a sequence of write
00199     return *this;
00200   }
00201
00202
00203 /** Some syntactic sugar to use \code a « v \endcode instead of
00204 \code a.write(v) \endcode */
00205   const pipe_accessor &operator«(const value_type &value)const {
00206     static_assert(mode == access::mode::write,
00207                   "'«' operator on a pipe accessor is only possible"
00208                   " with write access mode");
00209     // Return a reference to *this so we can apply a sequence of »
00210     return write(value);
00211   }
00212
00213
00214 /** Try to read a value from the pipe
00215
00216 \param[out] value is the reference to where to store what is
00217 read
00218
00219 \return \code this \endcode so we can apply a sequence of read
00220 for example (but do not do this on a non blocking pipe...)
00221
00222 This function is const so it can work when the accessor is
00223 passed by copy in the [=] kernel lambda, which is not mutable by
00224 default
00225 */
00226   const pipe_accessor &read(value_type &value)const {
00227     static_assert(mode == access::mode::read,
00228                   "'.read(value_type &value)' method on a pipe accessor"
00229                   " is only possible with read access mode");
00230     ok = implementation->read(value, blocking);
00231     // Return a reference to *this so we can apply a sequence of read
00232     return *this;
00233   }
00234
00235
00236 /** Read a value from a blocking pipe
00237
00238 \return the read value directly, since it cannot fail on
00239 blocking pipe
00240
00241 This function is const so it can work when the accessor is
00242 passed by copy in the [=] kernel lambda, which is not mutable by
00243 default
00244 */
00245   value_type read()const {
00246     static_assert(mode == access::mode::read,
00247                   "'.read()' method on a pipe accessor is only possible"
00248                   " with read access mode");
00249     static_assert(blocking,
00250                   "'.read()' method on a pipe accessor is only possible"
00251                   " with a blocking pipe");
00252     value_type value;
00253     implementation->read(value, blocking);
00254     return value;
00255   }
00256
00257
00258 /** Some syntactic sugar to use \code a » v \endcode instead of
00259 \code a.read(v) \endcode */
00260   const pipe_accessor &operator»(value_type &value)const {
00261     static_assert(mode == access::mode::read,
00262                   "'»' operator on a pipe accessor is only possible"
00263                   " with read access mode");
00264     // Return a reference to *this so we can apply a sequence of »
00265     return read(value);
00266   }
00267
00268
00269   detail::sycl_2_2::pipe_reservation<pipe_accessor>
00270   reserve(std::size_t size)const {
00271     return { *implementation, size };
00272   }
00273
00274
00275 /// Set debug mode
00276   void set_debug(bool enable)const {
00277     implementation->debug_mode = enable;
00278   }
00279
00280
00281   auto &get_pipe_detail() {
00282     return implementation;
00283   }
00284
```

```
00285
00286   ~pipe_accessor() {
00287 /// Free the pipe for a future usage for the current mode
00288     if (mode == access::mode::write)
00289       implementation->used_for_writing = false;
00290     else
00291       implementation->used_for_reading = false;
00292   }
00293
00294 };
00295
00296 /// @} End the data Doxygen group
00297
00298 }
00299 }
00300 }
00301
00302 /*
00303 # Some Emacs stuff:
00304 ### Local Variables:
00305 ### ispell-local-dictionary:  "american"
00306 ### eval:  (flyspell-prog-mode)
00307 ### End:
00308 */
00309
00310 #endif // TRISYCL_SYCL_SYCL_2_2_PIPE_DETAIL_PIPE_ACCESSOR_HPP
```

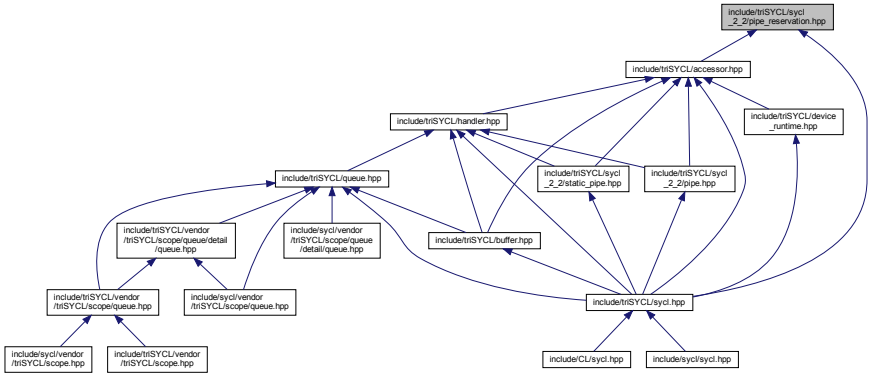
13.185 include/triSYCL/sycl_2_2/pipe_reservation.hpp File Reference

```
#include <cstddef>
#include <iterator>
#include <memory>
#include "triSYCL/sycl_2_2/pipe_reservation/detail/pipe_reservation.hpp"
```

Include dependency graph for pipe_reservation.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::sycl_2_2::pipe_reservation< PipeAccessor >

  The pipe reservation station allows to reserve an array-like view inside the pipe for ordered race-free access from various work-items for example. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::sycl_2_2

13.186 pipe_reservation.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_SYCL_2_2_PIPE_RESERVATION_HPP
00002 #define TRISYCL_SYCL_SYCL_2_2_PIPE_RESERVATION_HPP
00003
00004 /** \file The reservation station for SYCL pipe accessor<>
00005
00006 This is a proposal for the now abandoned SYCL 2.2 provisional specification.
00007 This is still here for historical reasons.
00008
00009 Ronan at Keryell point FR
00010
00011 This file is distributed under the University of Illinois Open Source
00012 License.  See LICENSE.TXT for details.
00013 */
00014
00015 #include <cstddef>
00016 #include <iterator>
00017 #include <memory>
00018
00019 #include "triSYCL/sycl_2_2/pipe_reservation/detail/pipe_reservation.hpp"
00020
00021 namespace trisycl::sycl_2_2 {
00022
00023 /** \addtogroup old_data Data access and storage in old version of SYCL
00024 @{
00025 */
00026
00027 /** The pipe reservation station allows to reserve an array-like view
00028 inside the pipe for ordered race-free access from various
00029 work-items for example
00030 */
```

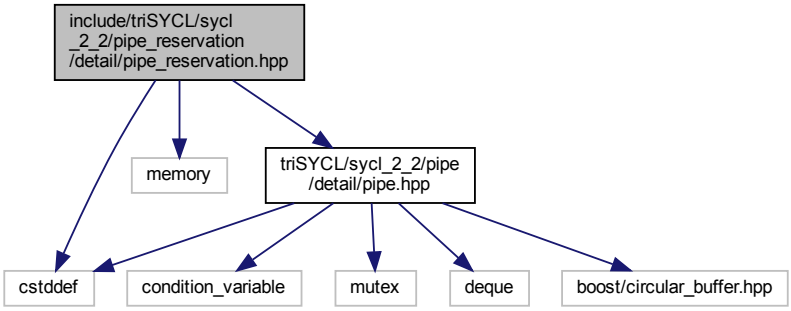
```
00031 template <typename PipeAccessor>
00032 struct pipe_reservation {
00033   using accessor_type = PipeAccessor;
00034   static constexpr bool blocking =
00035     (accessor_type::target == trisycl::access::target::blocking_pipe);
00036   using accessor_detail = typename accessor_type::accessor_detail;
00037 /// The STL-like types
00038   using value_type = typename accessor_type::value_type;
00039   using reference = value_type&;
00040   using const_reference = const value_type&;
00041   using pointer = value_type*;
00042   using const_pointer = const value_type*;
00043   using size_type = std::size_t;
00044   using difference_type = ptrdiff_t;
00045   using iterator =
00046     typename detail::sycl_2_2::pipe_reservation<accessor_detail>::iterator;
00047   using const_iterator =
00048     typename detail::sycl_2_2::pipe_reservation<accessor_detail>::const_iterator;
00049   using reverse_iterator = std::reverse_iterator<iterator>;
00050   using const_reverse_iterator = std::reverse_iterator<const_iterator>;
00051
00052 /** Point to the underlying implementation that can be shared in the
00053 SYCL model with a handler semantics */
00054   typename std::shared_ptr<detail::sycl_2_2::pipe_reservation<accessor_detail»
00055   implementation;
00056
00057 /** Use default constructors so that we can create a new buffer copy
00058 from another one, with either a l-value or a r-value (for
00059 std::move() for example).
00060
00061 Since we just copy the shared_ptr<> above, this is where/how the
00062 sharing magic is happening with reference counting in this case.
00063 */
00064   pipe_reservation() = default;
00065
00066
00067 /// Create a pipe_reservation for an accessor and a number of elements
00068   pipe_reservation(accessor_type &accessor, std::size_t s)
00069     : implementation {
00070     new detail::sycl_2_2::pipe_reservation<accessor_detail> {
00071       get_pipe_detail(accessor), s }
00072   } {}
00073
00074
00075 /** Create a pipe_reservation from the implementation detail
00076
00077 This is an internal constructor to allow reserve() on the
00078 implementation to lift a full-fledged object through
00079 accessor::reserve().
00080
00081 \todo Make it private and add required friends
00082 */
00083   pipe_reservation(detail::sycl_2_2::pipe_reservation<accessor_detail> &&pr)
00084     : implementation {
00085     new detail::sycl_2_2::pipe_reservation<accessor_detail> { std::move(pr) } }
00086   {}
00087
00088
00089 /** Test if the pipe_reservation has been correctly allocated
00090
00091 \return true if the pipe_reservation can be used and committed
00092 */
00093   operator bool()const {
00094     return *implementation;
00095   }
00096
00097
00098 /// Get the number of reserved element(s)
00099   std::size_t size()const {
00100     return implementation->size();
00101   }
00102
00103
00104 /// Access to a given element of the reservation
00105   reference operator[](std::size_t index)const {
00106     return (*implementation)[index];
00107   }
00108
00109
00110 /** Force a commit operation
00111
00112 Normally the commit is implicitly done in the destructor, but
00113 sometime it is useful to do it earlier.
00114 */
00115   void commit()const {
00116     return implementation->commit();
00117   }
```

```
00118
00119
00120 /// Get an iterator on the first element of the reservation station
00121   iterator begin()const {
00122     return implementation->begin();
00123   }
00124
00125
00126 /// Get an iterator past the end of the reservation station
00127   iterator end()const {
00128     return implementation->end();
00129   }
00130
00131
00132 /// Build a constant iterator on the first element of the reservation station
00133   const_iterator cbegin()const {
00134     return implementation->begin();
00135   }
00136
00137
00138 /// Build a constant iterator past the end of the reservation station
00139   const_iterator cend()const {
00140     return implementation->end();
00141   }
00142
00143
00144 /// Get a reverse iterator on the last element of the reservation station
00145   reverse_iterator rbegin()const {
00146     return std::make_reverse_iterator(end());
00147   }
00148
00149
00150 /** Get a reverse iterator on the first element past the end of the
00151 reservation station */
00152   reverse_iterator rend()const {
00153     return std::make_reverse_iterator(begin());
00154   }
00155
00156
00157 /** Get a constant reverse iterator on the last element of the
00158 reservation station */
00159   const_reverse_iterator crbegin()const {
00160     return std::make_reverse_iterator(cend());
00161   }
00162
00163
00164 /** Get a constant reverse iterator on the first element past the
00165 end of the reservation station */
00166   const_reverse_iterator crend()const {
00167     return std::make_reverse_iterator(cbegin());
00168   }
00169
00170 };
00171
00172 /// @} End the data Doxygen group
00173
00174 }
00175
00176 /*
00177 # Some Emacs stuff:
00178 ### Local Variables:
00179 ### ispell-local-dictionary:  "american"
00180 ### eval:  (flyspell-prog-mode)
00181 ### End:
00182 */
00183
00184 #endif // TRISYCL_SYCL_SYCL_2_2_PIPE_RESERVATION_HPP
```

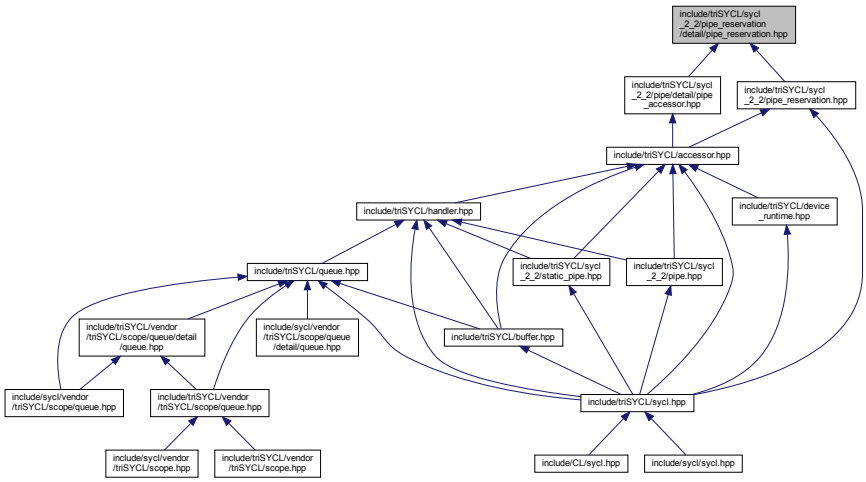
13.187 include/triSYCL/sycl_2_2/pipe_reservation/detail/pipe_↩ reservation.hpp File Reference

```
#include <cstddef>
#include <memory>
#include "triSYCL/sycl_2_2/pipe/detail/pipe.hpp"
```

Include dependency graph for pipe_reservation.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >

  The implementation of the pipe reservation station. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.
- namespace trisycl::detail::sycl_2_2

13.188 pipe_reservation.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_SYCL_2_2_PIPE_RESERVATION_DETAIL_PIPE_RESERVATION_HPP
00002 #define TRISYCL_SYCL_SYCL_2_2_PIPE_RESERVATION_DETAIL_PIPE_RESERVATION_HPP
00003
00004 /** \file The SYCL pipe reservation detail behind the scene
00005
00006 This is a proposal for the now abandoned SYCL 2.2 provisional specification.
00007 This is still here for historical reasons.
00008
00009 Ronan at Keryell point FR
00010
00011 This file is distributed under the University of Illinois Open Source
00012 License.  See LICENSE.TXT for details.
00013 */
00014
00015 #include <cstddef>
00016 #include <memory>
00017
00018 #include "triSYCL/sycl_2_2/pipe/detail/pipe.hpp"
00019
00020 namespace trisycl::detail {
00021
00022 template <typename T,
00023           int Dimensions,
00024           access::mode Mode,
00025           access::target Target>
00026 class accessor;
00027
00028 namespace sycl_2_2 {
00029
00030 /** \addtogroup old_data Data access and storage in old version of SYCL
00031 @{
00032 */
00033
00034 /// The implementation of the pipe reservation station
00035 template <typename PipeAccessor>
00036 class pipe_reservation :
00037     public detail::debug<detail::sycl_2_2::pipe_reservation<PipeAccessor>> {
00038   using accessor_type = PipeAccessor;
00039   static constexpr bool blocking =
00040     (accessor_type::target == trisycl::access::target::blocking_pipe);
00041   using value_type = typename accessor_type::value_type;
00042   using reference = typename accessor_type::reference;
00043
00044 public:
00045
00046   using iterator = typename
00047     detail::sycl_2_2::pipe<value_type>::implementation_t::iterator;
00048   using const_iterator = typename
00049     detail::sycl_2_2::pipe<value_type>::implementation_t::const_iterator;
00050
00051   // \todo Add to the specification
00052   static constexpr access::mode mode = accessor_type::mode;
00053   static constexpr access::target target = accessor_type::target;
00054
00055 /**  True if the reservation was successful and still uncommitted.  B
00056 default a pipe_reservation is not reserved and cannot be
00057 committed */
00058   bool ok = false;
00059
00060 /// Point into the reservation buffer.  Only valid if ok is true
00061   typename detail::sycl_2_2::pipe<value_type>::rid_iterator rid;
00062
00063 /** Keep a reference on the pipe to access to the data and methods
00064
00065 Note that with inlining and CSE it should not use more register
00066 when compiler optimization is in use.  */
00067   detail::sycl_2_2::pipe<value_type> &p;
00068
00069
00070 /** Test that the reservation is in a usable state
00071
00072 \todo Throw exception instead
00073 */
00074   void assume_validity() {
00075     assert(ok);
00076   }
00077
00078 public:
00079
00080 /// Create a pipe reservation station that reserves the pipe itself
00081   pipe_reservation(detail::sycl_2_2::pipe<value_type> &p,
00082                    std::size_t s)
```

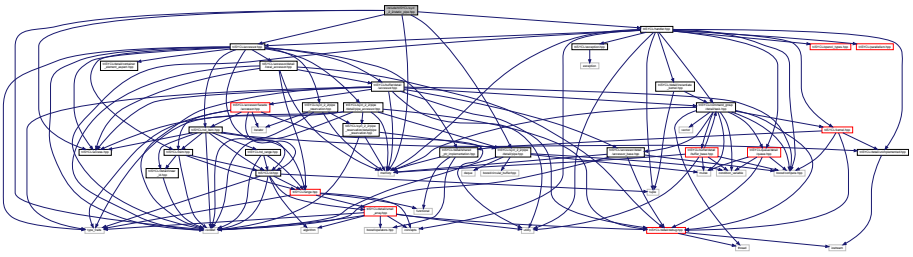
```
00083     :  p { p } {
00084     static_assert(mode == access::mode::write
00085                   || mode == access::mode::read,
00086                   "A pipe can only be accesed in read or write mode,"
00087                   " exclusively");
00088
00089     /* Since this test is constexpr and dependent of a template
00090 parameter, it should be equivalent to a specialization of the
00091 method but in a clearer way */
00092     if (mode == access::mode::write)
00093       ok = p.reserve_write(s, rid, blocking);
00094     else
00095       ok = p.reserve_read(s, rid, blocking);
00096   }
00097
00098
00099 /** No copy constructor with some spurious commit in the destructor
00100 of the original object
00101 */
00102   pipe_reservation(const pipe_reservation &) = delete;
00103
00104
00105 /// Only a move constructor is required to move it into the shared_ptr
00106   pipe_reservation(pipe_reservation &&orig) :
00107     ok {orig.ok },
00108     rid {orig.rid },
00109     p { orig.p } {
00110       /* Even when an object is moved, the destructor of the old
00111 object is eventually called, so leave the old object in a
00112 destructable state but without any commit capability */
00113       orig.ok = false;
00114     }
00115
00116
00117 /** Keep the default constructors too
00118
00119 Otherwise there is no move semantics and the copy is made by
00120 creating a new reservation and destructing the old one with a
00121 spurious commit in the meantime...
00122 */
00123   pipe_reservation() = default;
00124
00125
00126 /** Test if the reservation succeeded and thus if the reservation
00127 can be committed
00128
00129 Note that it is up to the user to ensure that all the
00130 reservation elements have been initialized correctly in the case
00131 of a write for example
00132 */
00133   operator bool() {
00134     return ok;
00135   }
00136
00137
00138 /// Start of the reservation area
00139   iterator begin() {
00140     assume_validity();
00141     return rid->start;
00142   }
00143
00144
00145 /// Past the end of the reservation area
00146   iterator end() {
00147     assume_validity();
00148     return rid->start + rid->size;
00149   }
00150
00151
00152 /// Get the number of elements in the reservation station
00153   std::size_t size() {
00154     assume_validity();
00155     return rid->size;
00156   }
00157
00158
00159 /// Access to an element of the reservation
00160   reference operator[](std::size_t index) {
00161     assume_validity();
00162     TRISYCL_DUMP_T("[] index = " « index
00163                    « " Reservation write address = " « &(rid->start[index]));
00164
00165     return rid->start[index];
00166   }
00167
00168
00169 /** Commit the reservation station
```

```
00170
00171 \todo Add to the specification that for simplicity a reservation
00172 can be commited several times but only the first one is taken
00173 into account
00174 */
00175   void commit() {
00176     if (ok) {
00177       // If the reservation is in a committable state, commit
00178       TRISYCL_DUMP_T("Commit");
00179       rid->ready = true;
00180       if (mode == access::mode::write)
00181         p.move_write_reservation_forward();
00182       else
00183         p.move_read_reservation_forward();
00184       ok = false;
00185     }
00186   }
00187
00188
00189 /// An implicit commit is made in the destructor
00190   ~pipe_reservation() {
00191     commit();
00192   }
00193
00194 };
00195
00196 /// @} End the data Doxygen group
00197
00198 }
00199 }
00200
00201 /*
00202 # Some Emacs stuff:
00203 ### Local Variables:
00204 ### ispell-local-dictionary:  "american"
00205 ### eval:  (flyspell-prog-mode)
00206 ### End:
00207 */
00208
00209 #endif // TRISYCL_SYCL_SYCL_2_2_PIPE_RESERVATION_DETAIL_PIPE_RESERVATION_HPP
```

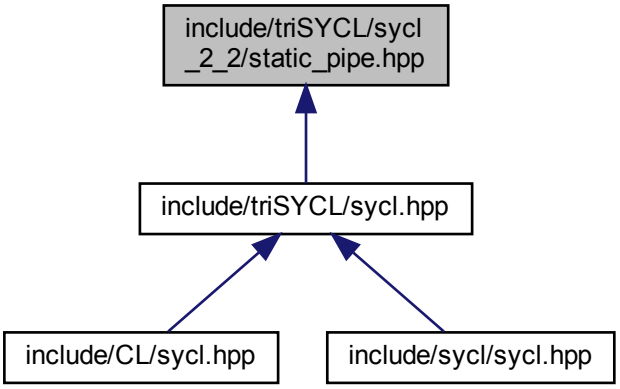
13.189 include/triSYCL/sycl_2_2/static_pipe.hpp File Reference

```
#include <cstddef>
#include <memory>
#include "triSYCL/access.hpp"
#include "triSYCL/accessor.hpp"
#include "triSYCL/handler.hpp"
#include "triSYCL/sycl_2_2/pipe/detail/pipe.hpp"
```

Include dependency graph for static_pipe.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::sycl_2_2::static_pipe< T, Capacity >

  A SYCL static-scoped pipe equivalent to an OpenCL program-scoped pipe. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::sycl_2_2

13.190 static_pipe.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_SYCL_2_2_STATIC_PIPE_HPP
00002 #define TRISYCL_SYCL_SYCL_2_2_STATIC_PIPE_HPP
00003
00004 /** \file TheSYCL static-scoped pipe equivalent to an OpenCL
00005 program-scoped pipe
00006
00007 This is a proposal for the now abandoned SYCL 2.2 provisional specification.
00008 This is still here for historical reasons.
00009
00010 Ronan at Keryell point FR
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include <cstddef>
00017 #include <memory>
00018
00019 #include "triSYCL/access.hpp"
00020 #include "triSYCL/accessor.hpp"
00021 #include "triSYCL/handler.hpp"
00022 #include "triSYCL/sycl_2_2/pipe/detail/pipe.hpp"
00023
00024 namespace trisycl::sycl_2_2 {
```

```
00025
00026 /** \addtogroup old_data Data access and storage in old version of SYCL
00027 @{
00028 */
00029
00030 /** A SYCL static-scoped pipe equivalent to an OpenCL program-scoped
00031 pipe
00032
00033 Implement a FIFO-style object that can be used through accessors
00034 to send some objects T from the input to the output.
00035
00036 Compared to a normal pipe, a static_pipe takes a constexpr size
00037 and is expected to be declared in a compile-unit static context so
00038 the compiler can generate everything at compile time.
00039
00040 This is useful to generate a fixed and optimized hardware
00041 implementation on FPGA for example, where the interconnection
00042 graph can be also inferred at compile time.
00043
00044 It is not directly mapped to the OpenCL program-scoped pipe
00045 because in SYCL there is not this concept of separated
00046 program.  But the SYCL device compiler is expected to generate some
00047 OpenCL program(s) with program-scoped pipes when a SYCL
00048 static-scoped pipe is used.  These details are implementation
00049 defined.
00050 */
00051 template <typename T, std::size_t Capacity>
00052 class static_pipe
00053     /* Use the underlying pipe implementation that can be shared in
00054 the SYCL model */
00055   :  public detail::shared_ptr_implementation<static_pipe<T, Capacity>,
00056                                               detail::sycl_2_2::pipe<T»,
00057     detail::debug<static_pipe<T, Capacity» {
00058
00059   // The type encapsulating the implementation
00060   using implementation_t = typename static_pipe::shared_ptr_implementation;
00061
00062   // Make the implementation member directly accessible in this class
00063   using implementation_t::implementation;
00064
00065   // Allows the comparison operation to access the implementation
00066   friend implementation_t;
00067
00068 public:
00069
00070 /// The STL-like types
00071   using value_type = T;
00072
00073
00074 /// Construct a static-scoped pipe able to store up to Capacity T objects
00075   static_pipe()
00076     :  implementation_t { new detail::sycl_2_2::pipe<T> { Capacity } } { }
00077
00078
00079 /** Get an accessor to the pipe with the required mode
00080
00081 \param Mode is the requested access mode
00082
00083 \param Target is the type of pipe access required
00084
00085 \param[in] command_group_handler is the command group handler in
00086 which the kernel is to be executed
00087 */
00088   template <access::mode Mode,
00089             access::target Target = access::target::pipe>
00090   accessor<value_type, 1, Mode, Target>
00091   get_access(handler &command_group_handler)const {
00092     static_assert(Target == access::target::pipe
00093                   || Target == access::target::blocking_pipe,
00094                   "get_access(handler) with pipes can only deal with "
00095                   "access::pipe or access::blocking_pipe");
00096     return { implementation, command_group_handler };
00097   }
00098
00099
00100 /** Get an accessor to the pipe with the required mode, outside of a
00101 normal kernel, for example in host code
00102
00103 \param Mode is the requested access mode
00104
00105 \param Target is the type of pipe access required
00106 */
00107   template <access::mode Mode,
00108             access::target Target = access::target::pipe>
00109   accessor<value_type, 1, Mode, Target>
00110   get_access()const {
00111     static_assert(Target == access::target::pipe
```

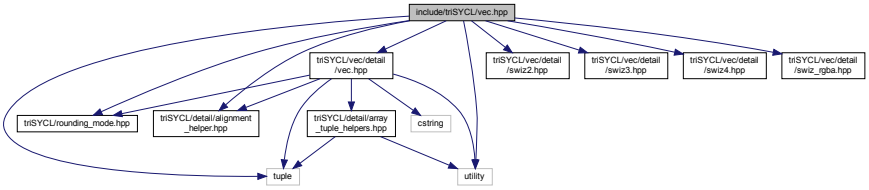
```
00112                   || Target == access::target::blocking_pipe,
00113                   "get_access(handler) with pipes can only deal with "
00114                   "access::pipe or access::blocking_pipe");
00115     return { implementation };
00116   }
00117
00118
00119 /** Return the maximum number of elements that can fit in the pipe
00120
00121 This is a constexpr since the capacity is in the type.
00122 */
00123   std::size_t constexpr capacity()const {
00124     return Capacity;
00125     }
00126
00127 };
00128
00129 /// @} End the execution Doxygen group
00130
00131 }
00132
00133 /*
00134 # Some Emacs stuff:
00135 ### Local Variables:
00136 ### ispell-local-dictionary:  "american"
00137 ### eval:  (flyspell-prog-mode)
00138 ### End:
00139 */
00140
00141 #endif // TRISYCL_SYCL_SYCL_2_2_STATIC_PIPE_HPP
```

13.191 include/triSYCL/vec.hpp File Reference

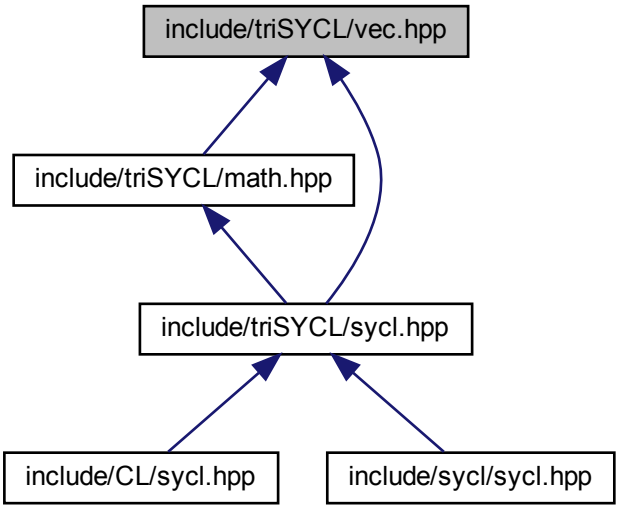
Implement the small vector class.

```
#include <tuple>
#include <utility>
#include "triSYCL/rounding_mode.hpp"
#include "triSYCL/detail/alignment_helper.hpp"
#include "triSYCL/vec/detail/vec.hpp"
#include "triSYCL/vec/detail/swiz2.hpp"
#include "triSYCL/vec/detail/swiz3.hpp"
#include "triSYCL/vec/detail/swiz4.hpp"
#include "triSYCL/vec/detail/swiz_rgba.hpp"
```

Include dependency graph for vec.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::elem

  Series of values to specify named swizzle indexes. More...
- class trisycl::vec< DataType, 1 >
- class trisycl::vec< DataType, 2 >
- class trisycl::vec< DataType, 3 >
- class trisycl::vec< DataType, 4 >
- class trisycl::vec< DataType, 8 >
- class trisycl::vec< DataType, 16 >
- struct std::tuple_element< I, trisycl::vec< T, Size > >

  The std::tuple_element specialization for vec.
- struct std::tuple_size< trisycl::vec< T, Size > >

  The std::tuple_size specialization for vec.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace std

Macros

- #define TRISYCL_DECLARE_S(x)

  Accessors to access hex indexed elements of a vector.
- #define TRISYCL_GEN_SWIZ2(str, idx0, idx1)
- #define TRISYCL_GEN_SWIZ3(str, idx0, idx1, idx2)
- #define TRISYCL_GEN_SWIZ4(str, idx0, idx1, idx2, idx3)
- #define TRISYCL_DEFINE_VEC_TYPE_SIZE(type, size, actual_type) using type##size = vec<actual_type, size>;

  A macro to define type alias, such as for type=uchar, size=4 and actual_type=unsigned char, uchar4 is equivalent to vec<unsigned char, 4>
- #define TRISYCL_DEFINE_VEC_TYPE(type, actual_type)

  Declare the vector types of a type for all the sizes.

Typedefs

- template<typename DataType , int numElements>
  using trisycl::__swizzled_vec__ = vec< DataType, numElements >

Functions

- template<int I, typename T , int Size>
  auto std::get (const trisycl::vec< T, Size > &v)

  Provide a get<> accessor to have C++17 structured binding working.

13.191.1 Detailed Description

Implement the small vector class.

Ronan at Keryell point FR

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file vec.hpp.

13.192 vec.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VEC_HPP
00002 #define TRISYCL_SYCL_VEC_HPP
00003
00004 /** \file
00005
00006 Implement the small vector class
00007
00008 Ronan at Keryell point FR
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include <tuple>
00015 #include <utility>
00016
00017 #include "triSYCL/rounding_mode.hpp"
00018 #include "triSYCL/detail/alignment_helper.hpp"
```

```
00019 #include "triSYCL/vec/detail/vec.hpp"
00020
00021 namespace trisycl {
00022
00023 /** Series of values to specify named swizzle indexes
00024
00025 Used when calling the swizzle member function template
00026 */
00027 struct elem {
00028   static constexpr int x = 0;
00029   static constexpr int y = 1;
00030   static constexpr int z = 2;
00031   static constexpr int w = 3;
00032   static constexpr int r = 0;
00033   static constexpr int g = 1;
00034   static constexpr int b = 2;
00035   static constexpr int a = 3;
00036   static constexpr int s0 = 0;
00037   static constexpr int s1 = 1;
00038   static constexpr int s2 = 2;
00039   static constexpr int s3 = 3;
00040   static constexpr int s4 = 4;
00041   static constexpr int s5 = 5;
00042   static constexpr int s6 = 6;
00043   static constexpr int s7 = 7;
00044   static constexpr int s8 = 8;
00045   static constexpr int s9 = 9;
00046   static constexpr int sA = 10;
00047   static constexpr int sB = 11;
00048   static constexpr int sC = 12;
00049   static constexpr int sD = 13;
00050   static constexpr int sE = 14;
00051   static constexpr int sF = 15;
00052 };
00053
00054 template <typename DataType, int numElements>
00055 using __swizzled_vec__ = vec<DataType, numElements>;
00056
00057 }
00058
00059 /** \addtogroup vector Vector types in SYCL
00060
00061 @{
00062 */
00063
00064 /// Small SYCL vector class
00065 namespace trisycl {
00066 /// Accessors to access hex indexed elements of a vector
00067 #define TRISYCL_DECLARE_S(x)                          \
00068 const DataType& s##x() const {                        \
00069 return (*this)[0x##x];                              \
00070 }                                                     \
00071 \
00072 DataType& s##x() {                                    \
00073 return (*this)[0x##x];                              \
00074 }
00075
00076 #define TRISYCL_GEN_SWIZ2(str,idx0,idx1)              \
00077 const __swizzled_vec__<DataType, 2> str() const {     \
00078 return base_vec::swizzle(idx0, idx1);               \
00079 }                                                     \
00080 __swizzled_vec__<DataType, 2> str() {                 \
00081 return base_vec::swizzle(idx0, idx1);               \
00082 }
00083 #define TRISYCL_GEN_SWIZ3(str,idx0,idx1,idx2)         \
00084 const __swizzled_vec__<DataType, 3> str() const {     \
00085 return base_vec::swizzle(idx0, idx1, idx2);         \
00086 }                                                     \
00087 __swizzled_vec__<DataType, 3> str() {                 \
00088 return base_vec::swizzle(idx0, idx1, idx2);         \
00089 }
00090 #define TRISYCL_GEN_SWIZ4(str,idx0,idx1,idx2,idx3)    \
00091 const __swizzled_vec__<DataType, 4> str() const {     \
00092 return base_vec::swizzle(idx0, idx1, idx2, idx3); \
00093 }                                                     \
00094 __swizzled_vec__<DataType, 4> str() {                 \
00095 return base_vec::swizzle(idx0, idx1, idx2, idx3); \
00096 }
00097
00098 template<typename DataType>
00099 class alignas(detail::alignment_v<::trisycl::vec<DataType, 1>>)
00100   vec<DataType, 1> :  public detail::vec<DataType, 1> {
00101   using base_vec = detail::vec<DataType, 1>;
00102
00103 public:
00104
00105   /* use base class constructors */
```

```
00106   using base_vec::base_vec;
00107
00108 /** Accessors to the first element of a vector
00109 */
00110   const DataType& x()const {
00111     return (*this)[0];
00112   }
00113
00114   DataType& x() {
00115     return (*this)[0];
00116   }
00117
00118   TRISYCL_DECLARE_S(0);
00119
00120   operator DataType()const {
00121     return (*this)[0];
00122   }
00123
00124   operator DataType() {
00125     return (*this)[0];
00126   }
00127 };
00128
00129 template<typename DataType>
00130 class alignas(detail::alignment_v<::trisycl::vec<DataType, 2>)
00131   vec<DataType, 2> :  public detail::vec<DataType, 2> {
00132   using base_vec = detail::vec<DataType, 2>;
00133
00134 public:
00135
00136   /* use base class constructors */
00137   using base_vec::base_vec;
00138
00139 /** Accessors to the first element of a vector
00140 */
00141   const DataType& x()const {
00142     return (*this)[0];
00143   }
00144
00145   DataType& x() {
00146     return (*this)[0];
00147   }
00148
00149 /** Accessors to the second element of a vector
00150 */
00151   const DataType& y()const {
00152     return (*this)[1];
00153   }
00154
00155   DataType& y() {
00156     return (*this)[1];
00157   }
00158
00159   TRISYCL_DECLARE_S(0);
00160   TRISYCL_DECLARE_S(1);
00161
00162   const __swizzled_vec__<DataType, 1> lo()const {
00163     return base_vec::swizzle(elem::s0);
00164   }
00165
00166   __swizzled_vec__<DataType, 1> lo() {
00167     return base_vec::swizzle(elem::s0);
00168   }
00169
00170   const __swizzled_vec__<DataType, 1> hi()const {
00171     return base_vec::swizzle(elem::s1);
00172   }
00173
00174   __swizzled_vec__<DataType, 1> hi() {
00175     return base_vec::swizzle(elem::s1);
00176   }
00177
00178   const __swizzled_vec__<DataType, 1> odd()const {
00179     return base_vec::swizzle(elem::s1);
00180   }
00181
00182   __swizzled_vec__<DataType, 1> odd() {
00183     return base_vec::swizzle(elem::s1);
00184   }
00185
00186   const __swizzled_vec__<DataType, 1> even()const {
00187     return base_vec::swizzle(elem::s0);
00188   }
00189
00190   __swizzled_vec__<DataType, 1> even() {
00191     return base_vec::swizzle(elem::s0);
00192   }
```

```
00193 #include "triSYCL/vec/detail/swiz2.hpp"
00194 };
00195
00196 template<typename DataType>
00197 class alignas(detail::alignment_v<::trisycl::vec<DataType, 3»)
00198   vec<DataType, 3> :  public detail::vec<DataType, 3> {
00199   using base_vec = detail::vec<DataType, 3>;
00200
00201 public:
00202
00203   /* use base class constructors */
00204   using base_vec::base_vec;
00205
00206 /** Accessors to the first element of a vector
00207 */
00208   const DataType& x()const {
00209     return (*this)[0];
00210   }
00211
00212   DataType& x() {
00213     return (*this)[0];
00214   }
00215
00216 /** Accessors to the second element of a vector
00217 */
00218   const DataType& y()const {
00219     return (*this)[1];
00220   }
00221
00222   DataType& y() {
00223     return (*this)[1];
00224   }
00225
00226 /** Accessors to the third element of a vector
00227 */
00228   const DataType& z()const {
00229      return (*this)[2];
00230   }
00231
00232   DataType& z() {
00233     return (*this)[2];
00234   }
00235
00236   TRISYCL_DECLARE_S(0);
00237   TRISYCL_DECLARE_S(1);
00238   TRISYCL_DECLARE_S(2);
00239
00240   const __swizzled_vec__<DataType, 2> lo()const {
00241     return base_vec::swizzle(elem::s0, elem::s1);
00242   }
00243
00244   __swizzled_vec__<DataType, 2> lo()  {
00245     return base_vec::swizzle(elem::s0, elem::s1);
00246   }
00247
00248   const __swizzled_vec__<DataType, 2> hi()const {
00249     return base_vec::swizzle(elem::s2, elem::s2);
00250   }
00251
00252   __swizzled_vec__<DataType, 2> hi() {
00253     return base_vec::swizzle(elem::s2, elem::s2);
00254   }
00255
00256   const __swizzled_vec__<DataType, 2> odd()const {
00257     return base_vec::swizzle(elem::s1, elem::s1);
00258   }
00259
00260   __swizzled_vec__<DataType, 2> odd() {
00261     return base_vec::swizzle(elem::s1, elem::s1);
00262   }
00263
00264   const __swizzled_vec__<DataType, 2> even()const {
00265     return base_vec::swizzle(elem::s0, elem::s2);
00266   }
00267
00268   __swizzled_vec__<DataType, 2> even() {
00269     return base_vec::swizzle(elem::s0, elem::s2);
00270   }
00271 #include "triSYCL/vec/detail/swiz3.hpp"
00272 };
00273
00274 template<typename DataType>
00275 class alignas(detail::alignment_v<::trisycl::vec<DataType, 4»)
00276   vec<DataType, 4> :  public detail::vec<DataType, 4> {
00277   using base_vec = detail::vec<DataType, 4>;
00278
00279 public:
```

```
00280
00281   /* use base class constructors */
00282   using base_vec::base_vec;
00283
00284 /** Accessors to the first element of a vector
00285 */
00286   const DataType& x()const {
00287     return (*this)[0];
00288   }
00289
00290   DataType& x() {
00291     return (*this)[0];
00292   }
00293
00294 /** Accessors to the second element of a vector
00295 */
00296   const DataType& y()const {
00297     return (*this)[1];
00298   }
00299
00300   DataType& y() {
00301     return (*this)[1];
00302   }
00303
00304 /** Accessors to the third element of a vector
00305 */
00306   const DataType& z()const {
00307      return (*this)[2];
00308   }
00309
00310   DataType& z() {
00311     return (*this)[2];
00312   }
00313
00314 /** Accessors to the fourth element of a vector
00315 */
00316   const DataType& w()const {
00317      return (*this)[3];
00318   }
00319
00320   DataType& w() {
00321     return (*this)[3];
00322   }
00323
00324 /** Accessors to the first element of a vector
00325 */
00326   const DataType& r()const {
00327      return (*this)[0];
00328   }
00329
00330   DataType& r() {
00331     return (*this)[0];
00332   }
00333
00334 /** Accessors to the second element of a vector
00335 */
00336   const DataType& g()const {
00337      return (*this)[1];
00338   }
00339
00340   DataType& g() {
00341     return (*this)[1];
00342   }
00343
00344 /** Accessors to the third element of a vector
00345 */
00346   const DataType& b()const {
00347      return (*this)[2];
00348   }
00349
00350   DataType& b() {
00351     return (*this)[2];
00352   }
00353
00354 /** Accessors to the third element of a vector
00355 */
00356   const DataType& a()const {
00357      return (*this)[3];
00358   }
00359
00360   DataType& a() {
00361     return (*this)[3];
00362   }
00363
00364   TRISYCL_DECLARE_S(0);
00365   TRISYCL_DECLARE_S(1);
00366   TRISYCL_DECLARE_S(2);
```

```
00367   TRISYCL_DECLARE_S(3);
00368
00369   const __swizzled_vec__<DataType, 2> lo()const {
00370     return base_vec::swizzle(elem::s0, elem::s1);
00371   }
00372
00373   __swizzled_vec__<DataType, 2> lo()  {
00374     return base_vec::swizzle(elem::s0, elem::s1);
00375   }
00376
00377   const __swizzled_vec__<DataType, 2> hi()const {
00378     return base_vec::swizzle(elem::s2, elem::s3);
00379   }
00380
00381   __swizzled_vec__<DataType, 2> hi() {
00382     return base_vec::swizzle(elem::s2, elem::s3);
00383   }
00384
00385   const __swizzled_vec__<DataType, 2> odd()const {
00386     return base_vec::swizzle(elem::s1, elem::s3);
00387   }
00388
00389   __swizzled_vec__<DataType, 2> odd() {
00390     return base_vec::swizzle(elem::s1, elem::s3);
00391   }
00392
00393   const __swizzled_vec__<DataType, 2> even()const {
00394     return base_vec::swizzle(elem::s0, elem::s2);
00395   }
00396
00397   __swizzled_vec__<DataType, 2> even() {
00398     return base_vec::swizzle(elem::s0, elem::s2);
00399   }
00400 #include "triSYCL/vec/detail/swiz4.hpp"
00401 #include "triSYCL/vec/detail/swiz_rgba.hpp"
00402 };
00403
00404 #undef TRISYCL_GEN_SWIZ2
00405 #undef TRISYCL_GEN_SWIZ3
00406 #undef TRISYCL_GEN_SWIZ4
00407
00408 template<typename DataType>
00409 class alignas(detail::alignment_v<::trisycl::vec<DataType, 8»)
00410   vec<DataType, 8> :  public detail::vec<DataType, 8> {
00411   using base_vec = detail::vec<DataType, 8>;
00412
00413 public:
00414
00415   /* use base class constructors */
00416   using base_vec::base_vec;
00417
00418   TRISYCL_DECLARE_S(0);
00419   TRISYCL_DECLARE_S(1);
00420   TRISYCL_DECLARE_S(2);
00421   TRISYCL_DECLARE_S(3);
00422   TRISYCL_DECLARE_S(4);
00423   TRISYCL_DECLARE_S(5);
00424   TRISYCL_DECLARE_S(6);
00425   TRISYCL_DECLARE_S(7);
00426   TRISYCL_DECLARE_S(8);
00427
00428   const __swizzled_vec__<DataType, 4> lo()const {
00429     return base_vec::swizzle(elem::s0, elem::s1, elem::s2, elem::s3);
00430   }
00431
00432   __swizzled_vec__<DataType, 4> lo() {
00433     return base_vec::swizzle(elem::s0, elem::s1, elem::s2, elem::s3);
00434   }
00435
00436   const __swizzled_vec__<DataType, 4> hi()const {
00437     return base_vec::swizzle(elem::s4, elem::s5, elem::s6, elem::s7);
00438   }
00439
00440   __swizzled_vec__<DataType, 4> hi() {
00441     return base_vec::swizzle(elem::s4, elem::s5, elem::s6, elem::s7);
00442   }
00443
00444   const __swizzled_vec__<DataType, 4> odd()const {
00445     return base_vec::swizzle(elem::s1, elem::s3, elem::s5, elem::s7);
00446   }
00447
00448   __swizzled_vec__<DataType, 4> odd() {
00449     return base_vec::swizzle(elem::s1, elem::s3, elem::s5, elem::s7);
00450   }
00451
00452   const __swizzled_vec__<DataType, 4> even()const {
00453     return base_vec::swizzle(elem::s0, elem::s2, elem::s4, elem::s6);
```

```
00454   }
00455
00456   __swizzled_vec__<DataType, 4> even() {
00457     return base_vec::swizzle(elem::s0, elem::s2, elem::s4, elem::s6);
00458   }
00459 };
00460
00461
00462 template<typename DataType>
00463 class alignas(detail::alignment_v<::trisycl::vec<DataType, 16>)
00464   vec<DataType, 16> :  public detail::vec<DataType, 16> {
00465   using base_vec = detail::vec<DataType, 16>;
00466
00467 public:
00468
00469   /* use base class constructors */
00470   using base_vec::base_vec;
00471
00472   TRISYCL_DECLARE_S(0);
00473   TRISYCL_DECLARE_S(1);
00474   TRISYCL_DECLARE_S(2);
00475   TRISYCL_DECLARE_S(3);
00476   TRISYCL_DECLARE_S(4);
00477   TRISYCL_DECLARE_S(5);
00478   TRISYCL_DECLARE_S(6);
00479   TRISYCL_DECLARE_S(7);
00480   TRISYCL_DECLARE_S(8);
00481   TRISYCL_DECLARE_S(9);
00482   TRISYCL_DECLARE_S(A);
00483   TRISYCL_DECLARE_S(B);
00484   TRISYCL_DECLARE_S(C);
00485   TRISYCL_DECLARE_S(D);
00486   TRISYCL_DECLARE_S(E);
00487   TRISYCL_DECLARE_S(F);
00488
00489   const __swizzled_vec__<DataType, 8> lo()const {
00490     return base_vec::swizzle(elem::s0, elem::s1, elem::s2, elem::s3, elem::s4,
00491                              elem::s5, elem::s6, elem::s7);
00492   }
00493
00494   __swizzled_vec__<DataType, 8> lo() {
00495     return base_vec::swizzle(elem::s0, elem::s1, elem::s2, elem::s3, elem::s4,
00496                              elem::s5, elem::s6, elem::s7);
00497   }
00498
00499   const __swizzled_vec__<DataType, 8> hi()const {
00500     return base_vec::swizzle(elem::s8, elem::s9, elem::sA, elem::sB, elem::sC,
00501                              elem::sD, elem::sE, elem::sF);
00502   }
00503
00504   __swizzled_vec__<DataType, 8> hi() {
00505     return base_vec::swizzle(elem::s8, elem::s9, elem::sA, elem::sB, elem::sC,
00506                              elem::sD, elem::sE, elem::sF);
00507   }
00508
00509   const __swizzled_vec__<DataType, 8> odd()const {
00510     return base_vec::swizzle(elem::s1, elem::s3, elem::s5, elem::s7, elem::s9,
00511                              elem::sB, elem::sD, elem::sF);
00512   }
00513
00514   __swizzled_vec__<DataType, 8> odd() {
00515     return base_vec::swizzle(elem::s1, elem::s3, elem::s5, elem::s7, elem::s9,
00516                              elem::sB, elem::sD, elem::sF);
00517   }
00518
00519   const __swizzled_vec__<DataType, 8> even()const {
00520     return base_vec::swizzle(elem::s0, elem::s2, elem::s4, elem::s6, elem::s8,
00521                              elem::sA, elem::sC, elem::sE);
00522   }
00523
00524   __swizzled_vec__<DataType, 8> even() {
00525     return base_vec::swizzle(elem::s0, elem::s2, elem::s4, elem::s6, elem::s8,
00526                              elem::sA, elem::sC, elem::sE);
00527   }
00528
00529 };
00530
00531 #undef TRISYCL_DECLARE_S
00532
00533 /** A macro to define type alias, such as for type=uchar, size=4 and
00534 actual_type=unsigned char, uchar4 is equivalent to vec<unsigned char, 4>
00535 */
00536 #define TRISYCL_DEFINE_VEC_TYPE_SIZE(type, size, actual_type) \
00537 using type##size = vec<actual_type, size>;
00538
00539 /// Declare the vector types of a type for all the sizes
00540 #define TRISYCL_DEFINE_VEC_TYPE(type, actual_type)           \
```

```
00541 TRISYCL_DEFINE_VEC_TYPE_SIZE(type, 1, actual_type)          \
00542 TRISYCL_DEFINE_VEC_TYPE_SIZE(type, 2, actual_type)          \
00543 TRISYCL_DEFINE_VEC_TYPE_SIZE(type, 3, actual_type)          \
00544 TRISYCL_DEFINE_VEC_TYPE_SIZE(type, 4, actual_type)          \
00545 TRISYCL_DEFINE_VEC_TYPE_SIZE(type, 8, actual_type)          \
00546 TRISYCL_DEFINE_VEC_TYPE_SIZE(type, 16, actual_type)
00547
00548 /// Declare all the possible vector type aliases
00549   TRISYCL_DEFINE_VEC_TYPE(char, char)
00550   TRISYCL_DEFINE_VEC_TYPE(uchar, unsigned char)
00551   TRISYCL_DEFINE_VEC_TYPE(short, short int)
00552   TRISYCL_DEFINE_VEC_TYPE(ushort, unsigned short int)
00553   TRISYCL_DEFINE_VEC_TYPE(int, int)
00554   TRISYCL_DEFINE_VEC_TYPE(uint, unsigned int)
00555   TRISYCL_DEFINE_VEC_TYPE(long, long int)
00556   TRISYCL_DEFINE_VEC_TYPE(ulong, unsigned long int)
00557   TRISYCL_DEFINE_VEC_TYPE(float, float)
00558   TRISYCL_DEFINE_VEC_TYPE(double, double)
00559
00560 /// @} End the vector Doxygen group
00561
00562 }
00563
00564
00565 namespace std {
00566 /* Provide a tuple-like interface to the SYCL vec with some type
00567 traits and function template specialization in std::, besides the
00568 get<> member function.
00569
00570 This is useful to have a data type compatible with structure binding
00571 */
00572
00573 /// The std::tuple_element specialization for vec
00574 template<std::size_t I, typename T, int Size>
00575 struct tuple_element<I, trisycl::vec<T, Size>> {
00576   static_assert(I < Size, "The vec<> has not enough elements!");
00577   // All the elements have the same type
00578   using type = T;
00579 };
00580
00581
00582 /// The std::tuple_size specialization for vec
00583 template <typename T, int Size>
00584 struct tuple_size<trisycl::vec<T, Size>>
00585   :  integral_constant<size_t, Size>
00586 { };
00587
00588
00589 /** Provide a get<> accessor to have C++17 structured binding working
00590
00591 Note that it works without this with current std::array based
00592 implementation on the host device but we prefer to have a more
00593 realistic const binding
00594 */
00595 template <int I, typename T, int Size>
00596 auto get(const trisycl::vec<T, Size>& v) {
00597   static_assert(I < Size, "The vec<> has not enough elements!");
00598   return v[I];
00599 }
00600
00601 }
00602
00603 /*
00604 # Some Emacs stuff:
00605 ### Local Variables:
00606 ### ispell-local-dictionary:  "american"
00607 ### eval:  (flyspell-prog-mode)
00608 ### End:
00609 */
00610
00611 #endif // TRISYCL_SYCL_VEC_HPP
```

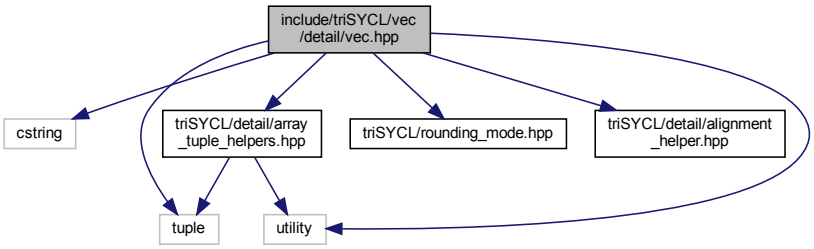
13.193 include/triSYCL/vec/detail/vec.hpp File Reference

Implement the detail base SYCL vector class.

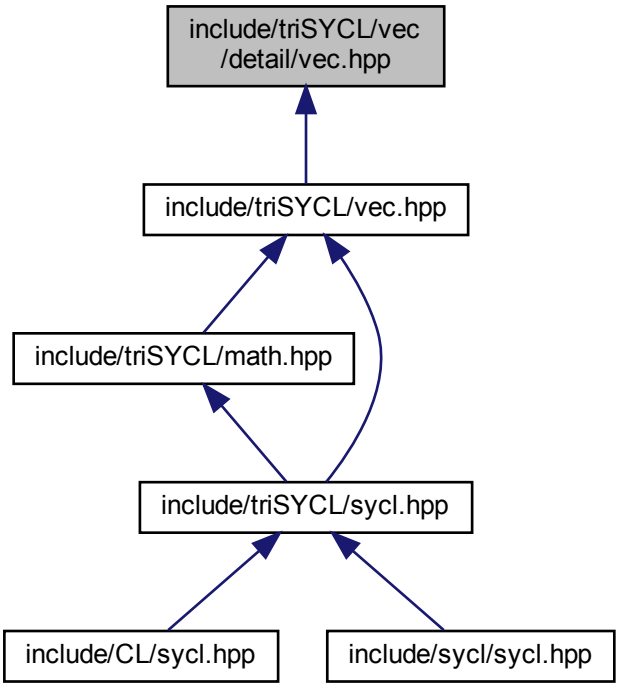
```
#include <cstring>
#include <tuple>
#include <utility>
```

```
#include "triSYCL/rounding_mode.hpp"
#include "triSYCL/detail/alignment_helper.hpp"
#include "triSYCL/detail/array_tuple_helpers.hpp"
```

Include dependency graph for vec.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::vec< DataType, NumElements >

  forward declaration to use in the detail class

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Typedefs

- template<typename DataType , int numElements>
  using trisycl::detail::__swizzled_base_vec__ = vec< DataType, numElements >

13.193.1 Detailed Description

Implement the detail base SYCL vector class.

Ronan at Keryell point FR Dave Airlie

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file vec.hpp.

13.194 vec.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VEC_DETAIL_VEC_HPP
00002 #define TRISYCL_SYCL_VEC_DETAIL_VEC_HPP
00003
00004 #include <cstring>
00005 #include <tuple>
00006 #include <utility>
00007
00008 /** \file
00009
00010 Implement the detail base SYCL vector class
00011
00012 Ronan at Keryell point FR
00013 Dave Airlie
00014
00015 This file is distributed under the University of Illinois Open Source
00016 License.  See LICENSE.TXT for details.
00017 */
00018
00019 #include "triSYCL/rounding_mode.hpp"
00020 #include "triSYCL/detail/alignment_helper.hpp"
00021 #include "triSYCL/detail/array_tuple_helpers.hpp"
00022
00023 namespace trisycl {
00024
00025 template <typename DataType, int numElements>
00026 using __swizzled_vec__ = vec<DataType, numElements>;
00027
00028 }
00029
00030 namespace trisycl::detail {
00031
00032 /// forward declaration to use in the detail class
00033 template <typename, int>
00034 class vec;
00035
00036 template <typename DataType, int numElements>
00037 using __swizzled_base_vec__ = vec<DataType, numElements>;
00038
00039 /// Small SYCL vector class
00040 template <typename DataType, int NumElements>
00041 class alignas(alignment_v<::trisycl::vec<DataType, NumElements>>)
```

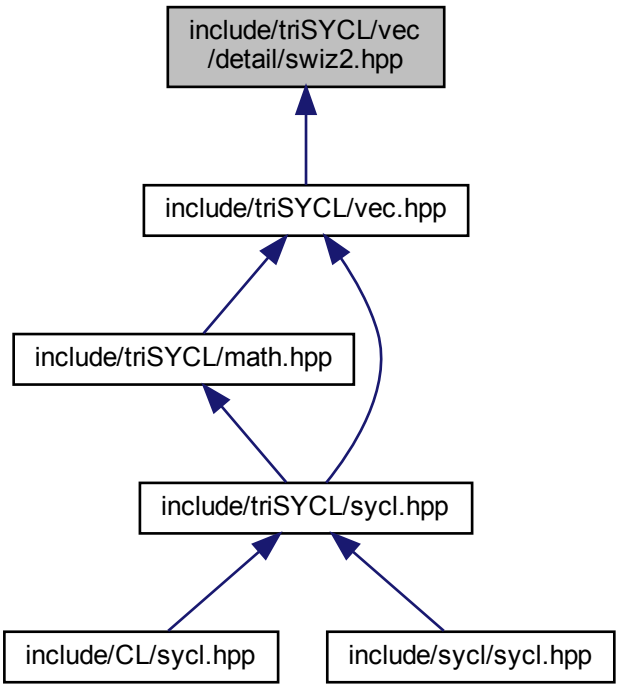
```
00042   vec :  public detail::small_array<DataType,
00043                                     ::trisycl::vec<DataType, NumElements>,
00044                                     NumElements> {
00045   using basic_type = typename detail::small_array<DataType,
00046                                                   ::trisycl::vec<DataType,
00047                                                                  NumElements>,
00048                                                   NumElements>;
00049
00050  public:
00051 /** Construct a vec from anything from a scalar (to initialize all the
00052 elements with this value) up to an aggregate of scalar and vector
00053 types (in this case the total number of elements must match the size
00054 of the vector).
00055
00056 Do not use this constructor with NumElements to avoid ambiguity
00057 with the inherited small_array constructor.
00058 */
00059   template <typename...  Types>
00060     requires(sizeof...(Types) != NumElements)
00061   vec(const Types...  args)
00062       :  basic_type { detail::expand<vec>(flatten_to_tuple<vec>(args...))  } {}
00063
00064 /// Use classical constructors too
00065   vec() = default;
00066
00067
00068   // Inherit of all the constructors
00069   using basic_type::basic_type;
00070
00071 private:
00072
00073 /** Flattening helper that does not change scalar values but flatten a
00074 vec<T, n> v into a tuple<T, T,..., T>{ v[0], v[1],..., v[n-1] }
00075
00076 If we have a vector, just forward its array content since an array
00077 has also a tuple interface :-) (23.3.2.9 Tuple interface to class
00078 template array [array.tuple])
00079 */
00080   template <typename V, typename Element, int s>
00081   static auto flatten(const vec<Element, s> i) {
00082     static_assert(s <= V::dimension,
00083                   "The element i will not fit in the vector");
00084     return static_cast<std::array<Element, s>>(i);
00085   }
00086
00087   template <typename V, typename Element, int s>
00088   static auto flatten(const ::trisycl::vec<Element, s> i) {
00089     static_assert(s <= V::dimension,
00090                   "The element i will not fit in the vector");
00091     return static_cast<std::array<Element, s>>(i);
00092   }
00093
00094
00095 /** If we do not have a vector, just forward it as a tuple up to the
00096 final initialization.
00097
00098 \return typically tuple<double>{ 2.4 } from 2.4 input for example
00099 */
00100   template <typename V, typename Type>
00101   static auto flatten(const Type i) {
00102     return std::make_tuple(i);
00103   }
00104
00105
00106 /** Take some initializer values and apply flattening on each value
00107
00108 \return a tuple of scalar initializer values
00109 */
00110   template <typename V, typename...  Types>
00111   static auto flatten_to_tuple(const Types...  i) {
00112     // Concatenate the tuples returned by each flattening
00113     return std::tuple_cat(flatten<V>(i)...);
00114   }
00115
00116
00117   /* apply_unary_functor_impl & apply_unary_functor generate a sequence from
00118 a passed in vector and then execute a function across the range of the
00119 vector returning the result as a new vector.
00120 */
00121   template <typename T, typename F, std::size_t...  Is>
00122   static auto apply_unary_functor_impl(T tuple, F f,
00123                                        std::index_sequence<Is...>) {
00124     return ::trisycl::vec<DataType, NumElements>{f(std::get<Is>(tuple))...};
00125   }
00126
00127   template<typename T, typename F>
00128   static auto apply_unary_functor(const T x, F f) {
```

```
00129     auto tuple = flatten_to_tuple<T>(x);
00130     return apply_unary_functor_impl(tuple, f,
00131         std::make_index_sequence<std::tuple_size<decltype(tuple)>::value>());
00132   }
00133
00134
00135   /* apply_binary_functor_impl & apply_binary_functor generate two sequences
00136 from a passed in vector and then execute a function across the range of
00137 vector x returning the result as a new vector.
00138 */
00139   template<typename T, typename T2, typename F, std::size_t...  Is>
00140   static auto apply_binary_functor_impl(T xTuple, T2 yTuple, F f,
00141     std::index_sequence<Is...>) {
00142     return ::trisycl::vec<DataType, NumElements>{f(std::get<Is>(xTuple),
00143        std::get<Is>(yTuple))...};
00144   }
00145
00146
00147   template<typename T, typename T2, typename T3, typename F, std::size_t...  Is>
00148   static auto apply_ternary_functor_impl(T xTuple, T2 yTuple, T3 zTuple, F f,
00149     std::index_sequence<Is...>) {
00150     return ::trisycl::vec<DataType, NumElements>{f(std::get<Is>(xTuple),
00151        std::get<Is>(yTuple),  std::get<Is>(zTuple))...};
00152   }
00153
00154
00155   template<typename T, typename T2, typename F>
00156   static auto apply_binary_functor(const T x, const T2 y, F f) {
00157     auto xTuple = flatten_to_tuple<T>(x);
00158     auto yTuple = flatten_to_tuple<T>(y);
00159     return apply_binary_functor_impl(xTuple, yTuple, f,
00160         std::make_index_sequence<std::tuple_size<decltype(xTuple)>::value>());
00161   }
00162
00163 protected:
00164
00165 /// Apply a swizzle operation from swizzleIndexes parameters
00166   template<typename...  Ts>
00167   __swizzled_base_vec__<DataType, sizeof...(Ts)>
00168   swizzle(Ts...  swizzleIndexes) const {
00169     // Construct a new vector from an elemental swizzle of each element
00170     return __swizzled_base_vec__<DataType, sizeof...(Ts)> {
00171       (*this)[swizzleIndexes]...  };
00172   }
00173
00174 public:
00175
00176 /// Return the number of elements in the vector
00177   auto get_count()const {
00178     return NumElements;
00179   }
00180
00181 /// Return the number of bytes
00182   auto get_size()const {
00183     // The alignment for the vec classes are equivalent to the vec's size e.g.
00184     // a vec of 4 floats would be aligned to 16 bytes which is also the vec's
00185     // size.
00186     return alignment_v<::trisycl::vec<DataType, NumElements»;
00187   }
00188
00189
00190   template<typename convertT, rounding_mode roundingMode>
00191   vec<convertT, NumElements> convert()const {
00192     vec<convertT, NumElements> result;
00193
00194     assert(result.get_count() == this->get_count());
00195     for (int n = 0; n < NumElements; n++) {
00196       result[n] = (*this)[n];
00197     }
00198     return result;
00199   };
00200
00201
00202   template<typename asT> asT as()const {
00203     asT result;
00204     assert(result.get_size() == this->get_size());
00205     std::memcpy(result.data(), this->data(), this->get_size());
00206     return result;
00207   };
00208
00209
00210   // Swizzle methods (see notes)
00211   template <int...  swizzleIndexs>
00212   __swizzled_vec__<DataType, sizeof...(swizzleIndexs)> swizzle() const {
00213     return static_cast<__swizzled_vec__<DataType, sizeof...(swizzleIndexs)>>(
00214         swizzle(swizzleIndexs...));
00215   }
```

```
00216
00217
00218   // Applies a function across each element of the vector to generate a new
00219   // vector, it leaves the original vector unaltered.
00220   template <typename F>
00221   auto map(F f)const {
00222     return apply_unary_functor(*this, f);
00223   }
00224
00225
00226   // Applies a binary function across each element of the calling vector and the
00227   // passed in vector x to generate a new vector.  It leaves both the original
00228   // vectors unaltered.
00229   //
00230   // Note:  could in theory be made less restrictive and allow zipping with a
00231   // scalar type for x and allow zipping across two differing sized vectors.
00232   // It would require an extra apply_binary_functor overload and a test to find
00233   // the smallest index_sequence between the calling vector and x.
00234   template <typename T, int size, typename F>
00235   auto zip(::trisycl::vec<T, size> x, F f)const {
00236     static_assert(size == NumElements,
00237       "zip currently does not support vec's of differing element size");
00238     return apply_binary_functor(*this, x, f);
00239   }
00240
00241
00242 /// \todo this should be generalized to any arity
00243   template <typename T, int size, typename F>
00244   auto zip(::trisycl::vec<T, size> y, ::trisycl::vec<T, size> z, F f)const {
00245     static_assert(size == NumElements,
00246       "zip currently does not support vec's of differing element size");
00247     return apply_ternary_functor(*this, y, z, f);
00248   }
00249
00250 };
00251
00252
00253 };
00254
00255 #endif
```

13.195 include/triSYCL/vec/detail/swiz2.hpp File Reference

This graph shows which files directly or indirectly include this file:



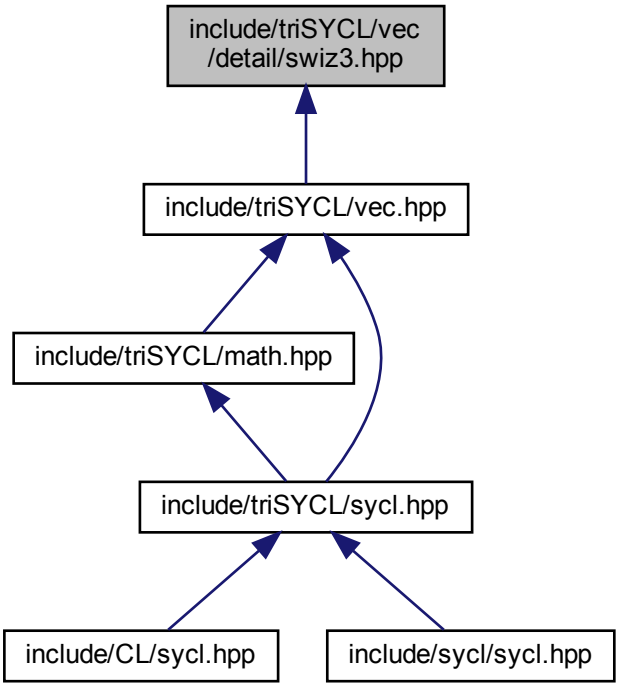
13.196 swiz2.hpp

Go to the documentation of this file.

```
00001 // Generated using a simple C generator
00002 #ifdef SYCL_SIMPLE_SWIZZLES
00003 TRISYCL_GEN_SWIZ2(xx, 0, 0)
00004 TRISYCL_GEN_SWIZ2(xy, 0, 1)
00005 TRISYCL_GEN_SWIZ2(yx, 1, 0)
00006 TRISYCL_GEN_SWIZ2(yy, 1, 1)
00007 #endif
```

13.197 include/triSYCL/vec/detail/swiz3.hpp File Reference

This graph shows which files directly or indirectly include this file:



13.198 swiz3.hpp

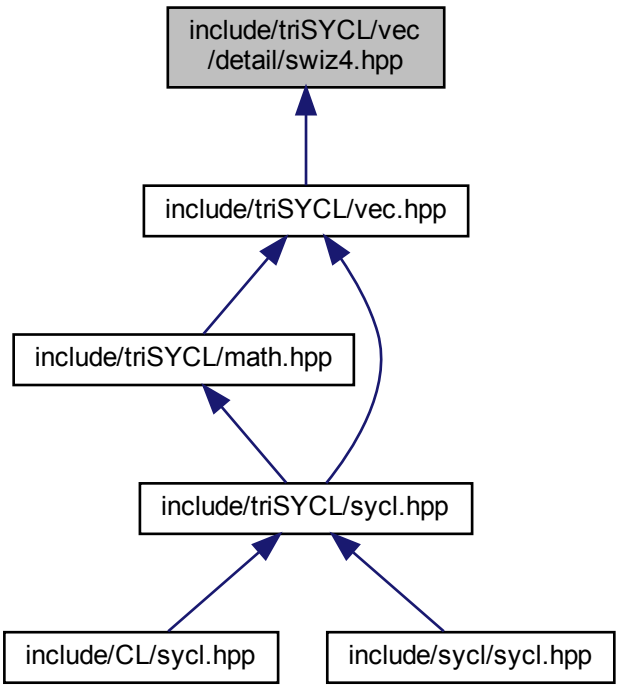
Go to the documentation of this file.

```
00001 // Generated using a simple C generator
00002 #ifdef SYCL_SIMPLE_SWIZZLES
00003 TRISYCL_GEN_SWIZ2(xx, 0, 0)
00004 TRISYCL_GEN_SWIZ2(xy, 0, 1)
00005 TRISYCL_GEN_SWIZ2(yx, 1, 0)
00006 TRISYCL_GEN_SWIZ2(yy, 1, 1)
00007 TRISYCL_GEN_SWIZ3(xxx, 0, 0, 0)
00008 TRISYCL_GEN_SWIZ3(xxy, 0, 0, 1)
00009 TRISYCL_GEN_SWIZ3(xxz, 0, 0, 2)
00010 TRISYCL_GEN_SWIZ3(xyx, 0, 1, 0)
00011 TRISYCL_GEN_SWIZ3(xyy, 0, 1, 1)
00012 TRISYCL_GEN_SWIZ3(xyz, 0, 1, 2)
00013 TRISYCL_GEN_SWIZ3(xzx, 0, 2, 0)
00014 TRISYCL_GEN_SWIZ3(xzy, 0, 2, 1)
00015 TRISYCL_GEN_SWIZ3(xzz, 0, 2, 2)
00016 TRISYCL_GEN_SWIZ3(yxx, 1, 0, 0)
00017 TRISYCL_GEN_SWIZ3(yxy, 1, 0, 1)
00018 TRISYCL_GEN_SWIZ3(yxz, 1, 0, 2)
00019 TRISYCL_GEN_SWIZ3(yyx, 1, 1, 0)
00020 TRISYCL_GEN_SWIZ3(yyy, 1, 1, 1)
00021 TRISYCL_GEN_SWIZ3(yyz, 1, 1, 2)
00022 TRISYCL_GEN_SWIZ3(yzx, 1, 2, 0)
00023 TRISYCL_GEN_SWIZ3(yzy, 1, 2, 1)
00024 TRISYCL_GEN_SWIZ3(yzz, 1, 2, 2)
00025 TRISYCL_GEN_SWIZ3(zxx, 2, 0, 0)
00026 TRISYCL_GEN_SWIZ3(zxy, 2, 0, 1)
00027 TRISYCL_GEN_SWIZ3(zxz, 2, 0, 2)
```

```
00028 TRISYCL_GEN_SWIZ3(zyx, 2, 1, 0)
00029 TRISYCL_GEN_SWIZ3(zyy, 2, 1, 1)
00030 TRISYCL_GEN_SWIZ3(zyz, 2, 1, 2)
00031 TRISYCL_GEN_SWIZ3(zzx, 2, 2, 0)
00032 TRISYCL_GEN_SWIZ3(zzy, 2, 2, 1)
00033 TRISYCL_GEN_SWIZ3(zzz, 2, 2, 2)
00034 #endif
```

13.199 include/triSYCL/vec/detail/swiz4.hpp File Reference

This graph shows which files directly or indirectly include this file:



13.200 swiz4.hpp

Go to the documentation of this file.

```
00001 // Generated using a simple C generator
00002 #ifdef SYCL_SIMPLE_SWIZZLES
00003 TRISYCL_GEN_SWIZ2(xx, 0, 0)
00004 TRISYCL_GEN_SWIZ2(xy, 0, 1)
00005 TRISYCL_GEN_SWIZ2(yx, 1, 0)
00006 TRISYCL_GEN_SWIZ2(yy, 1, 1)
00007 TRISYCL_GEN_SWIZ3(xxx, 0, 0, 0)
00008 TRISYCL_GEN_SWIZ3(xxy, 0, 0, 1)
00009 TRISYCL_GEN_SWIZ3(xxz, 0, 0, 2)
00010 TRISYCL_GEN_SWIZ3(xyx, 0, 1, 0)
00011 TRISYCL_GEN_SWIZ3(xyy, 0, 1, 1)
00012 TRISYCL_GEN_SWIZ3(xyz, 0, 1, 2)
00013 TRISYCL_GEN_SWIZ3(xzx, 0, 2, 0)
00014 TRISYCL_GEN_SWIZ3(xzy, 0, 2, 1)
00015 TRISYCL_GEN_SWIZ3(xzz, 0, 2, 2)
```

```
00016 TRISYCL_GEN_SWIZ3(yxx, 1, 0, 0)
00017 TRISYCL_GEN_SWIZ3(yxy, 1, 0, 1)
00018 TRISYCL_GEN_SWIZ3(yxz, 1, 0, 2)
00019 TRISYCL_GEN_SWIZ3(yyx, 1, 1, 0)
00020 TRISYCL_GEN_SWIZ3(yyy, 1, 1, 1)
00021 TRISYCL_GEN_SWIZ3(yyz, 1, 1, 2)
00022 TRISYCL_GEN_SWIZ3(yzx, 1, 2, 0)
00023 TRISYCL_GEN_SWIZ3(yzy, 1, 2, 1)
00024 TRISYCL_GEN_SWIZ3(yzz, 1, 2, 2)
00025 TRISYCL_GEN_SWIZ3(zxx, 2, 0, 0)
00026 TRISYCL_GEN_SWIZ3(zxy, 2, 0, 1)
00027 TRISYCL_GEN_SWIZ3(zxz, 2, 0, 2)
00028 TRISYCL_GEN_SWIZ3(zyx, 2, 1, 0)
00029 TRISYCL_GEN_SWIZ3(zyy, 2, 1, 1)
00030 TRISYCL_GEN_SWIZ3(zyz, 2, 1, 2)
00031 TRISYCL_GEN_SWIZ3(zzx, 2, 2, 0)
00032 TRISYCL_GEN_SWIZ3(zzy, 2, 2, 1)
00033 TRISYCL_GEN_SWIZ3(zzz, 2, 2, 2)
00034 TRISYCL_GEN_SWIZ4(xxxx, 0, 0, 0, 0)
00035 TRISYCL_GEN_SWIZ4(xxxy, 0, 0, 0, 1)
00036 TRISYCL_GEN_SWIZ4(xxxz, 0, 0, 0, 2)
00037 TRISYCL_GEN_SWIZ4(xxxw, 0, 0, 0, 3)
00038 TRISYCL_GEN_SWIZ4(xxyx, 0, 0, 1, 0)
00039 TRISYCL_GEN_SWIZ4(xxyy, 0, 0, 1, 1)
00040 TRISYCL_GEN_SWIZ4(xxyz, 0, 0, 1, 2)
00041 TRISYCL_GEN_SWIZ4(xxyw, 0, 0, 1, 3)
00042 TRISYCL_GEN_SWIZ4(xxzx, 0, 0, 2, 0)
00043 TRISYCL_GEN_SWIZ4(xxzy, 0, 0, 2, 1)
00044 TRISYCL_GEN_SWIZ4(xxzz, 0, 0, 2, 2)
00045 TRISYCL_GEN_SWIZ4(xxzw, 0, 0, 2, 3)
00046 TRISYCL_GEN_SWIZ4(xxwx, 0, 0, 3, 0)
00047 TRISYCL_GEN_SWIZ4(xxwy, 0, 0, 3, 1)
00048 TRISYCL_GEN_SWIZ4(xxwz, 0, 0, 3, 2)
00049 TRISYCL_GEN_SWIZ4(xxww, 0, 0, 3, 3)
00050 TRISYCL_GEN_SWIZ4(xyxx, 0, 1, 0, 0)
00051 TRISYCL_GEN_SWIZ4(xyxy, 0, 1, 0, 1)
00052 TRISYCL_GEN_SWIZ4(xyxz, 0, 1, 0, 2)
00053 TRISYCL_GEN_SWIZ4(xyxw, 0, 1, 0, 3)
00054 TRISYCL_GEN_SWIZ4(xyyx, 0, 1, 1, 0)
00055 TRISYCL_GEN_SWIZ4(xyyy, 0, 1, 1, 1)
00056 TRISYCL_GEN_SWIZ4(xyyz, 0, 1, 1, 2)
00057 TRISYCL_GEN_SWIZ4(xyyw, 0, 1, 1, 3)
00058 TRISYCL_GEN_SWIZ4(xyzx, 0, 1, 2, 0)
00059 TRISYCL_GEN_SWIZ4(xyzy, 0, 1, 2, 1)
00060 TRISYCL_GEN_SWIZ4(xyzz, 0, 1, 2, 2)
00061 TRISYCL_GEN_SWIZ4(xyzw, 0, 1, 2, 3)
00062 TRISYCL_GEN_SWIZ4(xywx, 0, 1, 3, 0)
00063 TRISYCL_GEN_SWIZ4(xywy, 0, 1, 3, 1)
00064 TRISYCL_GEN_SWIZ4(xywz, 0, 1, 3, 2)
00065 TRISYCL_GEN_SWIZ4(xyww, 0, 1, 3, 3)
00066 TRISYCL_GEN_SWIZ4(xzxx, 0, 2, 0, 0)
00067 TRISYCL_GEN_SWIZ4(xzxy, 0, 2, 0, 1)
00068 TRISYCL_GEN_SWIZ4(xzxz, 0, 2, 0, 2)
00069 TRISYCL_GEN_SWIZ4(xzxw, 0, 2, 0, 3)
00070 TRISYCL_GEN_SWIZ4(xzyx, 0, 2, 1, 0)
00071 TRISYCL_GEN_SWIZ4(xzyy, 0, 2, 1, 1)
00072 TRISYCL_GEN_SWIZ4(xzyz, 0, 2, 1, 2)
00073 TRISYCL_GEN_SWIZ4(xzyw, 0, 2, 1, 3)
00074 TRISYCL_GEN_SWIZ4(xzzx, 0, 2, 2, 0)
00075 TRISYCL_GEN_SWIZ4(xzzy, 0, 2, 2, 1)
00076 TRISYCL_GEN_SWIZ4(xzzz, 0, 2, 2, 2)
00077 TRISYCL_GEN_SWIZ4(xzzw, 0, 2, 2, 3)
00078 TRISYCL_GEN_SWIZ4(xzwx, 0, 2, 3, 0)
00079 TRISYCL_GEN_SWIZ4(xzwy, 0, 2, 3, 1)
00080 TRISYCL_GEN_SWIZ4(xzwz, 0, 2, 3, 2)
00081 TRISYCL_GEN_SWIZ4(xzww, 0, 2, 3, 3)
00082 TRISYCL_GEN_SWIZ4(xwxx, 0, 3, 0, 0)
00083 TRISYCL_GEN_SWIZ4(xwxy, 0, 3, 0, 1)
00084 TRISYCL_GEN_SWIZ4(xwxz, 0, 3, 0, 2)
00085 TRISYCL_GEN_SWIZ4(xwxw, 0, 3, 0, 3)
00086 TRISYCL_GEN_SWIZ4(xwyx, 0, 3, 1, 0)
00087 TRISYCL_GEN_SWIZ4(xwyy, 0, 3, 1, 1)
00088 TRISYCL_GEN_SWIZ4(xwyz, 0, 3, 1, 2)
00089 TRISYCL_GEN_SWIZ4(xwyw, 0, 3, 1, 3)
00090 TRISYCL_GEN_SWIZ4(xwzx, 0, 3, 2, 0)
00091 TRISYCL_GEN_SWIZ4(xwzy, 0, 3, 2, 1)
00092 TRISYCL_GEN_SWIZ4(xwzz, 0, 3, 2, 2)
00093 TRISYCL_GEN_SWIZ4(xwzw, 0, 3, 2, 3)
00094 TRISYCL_GEN_SWIZ4(xwwx, 0, 3, 3, 0)
00095 TRISYCL_GEN_SWIZ4(xwwy, 0, 3, 3, 1)
00096 TRISYCL_GEN_SWIZ4(xwwz, 0, 3, 3, 2)
00097 TRISYCL_GEN_SWIZ4(xwww, 0, 3, 3, 3)
00098 TRISYCL_GEN_SWIZ4(yxxx, 1, 0, 0, 0)
00099 TRISYCL_GEN_SWIZ4(yxxy, 1, 0, 0, 1)
00100 TRISYCL_GEN_SWIZ4(yxxz, 1, 0, 0, 2)
00101 TRISYCL_GEN_SWIZ4(yxxw, 1, 0, 0, 3)
00102 TRISYCL_GEN_SWIZ4(yxyx, 1, 0, 1, 0)
```

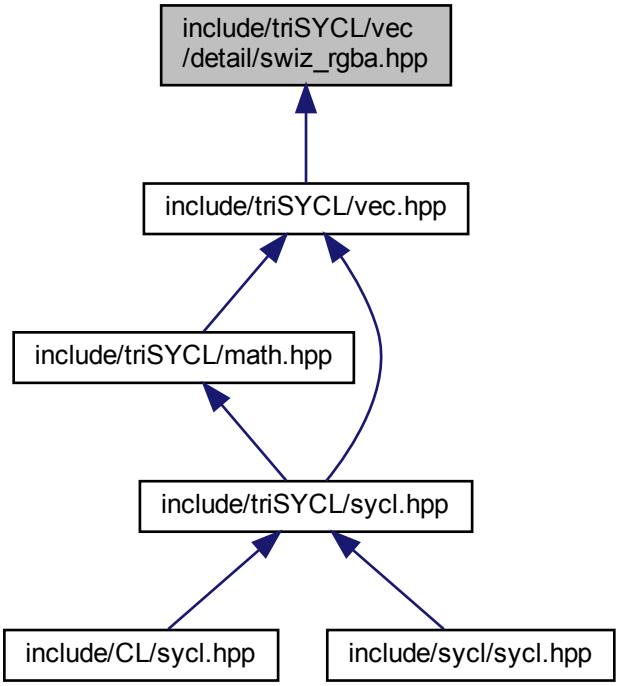
```
00103 TRISYCL_GEN_SWIZ4(yxyy, 1, 0, 1, 1)
00104 TRISYCL_GEN_SWIZ4(yxyz, 1, 0, 1, 2)
00105 TRISYCL_GEN_SWIZ4(yxyw, 1, 0, 1, 3)
00106 TRISYCL_GEN_SWIZ4(yxzx, 1, 0, 2, 0)
00107 TRISYCL_GEN_SWIZ4(yxzy, 1, 0, 2, 1)
00108 TRISYCL_GEN_SWIZ4(yxzz, 1, 0, 2, 2)
00109 TRISYCL_GEN_SWIZ4(yxzw, 1, 0, 2, 3)
00110 TRISYCL_GEN_SWIZ4(yxwx, 1, 0, 3, 0)
00111 TRISYCL_GEN_SWIZ4(yxwy, 1, 0, 3, 1)
00112 TRISYCL_GEN_SWIZ4(yxwz, 1, 0, 3, 2)
00113 TRISYCL_GEN_SWIZ4(yxww, 1, 0, 3, 3)
00114 TRISYCL_GEN_SWIZ4(yyxx, 1, 1, 0, 0)
00115 TRISYCL_GEN_SWIZ4(yyxy, 1, 1, 0, 1)
00116 TRISYCL_GEN_SWIZ4(yyxz, 1, 1, 0, 2)
00117 TRISYCL_GEN_SWIZ4(yyxw, 1, 1, 0, 3)
00118 TRISYCL_GEN_SWIZ4(yyyx, 1, 1, 1, 0)
00119 TRISYCL_GEN_SWIZ4(yyyy, 1, 1, 1, 1)
00120 TRISYCL_GEN_SWIZ4(yyyz, 1, 1, 1, 2)
00121 TRISYCL_GEN_SWIZ4(yyyw, 1, 1, 1, 3)
00122 TRISYCL_GEN_SWIZ4(yyzx, 1, 1, 2, 0)
00123 TRISYCL_GEN_SWIZ4(yyzy, 1, 1, 2, 1)
00124 TRISYCL_GEN_SWIZ4(yyzz, 1, 1, 2, 2)
00125 TRISYCL_GEN_SWIZ4(yyzw, 1, 1, 2, 3)
00126 TRISYCL_GEN_SWIZ4(yywx, 1, 1, 3, 0)
00127 TRISYCL_GEN_SWIZ4(yywy, 1, 1, 3, 1)
00128 TRISYCL_GEN_SWIZ4(yywz, 1, 1, 3, 2)
00129 TRISYCL_GEN_SWIZ4(yyww, 1, 1, 3, 3)
00130 TRISYCL_GEN_SWIZ4(yzxx, 1, 2, 0, 0)
00131 TRISYCL_GEN_SWIZ4(yzxy, 1, 2, 0, 1)
00132 TRISYCL_GEN_SWIZ4(yzxz, 1, 2, 0, 2)
00133 TRISYCL_GEN_SWIZ4(yzxw, 1, 2, 0, 3)
00134 TRISYCL_GEN_SWIZ4(yzyx, 1, 2, 1, 0)
00135 TRISYCL_GEN_SWIZ4(yzyy, 1, 2, 1, 1)
00136 TRISYCL_GEN_SWIZ4(yzyz, 1, 2, 1, 2)
00137 TRISYCL_GEN_SWIZ4(yzyw, 1, 2, 1, 3)
00138 TRISYCL_GEN_SWIZ4(yzzx, 1, 2, 2, 0)
00139 TRISYCL_GEN_SWIZ4(yzzy, 1, 2, 2, 1)
00140 TRISYCL_GEN_SWIZ4(yzzz, 1, 2, 2, 2)
00141 TRISYCL_GEN_SWIZ4(yzzw, 1, 2, 2, 3)
00142 TRISYCL_GEN_SWIZ4(yzwx, 1, 2, 3, 0)
00143 TRISYCL_GEN_SWIZ4(yzwy, 1, 2, 3, 1)
00144 TRISYCL_GEN_SWIZ4(yzwz, 1, 2, 3, 2)
00145 TRISYCL_GEN_SWIZ4(yzww, 1, 2, 3, 3)
00146 TRISYCL_GEN_SWIZ4(ywxx, 1, 3, 0, 0)
00147 TRISYCL_GEN_SWIZ4(ywxy, 1, 3, 0, 1)
00148 TRISYCL_GEN_SWIZ4(ywxz, 1, 3, 0, 2)
00149 TRISYCL_GEN_SWIZ4(ywxw, 1, 3, 0, 3)
00150 TRISYCL_GEN_SWIZ4(ywyx, 1, 3, 1, 0)
00151 TRISYCL_GEN_SWIZ4(ywyy, 1, 3, 1, 1)
00152 TRISYCL_GEN_SWIZ4(ywyz, 1, 3, 1, 2)
00153 TRISYCL_GEN_SWIZ4(ywyw, 1, 3, 1, 3)
00154 TRISYCL_GEN_SWIZ4(ywzx, 1, 3, 2, 0)
00155 TRISYCL_GEN_SWIZ4(ywzy, 1, 3, 2, 1)
00156 TRISYCL_GEN_SWIZ4(ywzz, 1, 3, 2, 2)
00157 TRISYCL_GEN_SWIZ4(ywzw, 1, 3, 2, 3)
00158 TRISYCL_GEN_SWIZ4(ywwx, 1, 3, 3, 0)
00159 TRISYCL_GEN_SWIZ4(ywwy, 1, 3, 3, 1)
00160 TRISYCL_GEN_SWIZ4(ywwz, 1, 3, 3, 2)
00161 TRISYCL_GEN_SWIZ4(ywww, 1, 3, 3, 3)
00162 TRISYCL_GEN_SWIZ4(zxxx, 2, 0, 0, 0)
00163 TRISYCL_GEN_SWIZ4(zxxy, 2, 0, 0, 1)
00164 TRISYCL_GEN_SWIZ4(zxxz, 2, 0, 0, 2)
00165 TRISYCL_GEN_SWIZ4(zxxw, 2, 0, 0, 3)
00166 TRISYCL_GEN_SWIZ4(zxyx, 2, 0, 1, 0)
00167 TRISYCL_GEN_SWIZ4(zxyy, 2, 0, 1, 1)
00168 TRISYCL_GEN_SWIZ4(zxyz, 2, 0, 1, 2)
00169 TRISYCL_GEN_SWIZ4(zxyw, 2, 0, 1, 3)
00170 TRISYCL_GEN_SWIZ4(zxzx, 2, 0, 2, 0)
00171 TRISYCL_GEN_SWIZ4(zxzy, 2, 0, 2, 1)
00172 TRISYCL_GEN_SWIZ4(zxzz, 2, 0, 2, 2)
00173 TRISYCL_GEN_SWIZ4(zxzw, 2, 0, 2, 3)
00174 TRISYCL_GEN_SWIZ4(zxwx, 2, 0, 3, 0)
00175 TRISYCL_GEN_SWIZ4(zxwy, 2, 0, 3, 1)
00176 TRISYCL_GEN_SWIZ4(zxwz, 2, 0, 3, 2)
00177 TRISYCL_GEN_SWIZ4(zxww, 2, 0, 3, 3)
00178 TRISYCL_GEN_SWIZ4(zyxx, 2, 1, 0, 0)
00179 TRISYCL_GEN_SWIZ4(zyxy, 2, 1, 0, 1)
00180 TRISYCL_GEN_SWIZ4(zyxz, 2, 1, 0, 2)
00181 TRISYCL_GEN_SWIZ4(zyxw, 2, 1, 0, 3)
00182 TRISYCL_GEN_SWIZ4(zyyx, 2, 1, 1, 0)
00183 TRISYCL_GEN_SWIZ4(zyyy, 2, 1, 1, 1)
00184 TRISYCL_GEN_SWIZ4(zyyz, 2, 1, 1, 2)
00185 TRISYCL_GEN_SWIZ4(zyyw, 2, 1, 1, 3)
00186 TRISYCL_GEN_SWIZ4(zyzx, 2, 1, 2, 0)
00187 TRISYCL_GEN_SWIZ4(zyzy, 2, 1, 2, 1)
00188 TRISYCL_GEN_SWIZ4(zyzz, 2, 1, 2, 2)
00189 TRISYCL_GEN_SWIZ4(zyzw, 2, 1, 2, 3)
```

```
00190 TRISYCL_GEN_SWIZ4(zywx, 2, 1, 3, 0)
00191 TRISYCL_GEN_SWIZ4(zywy, 2, 1, 3, 1)
00192 TRISYCL_GEN_SWIZ4(zywz, 2, 1, 3, 2)
00193 TRISYCL_GEN_SWIZ4(zyww, 2, 1, 3, 3)
00194 TRISYCL_GEN_SWIZ4(zzxx, 2, 2, 0, 0)
00195 TRISYCL_GEN_SWIZ4(zzxy, 2, 2, 0, 1)
00196 TRISYCL_GEN_SWIZ4(zzxz, 2, 2, 0, 2)
00197 TRISYCL_GEN_SWIZ4(zzxw, 2, 2, 0, 3)
00198 TRISYCL_GEN_SWIZ4(zzyx, 2, 2, 1, 0)
00199 TRISYCL_GEN_SWIZ4(zzyy, 2, 2, 1, 1)
00200 TRISYCL_GEN_SWIZ4(zzyz, 2, 2, 1, 2)
00201 TRISYCL_GEN_SWIZ4(zzyw, 2, 2, 1, 3)
00202 TRISYCL_GEN_SWIZ4(zzzx, 2, 2, 2, 0)
00203 TRISYCL_GEN_SWIZ4(zzzy, 2, 2, 2, 1)
00204 TRISYCL_GEN_SWIZ4(zzzz, 2, 2, 2, 2)
00205 TRISYCL_GEN_SWIZ4(zzzw, 2, 2, 2, 3)
00206 TRISYCL_GEN_SWIZ4(zzwx, 2, 2, 3, 0)
00207 TRISYCL_GEN_SWIZ4(zzwy, 2, 2, 3, 1)
00208 TRISYCL_GEN_SWIZ4(zzwz, 2, 2, 3, 2)
00209 TRISYCL_GEN_SWIZ4(zzww, 2, 2, 3, 3)
00210 TRISYCL_GEN_SWIZ4(zwxx, 2, 3, 0, 0)
00211 TRISYCL_GEN_SWIZ4(zwxy, 2, 3, 0, 1)
00212 TRISYCL_GEN_SWIZ4(zwxz, 2, 3, 0, 2)
00213 TRISYCL_GEN_SWIZ4(zwxw, 2, 3, 0, 3)
00214 TRISYCL_GEN_SWIZ4(zwyx, 2, 3, 1, 0)
00215 TRISYCL_GEN_SWIZ4(zwyy, 2, 3, 1, 1)
00216 TRISYCL_GEN_SWIZ4(zwyz, 2, 3, 1, 2)
00217 TRISYCL_GEN_SWIZ4(zwyw, 2, 3, 1, 3)
00218 TRISYCL_GEN_SWIZ4(zwzx, 2, 3, 2, 0)
00219 TRISYCL_GEN_SWIZ4(zwzy, 2, 3, 2, 1)
00220 TRISYCL_GEN_SWIZ4(zwzz, 2, 3, 2, 2)
00221 TRISYCL_GEN_SWIZ4(zwzw, 2, 3, 2, 3)
00222 TRISYCL_GEN_SWIZ4(zwwx, 2, 3, 3, 0)
00223 TRISYCL_GEN_SWIZ4(zwwy, 2, 3, 3, 1)
00224 TRISYCL_GEN_SWIZ4(zwwz, 2, 3, 3, 2)
00225 TRISYCL_GEN_SWIZ4(zwww, 2, 3, 3, 3)
00226 TRISYCL_GEN_SWIZ4(wxxx, 3, 0, 0, 0)
00227 TRISYCL_GEN_SWIZ4(wxxy, 3, 0, 0, 1)
00228 TRISYCL_GEN_SWIZ4(wxxz, 3, 0, 0, 2)
00229 TRISYCL_GEN_SWIZ4(wxxw, 3, 0, 0, 3)
00230 TRISYCL_GEN_SWIZ4(wxyx, 3, 0, 1, 0)
00231 TRISYCL_GEN_SWIZ4(wxyy, 3, 0, 1, 1)
00232 TRISYCL_GEN_SWIZ4(wxyz, 3, 0, 1, 2)
00233 TRISYCL_GEN_SWIZ4(wxyw, 3, 0, 1, 3)
00234 TRISYCL_GEN_SWIZ4(wxzx, 3, 0, 2, 0)
00235 TRISYCL_GEN_SWIZ4(wxzy, 3, 0, 2, 1)
00236 TRISYCL_GEN_SWIZ4(wxzz, 3, 0, 2, 2)
00237 TRISYCL_GEN_SWIZ4(wxzw, 3, 0, 2, 3)
00238 TRISYCL_GEN_SWIZ4(wxwx, 3, 0, 3, 0)
00239 TRISYCL_GEN_SWIZ4(wxwy, 3, 0, 3, 1)
00240 TRISYCL_GEN_SWIZ4(wxwz, 3, 0, 3, 2)
00241 TRISYCL_GEN_SWIZ4(wxww, 3, 0, 3, 3)
00242 TRISYCL_GEN_SWIZ4(wyxx, 3, 1, 0, 0)
00243 TRISYCL_GEN_SWIZ4(wyxy, 3, 1, 0, 1)
00244 TRISYCL_GEN_SWIZ4(wyxz, 3, 1, 0, 2)
00245 TRISYCL_GEN_SWIZ4(wyxw, 3, 1, 0, 3)
00246 TRISYCL_GEN_SWIZ4(wyyx, 3, 1, 1, 0)
00247 TRISYCL_GEN_SWIZ4(wyyy, 3, 1, 1, 1)
00248 TRISYCL_GEN_SWIZ4(wyyz, 3, 1, 1, 2)
00249 TRISYCL_GEN_SWIZ4(wyyw, 3, 1, 1, 3)
00250 TRISYCL_GEN_SWIZ4(wyzx, 3, 1, 2, 0)
00251 TRISYCL_GEN_SWIZ4(wyzy, 3, 1, 2, 1)
00252 TRISYCL_GEN_SWIZ4(wyzz, 3, 1, 2, 2)
00253 TRISYCL_GEN_SWIZ4(wyzw, 3, 1, 2, 3)
00254 TRISYCL_GEN_SWIZ4(wywx, 3, 1, 3, 0)
00255 TRISYCL_GEN_SWIZ4(wywy, 3, 1, 3, 1)
00256 TRISYCL_GEN_SWIZ4(wywz, 3, 1, 3, 2)
00257 TRISYCL_GEN_SWIZ4(wyww, 3, 1, 3, 3)
00258 TRISYCL_GEN_SWIZ4(wzxx, 3, 2, 0, 0)
00259 TRISYCL_GEN_SWIZ4(wzxy, 3, 2, 0, 1)
00260 TRISYCL_GEN_SWIZ4(wzxz, 3, 2, 0, 2)
00261 TRISYCL_GEN_SWIZ4(wzxw, 3, 2, 0, 3)
00262 TRISYCL_GEN_SWIZ4(wzyx, 3, 2, 1, 0)
00263 TRISYCL_GEN_SWIZ4(wzyy, 3, 2, 1, 1)
00264 TRISYCL_GEN_SWIZ4(wzyz, 3, 2, 1, 2)
00265 TRISYCL_GEN_SWIZ4(wzyw, 3, 2, 1, 3)
00266 TRISYCL_GEN_SWIZ4(wzzx, 3, 2, 2, 0)
00267 TRISYCL_GEN_SWIZ4(wzzy, 3, 2, 2, 1)
00268 TRISYCL_GEN_SWIZ4(wzzz, 3, 2, 2, 2)
00269 TRISYCL_GEN_SWIZ4(wzzw, 3, 2, 2, 3)
00270 TRISYCL_GEN_SWIZ4(wzwx, 3, 2, 3, 0)
00271 TRISYCL_GEN_SWIZ4(wzwy, 3, 2, 3, 1)
00272 TRISYCL_GEN_SWIZ4(wzwz, 3, 2, 3, 2)
00273 TRISYCL_GEN_SWIZ4(wzww, 3, 2, 3, 3)
00274 TRISYCL_GEN_SWIZ4(wwxx, 3, 3, 0, 0)
00275 TRISYCL_GEN_SWIZ4(wwxy, 3, 3, 0, 1)
00276 TRISYCL_GEN_SWIZ4(wwxz, 3, 3, 0, 2)
```

```
00277 TRISYCL_GEN_SWIZ4(wwxw, 3, 3, 0, 3)
00278 TRISYCL_GEN_SWIZ4(wwyx, 3, 3, 1, 0)
00279 TRISYCL_GEN_SWIZ4(wwyy, 3, 3, 1, 1)
00280 TRISYCL_GEN_SWIZ4(wwyz, 3, 3, 1, 2)
00281 TRISYCL_GEN_SWIZ4(wwyw, 3, 3, 1, 3)
00282 TRISYCL_GEN_SWIZ4(wwzx, 3, 3, 2, 0)
00283 TRISYCL_GEN_SWIZ4(wwzy, 3, 3, 2, 1)
00284 TRISYCL_GEN_SWIZ4(wwzz, 3, 3, 2, 2)
00285 TRISYCL_GEN_SWIZ4(wwzw, 3, 3, 2, 3)
00286 TRISYCL_GEN_SWIZ4(wwwx, 3, 3, 3, 0)
00287 TRISYCL_GEN_SWIZ4(wwwy, 3, 3, 3, 1)
00288 TRISYCL_GEN_SWIZ4(wwwz, 3, 3, 3, 2)
00289 TRISYCL_GEN_SWIZ4(wwww, 3, 3, 3, 3)
00290 #endif
```

13.201 include/triSYCL/vec/detail/swiz_rgba.hpp File Reference

This graph shows which files directly or indirectly include this file:



13.202 swiz_rgba.hpp

Go to the documentation of this file.

```
00001 // Generated using a simple C generator
00002 #ifdef SYCL_SIMPLE_SWIZZLES
00003 TRISYCL_GEN_SWIZ4(rrrr, 0, 0, 0, 0)
00004 TRISYCL_GEN_SWIZ4(rrrg, 0, 0, 0, 1)
00005 TRISYCL_GEN_SWIZ4(rrrb, 0, 0, 0, 2)
00006 TRISYCL_GEN_SWIZ4(rrra, 0, 0, 0, 3)
00007 TRISYCL_GEN_SWIZ4(rrgr, 0, 0, 1, 0)
00008 TRISYCL_GEN_SWIZ4(rrgg, 0, 0, 1, 1)
```

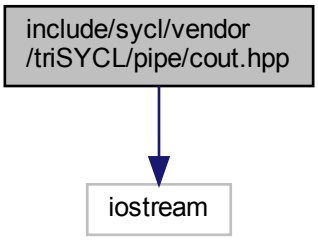
```
00009 TRISYCL_GEN_SWIZ4(rrgb, 0, 0, 1, 2)
00010 TRISYCL_GEN_SWIZ4(rrga, 0, 0, 1, 3)
00011 TRISYCL_GEN_SWIZ4(rrbr, 0, 0, 2, 0)
00012 TRISYCL_GEN_SWIZ4(rrbg, 0, 0, 2, 1)
00013 TRISYCL_GEN_SWIZ4(rrbb, 0, 0, 2, 2)
00014 TRISYCL_GEN_SWIZ4(rrba, 0, 0, 2, 3)
00015 TRISYCL_GEN_SWIZ4(rrar, 0, 0, 3, 0)
00016 TRISYCL_GEN_SWIZ4(rrag, 0, 0, 3, 1)
00017 TRISYCL_GEN_SWIZ4(rrab, 0, 0, 3, 2)
00018 TRISYCL_GEN_SWIZ4(rraa, 0, 0, 3, 3)
00019 TRISYCL_GEN_SWIZ4(rgrr, 0, 1, 0, 0)
00020 TRISYCL_GEN_SWIZ4(rgrg, 0, 1, 0, 1)
00021 TRISYCL_GEN_SWIZ4(rgrb, 0, 1, 0, 2)
00022 TRISYCL_GEN_SWIZ4(rgra, 0, 1, 0, 3)
00023 TRISYCL_GEN_SWIZ4(rggr, 0, 1, 1, 0)
00024 TRISYCL_GEN_SWIZ4(rggg, 0, 1, 1, 1)
00025 TRISYCL_GEN_SWIZ4(rggb, 0, 1, 1, 2)
00026 TRISYCL_GEN_SWIZ4(rgga, 0, 1, 1, 3)
00027 TRISYCL_GEN_SWIZ4(rgbr, 0, 1, 2, 0)
00028 TRISYCL_GEN_SWIZ4(rgbg, 0, 1, 2, 1)
00029 TRISYCL_GEN_SWIZ4(rgbb, 0, 1, 2, 2)
00030 TRISYCL_GEN_SWIZ4(rgba, 0, 1, 2, 3)
00031 TRISYCL_GEN_SWIZ4(rgar, 0, 1, 3, 0)
00032 TRISYCL_GEN_SWIZ4(rgag, 0, 1, 3, 1)
00033 TRISYCL_GEN_SWIZ4(rgab, 0, 1, 3, 2)
00034 TRISYCL_GEN_SWIZ4(rgaa, 0, 1, 3, 3)
00035 TRISYCL_GEN_SWIZ4(rbrr, 0, 2, 0, 0)
00036 TRISYCL_GEN_SWIZ4(rbrg, 0, 2, 0, 1)
00037 TRISYCL_GEN_SWIZ4(rbrb, 0, 2, 0, 2)
00038 TRISYCL_GEN_SWIZ4(rbra, 0, 2, 0, 3)
00039 TRISYCL_GEN_SWIZ4(rbgr, 0, 2, 1, 0)
00040 TRISYCL_GEN_SWIZ4(rbgg, 0, 2, 1, 1)
00041 TRISYCL_GEN_SWIZ4(rbgb, 0, 2, 1, 2)
00042 TRISYCL_GEN_SWIZ4(rbga, 0, 2, 1, 3)
00043 TRISYCL_GEN_SWIZ4(rbbr, 0, 2, 2, 0)
00044 TRISYCL_GEN_SWIZ4(rbbg, 0, 2, 2, 1)
00045 TRISYCL_GEN_SWIZ4(rbbb, 0, 2, 2, 2)
00046 TRISYCL_GEN_SWIZ4(rbba, 0, 2, 2, 3)
00047 TRISYCL_GEN_SWIZ4(rbar, 0, 2, 3, 0)
00048 TRISYCL_GEN_SWIZ4(rbag, 0, 2, 3, 1)
00049 TRISYCL_GEN_SWIZ4(rbab, 0, 2, 3, 2)
00050 TRISYCL_GEN_SWIZ4(rbaa, 0, 2, 3, 3)
00051 TRISYCL_GEN_SWIZ4(rarr, 0, 3, 0, 0)
00052 TRISYCL_GEN_SWIZ4(rarg, 0, 3, 0, 1)
00053 TRISYCL_GEN_SWIZ4(rarb, 0, 3, 0, 2)
00054 TRISYCL_GEN_SWIZ4(rara, 0, 3, 0, 3)
00055 TRISYCL_GEN_SWIZ4(ragr, 0, 3, 1, 0)
00056 TRISYCL_GEN_SWIZ4(ragg, 0, 3, 1, 1)
00057 TRISYCL_GEN_SWIZ4(ragb, 0, 3, 1, 2)
00058 TRISYCL_GEN_SWIZ4(raga, 0, 3, 1, 3)
00059 TRISYCL_GEN_SWIZ4(rabr, 0, 3, 2, 0)
00060 TRISYCL_GEN_SWIZ4(rabg, 0, 3, 2, 1)
00061 TRISYCL_GEN_SWIZ4(rabb, 0, 3, 2, 2)
00062 TRISYCL_GEN_SWIZ4(raba, 0, 3, 2, 3)
00063 TRISYCL_GEN_SWIZ4(raar, 0, 3, 3, 0)
00064 TRISYCL_GEN_SWIZ4(raag, 0, 3, 3, 1)
00065 TRISYCL_GEN_SWIZ4(raab, 0, 3, 3, 2)
00066 TRISYCL_GEN_SWIZ4(raaa, 0, 3, 3, 3)
00067 TRISYCL_GEN_SWIZ4(grrr, 1, 0, 0, 0)
00068 TRISYCL_GEN_SWIZ4(grrg, 1, 0, 0, 1)
00069 TRISYCL_GEN_SWIZ4(grrb, 1, 0, 0, 2)
00070 TRISYCL_GEN_SWIZ4(grra, 1, 0, 0, 3)
00071 TRISYCL_GEN_SWIZ4(grgr, 1, 0, 1, 0)
00072 TRISYCL_GEN_SWIZ4(grgg, 1, 0, 1, 1)
00073 TRISYCL_GEN_SWIZ4(grgb, 1, 0, 1, 2)
00074 TRISYCL_GEN_SWIZ4(grga, 1, 0, 1, 3)
00075 TRISYCL_GEN_SWIZ4(grbr, 1, 0, 2, 0)
00076 TRISYCL_GEN_SWIZ4(grbg, 1, 0, 2, 1)
00077 TRISYCL_GEN_SWIZ4(grbb, 1, 0, 2, 2)
00078 TRISYCL_GEN_SWIZ4(grba, 1, 0, 2, 3)
00079 TRISYCL_GEN_SWIZ4(grar, 1, 0, 3, 0)
00080 TRISYCL_GEN_SWIZ4(grag, 1, 0, 3, 1)
00081 TRISYCL_GEN_SWIZ4(grab, 1, 0, 3, 2)
00082 TRISYCL_GEN_SWIZ4(graa, 1, 0, 3, 3)
00083 TRISYCL_GEN_SWIZ4(ggrr, 1, 1, 0, 0)
00084 TRISYCL_GEN_SWIZ4(ggrg, 1, 1, 0, 1)
00085 TRISYCL_GEN_SWIZ4(ggrb, 1, 1, 0, 2)
00086 TRISYCL_GEN_SWIZ4(ggra, 1, 1, 0, 3)
00087 TRISYCL_GEN_SWIZ4(gggr, 1, 1, 1, 0)
00088 TRISYCL_GEN_SWIZ4(gggg, 1, 1, 1, 1)
00089 TRISYCL_GEN_SWIZ4(gggb, 1, 1, 1, 2)
00090 TRISYCL_GEN_SWIZ4(ggga, 1, 1, 1, 3)
00091 TRISYCL_GEN_SWIZ4(ggbr, 1, 1, 2, 0)
00092 TRISYCL_GEN_SWIZ4(ggbg, 1, 1, 2, 1)
00093 TRISYCL_GEN_SWIZ4(ggbb, 1, 1, 2, 2)
00094 TRISYCL_GEN_SWIZ4(ggba, 1, 1, 2, 3)
00095 TRISYCL_GEN_SWIZ4(ggar, 1, 1, 3, 0)
```

```
00096 TRISYCL_GEN_SWIZ4(ggag, 1, 1, 3, 1)
00097 TRISYCL_GEN_SWIZ4(ggab, 1, 1, 3, 2)
00098 TRISYCL_GEN_SWIZ4(ggaa, 1, 1, 3, 3)
00099 TRISYCL_GEN_SWIZ4(gbrr, 1, 2, 0, 0)
00100 TRISYCL_GEN_SWIZ4(gbrg, 1, 2, 0, 1)
00101 TRISYCL_GEN_SWIZ4(gbrb, 1, 2, 0, 2)
00102 TRISYCL_GEN_SWIZ4(gbra, 1, 2, 0, 3)
00103 TRISYCL_GEN_SWIZ4(gbgr, 1, 2, 1, 0)
00104 TRISYCL_GEN_SWIZ4(gbgg, 1, 2, 1, 1)
00105 TRISYCL_GEN_SWIZ4(gbgb, 1, 2, 1, 2)
00106 TRISYCL_GEN_SWIZ4(gbga, 1, 2, 1, 3)
00107 TRISYCL_GEN_SWIZ4(gbbr, 1, 2, 2, 0)
00108 TRISYCL_GEN_SWIZ4(gbbg, 1, 2, 2, 1)
00109 TRISYCL_GEN_SWIZ4(gbbb, 1, 2, 2, 2)
00110 TRISYCL_GEN_SWIZ4(gbba, 1, 2, 2, 3)
00111 TRISYCL_GEN_SWIZ4(gbar, 1, 2, 3, 0)
00112 TRISYCL_GEN_SWIZ4(gbag, 1, 2, 3, 1)
00113 TRISYCL_GEN_SWIZ4(gbab, 1, 2, 3, 2)
00114 TRISYCL_GEN_SWIZ4(gbaa, 1, 2, 3, 3)
00115 TRISYCL_GEN_SWIZ4(garr, 1, 3, 0, 0)
00116 TRISYCL_GEN_SWIZ4(garg, 1, 3, 0, 1)
00117 TRISYCL_GEN_SWIZ4(garb, 1, 3, 0, 2)
00118 TRISYCL_GEN_SWIZ4(gara, 1, 3, 0, 3)
00119 TRISYCL_GEN_SWIZ4(gagr, 1, 3, 1, 0)
00120 TRISYCL_GEN_SWIZ4(gagg, 1, 3, 1, 1)
00121 TRISYCL_GEN_SWIZ4(gagb, 1, 3, 1, 2)
00122 TRISYCL_GEN_SWIZ4(gaga, 1, 3, 1, 3)
00123 TRISYCL_GEN_SWIZ4(gabr, 1, 3, 2, 0)
00124 TRISYCL_GEN_SWIZ4(gabg, 1, 3, 2, 1)
00125 TRISYCL_GEN_SWIZ4(gabb, 1, 3, 2, 2)
00126 TRISYCL_GEN_SWIZ4(gaba, 1, 3, 2, 3)
00127 TRISYCL_GEN_SWIZ4(gaar, 1, 3, 3, 0)
00128 TRISYCL_GEN_SWIZ4(gaag, 1, 3, 3, 1)
00129 TRISYCL_GEN_SWIZ4(gaab, 1, 3, 3, 2)
00130 TRISYCL_GEN_SWIZ4(gaaa, 1, 3, 3, 3)
00131 TRISYCL_GEN_SWIZ4(brrr, 2, 0, 0, 0)
00132 TRISYCL_GEN_SWIZ4(brrg, 2, 0, 0, 1)
00133 TRISYCL_GEN_SWIZ4(brrb, 2, 0, 0, 2)
00134 TRISYCL_GEN_SWIZ4(brra, 2, 0, 0, 3)
00135 TRISYCL_GEN_SWIZ4(brgr, 2, 0, 1, 0)
00136 TRISYCL_GEN_SWIZ4(brgg, 2, 0, 1, 1)
00137 TRISYCL_GEN_SWIZ4(brgb, 2, 0, 1, 2)
00138 TRISYCL_GEN_SWIZ4(brga, 2, 0, 1, 3)
00139 TRISYCL_GEN_SWIZ4(brbr, 2, 0, 2, 0)
00140 TRISYCL_GEN_SWIZ4(brbg, 2, 0, 2, 1)
00141 TRISYCL_GEN_SWIZ4(brbb, 2, 0, 2, 2)
00142 TRISYCL_GEN_SWIZ4(brba, 2, 0, 2, 3)
00143 TRISYCL_GEN_SWIZ4(brar, 2, 0, 3, 0)
00144 TRISYCL_GEN_SWIZ4(brag, 2, 0, 3, 1)
00145 TRISYCL_GEN_SWIZ4(brab, 2, 0, 3, 2)
00146 TRISYCL_GEN_SWIZ4(braa, 2, 0, 3, 3)
00147 TRISYCL_GEN_SWIZ4(bgrr, 2, 1, 0, 0)
00148 TRISYCL_GEN_SWIZ4(bgrg, 2, 1, 0, 1)
00149 TRISYCL_GEN_SWIZ4(bgrb, 2, 1, 0, 2)
00150 TRISYCL_GEN_SWIZ4(bgra, 2, 1, 0, 3)
00151 TRISYCL_GEN_SWIZ4(bggr, 2, 1, 1, 0)
00152 TRISYCL_GEN_SWIZ4(bggg, 2, 1, 1, 1)
00153 TRISYCL_GEN_SWIZ4(bggb, 2, 1, 1, 2)
00154 TRISYCL_GEN_SWIZ4(bgga, 2, 1, 1, 3)
00155 TRISYCL_GEN_SWIZ4(bgbr, 2, 1, 2, 0)
00156 TRISYCL_GEN_SWIZ4(bgbg, 2, 1, 2, 1)
00157 TRISYCL_GEN_SWIZ4(bgbb, 2, 1, 2, 2)
00158 TRISYCL_GEN_SWIZ4(bgba, 2, 1, 2, 3)
00159 TRISYCL_GEN_SWIZ4(bgar, 2, 1, 3, 0)
00160 TRISYCL_GEN_SWIZ4(bgag, 2, 1, 3, 1)
00161 TRISYCL_GEN_SWIZ4(bgab, 2, 1, 3, 2)
00162 TRISYCL_GEN_SWIZ4(bgaa, 2, 1, 3, 3)
00163 TRISYCL_GEN_SWIZ4(bbrr, 2, 2, 0, 0)
00164 TRISYCL_GEN_SWIZ4(bbrg, 2, 2, 0, 1)
00165 TRISYCL_GEN_SWIZ4(bbrb, 2, 2, 0, 2)
00166 TRISYCL_GEN_SWIZ4(bbra, 2, 2, 0, 3)
00167 TRISYCL_GEN_SWIZ4(bbgr, 2, 2, 1, 0)
00168 TRISYCL_GEN_SWIZ4(bbgg, 2, 2, 1, 1)
00169 TRISYCL_GEN_SWIZ4(bbgb, 2, 2, 1, 2)
00170 TRISYCL_GEN_SWIZ4(bbga, 2, 2, 1, 3)
00171 TRISYCL_GEN_SWIZ4(bbbr, 2, 2, 2, 0)
00172 TRISYCL_GEN_SWIZ4(bbbg, 2, 2, 2, 1)
00173 TRISYCL_GEN_SWIZ4(bbbb, 2, 2, 2, 2)
00174 TRISYCL_GEN_SWIZ4(bbba, 2, 2, 2, 3)
00175 TRISYCL_GEN_SWIZ4(bbar, 2, 2, 3, 0)
00176 TRISYCL_GEN_SWIZ4(bbag, 2, 2, 3, 1)
00177 TRISYCL_GEN_SWIZ4(bbab, 2, 2, 3, 2)
00178 TRISYCL_GEN_SWIZ4(bbaa, 2, 2, 3, 3)
00179 TRISYCL_GEN_SWIZ4(barr, 2, 3, 0, 0)
00180 TRISYCL_GEN_SWIZ4(barg, 2, 3, 0, 1)
00181 TRISYCL_GEN_SWIZ4(barb, 2, 3, 0, 2)
00182 TRISYCL_GEN_SWIZ4(bara, 2, 3, 0, 3)
```

```
00183 TRISYCL_GEN_SWIZ4(bagr, 2, 3, 1, 0)
00184 TRISYCL_GEN_SWIZ4(bagg, 2, 3, 1, 1)
00185 TRISYCL_GEN_SWIZ4(bagb, 2, 3, 1, 2)
00186 TRISYCL_GEN_SWIZ4(baga, 2, 3, 1, 3)
00187 TRISYCL_GEN_SWIZ4(babr, 2, 3, 2, 0)
00188 TRISYCL_GEN_SWIZ4(babg, 2, 3, 2, 1)
00189 TRISYCL_GEN_SWIZ4(babb, 2, 3, 2, 2)
00190 TRISYCL_GEN_SWIZ4(baba, 2, 3, 2, 3)
00191 TRISYCL_GEN_SWIZ4(baar, 2, 3, 3, 0)
00192 TRISYCL_GEN_SWIZ4(baag, 2, 3, 3, 1)
00193 TRISYCL_GEN_SWIZ4(baab, 2, 3, 3, 2)
00194 TRISYCL_GEN_SWIZ4(baaa, 2, 3, 3, 3)
00195 TRISYCL_GEN_SWIZ4(arrr, 3, 0, 0, 0)
00196 TRISYCL_GEN_SWIZ4(arrg, 3, 0, 0, 1)
00197 TRISYCL_GEN_SWIZ4(arrb, 3, 0, 0, 2)
00198 TRISYCL_GEN_SWIZ4(arra, 3, 0, 0, 3)
00199 TRISYCL_GEN_SWIZ4(argr, 3, 0, 1, 0)
00200 TRISYCL_GEN_SWIZ4(argg, 3, 0, 1, 1)
00201 TRISYCL_GEN_SWIZ4(argb, 3, 0, 1, 2)
00202 TRISYCL_GEN_SWIZ4(arga, 3, 0, 1, 3)
00203 TRISYCL_GEN_SWIZ4(arbr, 3, 0, 2, 0)
00204 TRISYCL_GEN_SWIZ4(arbg, 3, 0, 2, 1)
00205 TRISYCL_GEN_SWIZ4(arbb, 3, 0, 2, 2)
00206 TRISYCL_GEN_SWIZ4(arba, 3, 0, 2, 3)
00207 TRISYCL_GEN_SWIZ4(arar, 3, 0, 3, 0)
00208 TRISYCL_GEN_SWIZ4(arag, 3, 0, 3, 1)
00209 TRISYCL_GEN_SWIZ4(arab, 3, 0, 3, 2)
00210 TRISYCL_GEN_SWIZ4(araa, 3, 0, 3, 3)
00211 TRISYCL_GEN_SWIZ4(agrr, 3, 1, 0, 0)
00212 TRISYCL_GEN_SWIZ4(agrg, 3, 1, 0, 1)
00213 TRISYCL_GEN_SWIZ4(agrb, 3, 1, 0, 2)
00214 TRISYCL_GEN_SWIZ4(agra, 3, 1, 0, 3)
00215 TRISYCL_GEN_SWIZ4(aggr, 3, 1, 1, 0)
00216 TRISYCL_GEN_SWIZ4(aggg, 3, 1, 1, 1)
00217 TRISYCL_GEN_SWIZ4(aggb, 3, 1, 1, 2)
00218 TRISYCL_GEN_SWIZ4(agga, 3, 1, 1, 3)
00219 TRISYCL_GEN_SWIZ4(agbr, 3, 1, 2, 0)
00220 TRISYCL_GEN_SWIZ4(agbg, 3, 1, 2, 1)
00221 TRISYCL_GEN_SWIZ4(agbb, 3, 1, 2, 2)
00222 TRISYCL_GEN_SWIZ4(agba, 3, 1, 2, 3)
00223 TRISYCL_GEN_SWIZ4(agar, 3, 1, 3, 0)
00224 TRISYCL_GEN_SWIZ4(agag, 3, 1, 3, 1)
00225 TRISYCL_GEN_SWIZ4(agab, 3, 1, 3, 2)
00226 TRISYCL_GEN_SWIZ4(agaa, 3, 1, 3, 3)
00227 TRISYCL_GEN_SWIZ4(abrr, 3, 2, 0, 0)
00228 TRISYCL_GEN_SWIZ4(abrg, 3, 2, 0, 1)
00229 TRISYCL_GEN_SWIZ4(abrb, 3, 2, 0, 2)
00230 TRISYCL_GEN_SWIZ4(abra, 3, 2, 0, 3)
00231 TRISYCL_GEN_SWIZ4(abgr, 3, 2, 1, 0)
00232 TRISYCL_GEN_SWIZ4(abgg, 3, 2, 1, 1)
00233 TRISYCL_GEN_SWIZ4(abgb, 3, 2, 1, 2)
00234 TRISYCL_GEN_SWIZ4(abga, 3, 2, 1, 3)
00235 TRISYCL_GEN_SWIZ4(abbr, 3, 2, 2, 0)
00236 TRISYCL_GEN_SWIZ4(abbg, 3, 2, 2, 1)
00237 TRISYCL_GEN_SWIZ4(abbb, 3, 2, 2, 2)
00238 TRISYCL_GEN_SWIZ4(abba, 3, 2, 2, 3)
00239 TRISYCL_GEN_SWIZ4(abar, 3, 2, 3, 0)
00240 TRISYCL_GEN_SWIZ4(abag, 3, 2, 3, 1)
00241 TRISYCL_GEN_SWIZ4(abab, 3, 2, 3, 2)
00242 TRISYCL_GEN_SWIZ4(abaa, 3, 2, 3, 3)
00243 TRISYCL_GEN_SWIZ4(aarr, 3, 3, 0, 0)
00244 TRISYCL_GEN_SWIZ4(aarg, 3, 3, 0, 1)
00245 TRISYCL_GEN_SWIZ4(aarb, 3, 3, 0, 2)
00246 TRISYCL_GEN_SWIZ4(aara, 3, 3, 0, 3)
00247 TRISYCL_GEN_SWIZ4(aagr, 3, 3, 1, 0)
00248 TRISYCL_GEN_SWIZ4(aagg, 3, 3, 1, 1)
00249 TRISYCL_GEN_SWIZ4(aagb, 3, 3, 1, 2)
00250 TRISYCL_GEN_SWIZ4(aaga, 3, 3, 1, 3)
00251 TRISYCL_GEN_SWIZ4(aabr, 3, 3, 2, 0)
00252 TRISYCL_GEN_SWIZ4(aabg, 3, 3, 2, 1)
00253 TRISYCL_GEN_SWIZ4(aabb, 3, 3, 2, 2)
00254 TRISYCL_GEN_SWIZ4(aaba, 3, 3, 2, 3)
00255 TRISYCL_GEN_SWIZ4(aaar, 3, 3, 3, 0)
00256 TRISYCL_GEN_SWIZ4(aaag, 3, 3, 3, 1)
00257 TRISYCL_GEN_SWIZ4(aaab, 3, 3, 3, 2)
00258 TRISYCL_GEN_SWIZ4(aaaa, 3, 3, 3, 3)
00259 #endif
```

13.203 include/sycl/vendor/triSYCL/pipe/cout.hpp File Reference

`#include <iostream>`
Include dependency graph for cout.hpp:



Classes

- class trisycl::vendor::trisycl::pipe::cout

  triSYCL extension for SYCL pipe object sending data to std::cout More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::pipe

13.204 cout.hpp

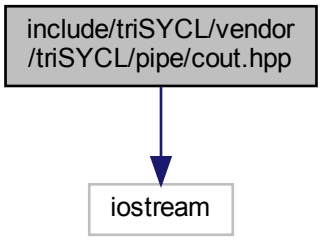
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_PIPE_COUT_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_PIPE_COUT_HPP
00003
00004 /** \file The triSYCL data-flow pipe extension to send data to std::cout
00005
00006 This file is distributed under the University of Illinois Open Source
00007 License.  See LICENSE.TXT for details.
00008 */
00009
00010 #include <iostream>
00011
00012 namespace trisycl::vendor::trisycl::pipe {
00013
00014 /** \addtogroup data Data access and storage in SYCL
00015 @{
00016 */
00017
00018 /** triSYCL extension for SYCL pipe object sending data to std::cout
00019
00020 Experiment with a new pipe extension to send data to std::cout
00021 with a simplified pipe extension like with:
00022
```

```
00023 \code
00024 sycl::vendor::trisycl::pipe::cout::write("salut !\n");
00025 sycl::vendor::trisycl::pipe::cout::stream() « "hello " « 42 « std::endl;
00026 \endcode
00027 */
00028 class cout {
00029
00030 public:
00031
00032 /// Provide the usual stream interface to std::cout
00033   static auto& stream() {
00034     return std::cout;
00035   }
00036
00037 /// Send some value to std::cout
00038   template <typename T>
00039   static void write(const T &value) {
00040     stream() « value;
00041   }
00042
00043 };
00044
00045
00046 /// @} End the execution Doxygen group
00047
00048 }
00049
00050 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_PIPE_COUT_HPP
```

13.205 include/triSYCL/vendor/triSYCL/pipe/cout.hpp File Reference

`#include <iostream>`
Include dependency graph for cout.hpp:



Classes

- class trisycl::vendor::trisycl::pipe::cout

  triSYCL extension for SYCL pipe object sending data to std::cout More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::pipe

13.206 cout.hpp

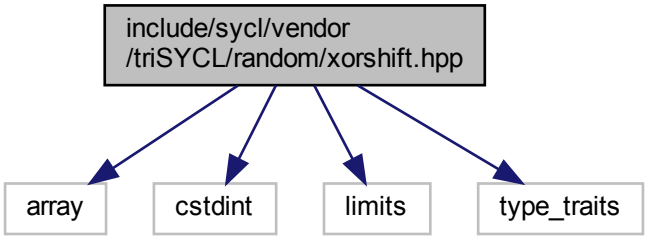
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_PIPE_COUT_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_PIPE_COUT_HPP
00003
00004 /** \file The triSYCL data-flow pipe extension to send data to std::cout
00005
00006 This file is distributed under the University of Illinois Open Source
00007 License.  See LICENSE.TXT for details.
00008 */
00009
00010 #include <iostream>
00011
00012 namespace trisycl::vendor::trisycl::pipe {
00013
00014 /** \addtogroup data Data access and storage in SYCL
00015 @{
00016 */
00017
00018 /** triSYCL extension for SYCL pipe object sending data to std::cout
00019
00020 Experiment with a new pipe extension to send data to std::cout
00021 with a simplified pipe extension like with:
00022
00023 \code
00024 sycl::vendor::trisycl::pipe::cout::write("salut !\n");
00025 sycl::vendor::trisycl::pipe::cout::stream() « "hello " « 42 « std::endl;
00026 \endcode
00027 */
00028 class cout {
00029
00030 public:
00031
00032 /// Provide the usual stream interface to std::cout
00033   static auto& stream() {
00034     return std::cout;
00035   }
00036
00037 /// Send some value to std::cout
00038   template <typename T>
00039   static void write(const T &value) {
00040     stream() « value;
00041   }
00042
00043 };
00044
00045
00046 /// @} End the execution Doxygen group
00047
00048 }
00049
00050 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_PIPE_COUT_HPP
```

13.207 include/sycl/vendor/triSYCL/random/xorshift.hpp File Reference

```
#include <array>
#include <cstdint>
#include <limits>
#include <type_traits>
```

Include dependency graph for xorshift.hpp:



Classes

- struct trisycl::vendor::trisycl::random::xorshift< Nbits >

  A possible implementation of the xorshift RNGs from.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::random

  Random generators.

13.208 xorshift.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_RANDOM_XORSHIFT_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_RANDOM_XORSHIFT_HPP
00003
00004 /** \file A simple and fast pseudo-random generator based on binary shifts
00005 and exclusive or
00006
00007 Warning:  do not even think about using it in any secure application!
00008
00009 https://en.wikipedia.org/wiki/Xorshift
00010
00011 This file is distributed under the University of Illinois Open Source
00012 License.  See LICENSE.TXT for details.
00013 */
00014
00015 #include <array>
00016 #include <cstdint>
00017 #include <limits>
00018 #include <type_traits>
00019
00020 /// Random generators
00021 namespace trisycl::vendor::trisycl::random {
00022
00023 /** A possible implementation of the xorshift RNGs from
00024
00025 Marsaglia, George (July 2003).  "Xorshift RNGs".  Journal of
00026 Statistical Software.  8 (14).  doi:10.18637/jss.v008.i14
```

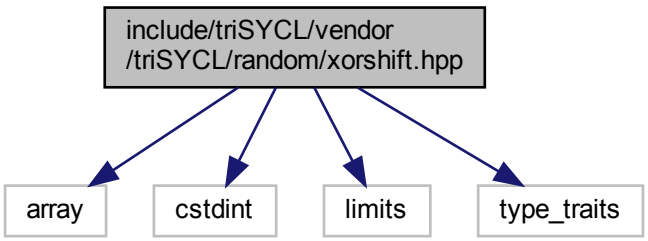
```
00027
00028 Panneton, François; L'Écuyer, Pierre (October 2005).  "On the
00029 xorshift random number generators" (PDF). ACM Transactions on
00030 Modeling and Computer Simulation.  15
00031 (4):346-361.  doi:10.1145/1113316.1113319
00032 */
00033 template <int Nbits = 32>
00034 struct xorshift {
00035 /// Bit size of the random generator data type
00036   static auto constexpr bit_size = Nbits;
00037
00038 /// The default initial internal state of the pseudo random generator
00039   static auto constexpr initial_state = [] {
00040     if constexpr (bit_size == 32) {
00041       /* Default value for xor32 from Marsaglia,
00042 p.  4 Section "3 Application to Xorshift RNGs" */
00043       return std::uint32_t { 2463534242 };
00044     } else if constexpr (bit_size == 64) {
00045       /* Default value for xor32 from Marsaglia,
00046 p.  4 Section "3 Application to Xorshift RNGs" */
00047       return std::uint64_t { 88172645463325252 };
00048     } else if constexpr (bit_size == 128) {
00049 /// Default value for xor128 from Marsaglia, p.  5 Section "4 Summary"
00050       return std::array<std::uint32_t, 4>
00051         { 123456789, 362436069, 521288629, 88675123 };
00052     } else
00053       return nullptr;
00054   }();
00055
00056 /// The type of the internal state of the generator
00057   using value_type = std::remove_const_t<decltype(initial_state)>;
00058
00059 /// The type of the result
00060   using result_type = value_type;
00061
00062   static_assert(!std::is_same_v<value_type, std::nullptr_t>,
00063                 "Bit size not implemented");
00064
00065 /// The internal state of the pseudo random generator
00066   value_type state = initial_state;
00067
00068
00069 /// The minimum returned value
00070   static auto constexpr min() {
00071     // It cannot return 0
00072     return std::numeric_limits<result_type>::min() + 1;
00073   };
00074
00075
00076 /// The maximum returned value
00077   static auto constexpr max() {
00078     return std::numeric_limits<result_type>::max();
00079   };
00080
00081
00082 /** Initialize the internal state from a user-given value
00083
00084 Do not use the value 0 or the output is always 0
00085 */
00086   xorshift(const value_type& s)
00087     :  state { s }
00088   {}
00089
00090
00091   xorshift() = default;
00092
00093
00094 /// Compute a new pseudo random integer
00095   const result_type& operator()() {
00096     if constexpr (bit_size == 32) {
00097       /* Pick the one of type "I" with best bit equidistribution from
00098 Panneton & L'Écuyer, Section "5.1 Equidistribution
00099 properties"
00100
00101 X4 = (I + Ra )(I + Lb )(I + Rc )
00102 */
00103       state ^= state » 7;
00104       state ^= state « 1;
00105       state ^= state » 9;
00106     }
00107     else if constexpr (bit_size == 64) {
00108       /* The xor64 from Marsaglia, p.  4 Section "3 Application to
00109 Xorshift RNGs" */
00110       state ^= state « 13;
00111       state ^= state » 7;
00112       state ^= state « 17;
00113     }
```

```
00114     else if constexpr (bit_size == 128) {
00115       // The xor128 from Marsaglia, p.  5 Section "4 Summary"
00116       auto t = state[0]^(state[0]«11);
00117       state[0] = state[1];
00118       state[1] = state[2];
00119       state[2] = state[3];
00120       state[3] = state[3]^(state[3]»19)^(t^(t»8));
00121     }
00122     else
00123       // Just to error in that case
00124       return nullptr;
00125
00126     return state;
00127   }
00128
00129 };
00130
00131 }
00132
00133 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_RANDOM_XORSHIFT_HPP
```

13.209 include/triSYCL/vendor/triSYCL/random/xorshift.hpp File Reference

```
#include <array>
#include <cstdint>
#include <limits>
#include <type_traits>
```

Include dependency graph for xorshift.hpp:



Classes

- struct trisycl::vendor::trisycl::random::xorshift< Nbits >

  A possible implementation of the xorshift RNGs from.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::random

  Random generators.

13.210 xorshift.hpp

Go to the documentation of this file.

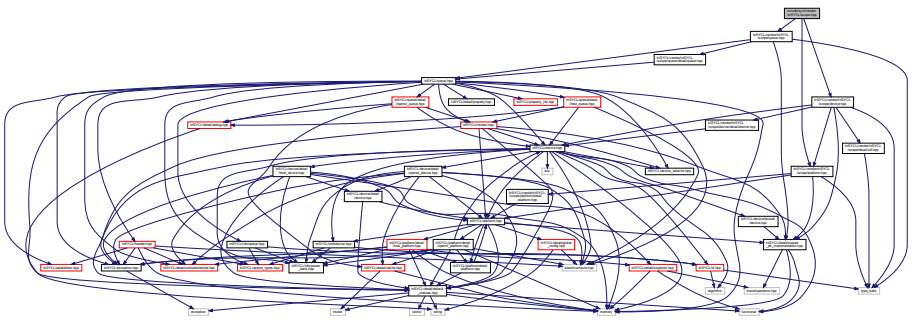
```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_RANDOM_XORSHIFT_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_RANDOM_XORSHIFT_HPP
00003
00004 /** \file A simple and fast pseudo-random generator based on binary shifts
00005 and exclusive or
00006
00007 Warning:  do not even think about using it in any secure application!
00008
00009 https://en.wikipedia.org/wiki/Xorshift
00010
00011 This file is distributed under the University of Illinois Open Source
00012 License.  See LICENSE.TXT for details.
00013 */
00014
00015 #include <array>
00016 #include <cstdint>
00017 #include <limits>
00018 #include <type_traits>
00019
00020 /// Random generators
00021 namespace trisycl::vendor::trisycl::random {
00022
00023 /** A possible implementation of the xorshift RNGs from
00024
00025 Marsaglia, George (July 2003).  "Xorshift RNGs".  Journal of
00026 Statistical Software.  8 (14).  doi:10.18637/jss.v008.i14
00027
00028 Panneton, François; L'Écuyer, Pierre (October 2005).  "On the
00029 xorshift random number generators" (PDF). ACM Transactions on
00030 Modeling and Computer Simulation.  15
00031 (4):346-361.  doi:10.1145/1113316.1113319
00032 */
00033 template <int Nbits = 32>
00034 struct xorshift {
00035 /// Bit size of the random generator data type
00036   static auto constexpr bit_size = Nbits;
00037
00038 /// The default initial internal state of the pseudo random generator
00039   static auto constexpr initial_state = [] {
00040     if constexpr (bit_size == 32) {
00041       /* Default value for xor32 from Marsaglia,
00042 p.  4 Section "3 Application to Xorshift RNGs" */
00043       return std::uint32_t { 2463534242 };
00044     } else if constexpr (bit_size == 64) {
00045       /* Default value for xor32 from Marsaglia,
00046 p.  4 Section "3 Application to Xorshift RNGs" */
00047       return std::uint64_t { 88172645463325252 };
00048     } else if constexpr (bit_size == 128) {
00049 /// Default value for xor128 from Marsaglia, p.  5 Section "4 Summary"
00050       return std::array<std::uint32_t, 4>
00051         { 123456789, 362436069, 521288629, 88675123 };
00052     } else
00053       return nullptr;
00054   }();
00055
00056 /// The type of the internal state of the generator
00057   using value_type = std::remove_const_t<decltype(initial_state)>;
00058
00059 /// The type of the result
00060   using result_type = value_type;
00061
00062   static_assert(!std::is_same_v<value_type, std::nullptr_t>,
00063                 "Bit size not implemented");
00064
00065 /// The internal state of the pseudo random generator
00066   value_type state = initial_state;
00067
00068
00069 /// The minimum returned value
00070   static auto constexpr min() {
00071     // It cannot return 0
00072     return std::numeric_limits<result_type>::min() + 1;
00073   };
00074
00075
00076 /// The maximum returned value
00077   static auto constexpr max() {
00078     return std::numeric_limits<result_type>::max();
00079   };
00080
00081
00082 /** Initialize the internal state from a user-given value
```

```
00083
00084 Do not use the value 0 or the output is always 0
00085 */
00086   xorshift(const value_type& s)
00087     :  state { s }
00088   {}
00089
00090
00091   xorshift() = default;
00092
00093
00094 /// Compute a new pseudo random integer
00095   const result_type& operator()() {
00096     if constexpr (bit_size == 32) {
00097       /* Pick the one of type "I" with best bit equidistribution from
00098 Panneton & L'Écuyer, Section "5.1 Equidistribution
00099 properties"
00100
00101 X4 = (I + Ra )(I + Lb )(I + Rc )
00102 */
00103       state ^= state » 7;
00104       state ^= state « 1;
00105       state ^= state » 9;
00106     }
00107     else if constexpr (bit_size == 64) {
00108       /* The xor64 from Marsaglia, p.  4 Section "3 Application to
00109 Xorshift RNGs" */
00110       state ^= state « 13;
00111       state ^= state » 7;
00112       state ^= state « 17;
00113     }
00114     else if constexpr (bit_size == 128) {
00115       // The xor128 from Marsaglia, p.  5 Section "4 Summary"
00116       auto t = state[0]^(state[0]«11);
00117       state[0] = state[1];
00118       state[1] = state[2];
00119       state[2] = state[3];
00120       state[3] = state[3]^(state[3]»19)^(t^(t»8));
00121     }
00122     else
00123       // Just to error in that case
00124       return nullptr;
00125
00126     return state;
00127   }
00128
00129 };
00130
00131 }
00132
00133 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_RANDOM_XORSHIFT_HPP
```

13.211 include/sycl/vendor/triSYCL/scope.hpp File Reference

```
#include "triSYCL/vendor/triSYCL/scope/device.hpp"
#include "triSYCL/vendor/triSYCL/scope/platform.hpp"
#include "triSYCL/vendor/triSYCL/scope/queue.hpp"
```

Include dependency graph for scope.hpp:



Macros

- #define SYCL_VENDOR_TRISYCL_CONCEPTUAL_API 1

  This is an extension providing a conceptual API for devices, platforms...
- #define SYCL_VENDOR_TRISYCL_SCOPE 1

  This is an extension providing scope to devices & platforms.

13.211.1 Macro Definition Documentation

13.211.1.1 SYCL_VENDOR_TRISYCL_CONCEPTUAL_API

```
#define SYCL_VENDOR_TRISYCL_CONCEPTUAL_API 1
```

This is an extension providing a conceptual API for devices, platforms...

Definition at line 17 of file scope.hpp.

13.211.1.2 SYCL_VENDOR_TRISYCL_SCOPE

```
#define SYCL_VENDOR_TRISYCL_SCOPE 1
```

This is an extension providing scope to devices & platforms.

Definition at line 20 of file scope.hpp.

13.212 scope.hpp

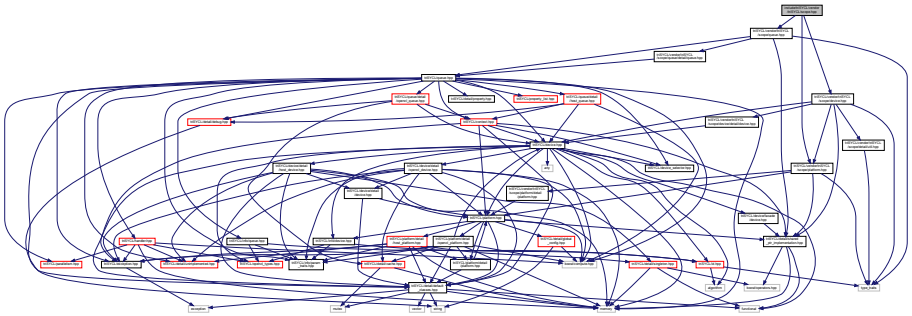
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_HPP
00003
00004 /** \file An extension adding conceptual scope storage to SYCL objects
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include "triSYCL/vendor/triSYCL/scope/device.hpp"
00013 #include "triSYCL/vendor/triSYCL/scope/platform.hpp"
00014 #include "triSYCL/vendor/triSYCL/scope/queue.hpp"
00015
00016 /// This is an extension providing a conceptual API for devices, platforms...
00017 #define SYCL_VENDOR_TRISYCL_CONCEPTUAL_API 1
00018
00019 /// This is an extension providing scope to devices & platforms
00020 #define SYCL_VENDOR_TRISYCL_SCOPE 1
00021
00022 /*
00023 # Some Emacs stuff:
00024 ### Local Variables:
00025 ### ispell-local-dictionary:  "american"
00026 ### eval:  (flyspell-prog-mode)
00027 ### End:
00028 */
00029
00030 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_HPP
```

13.213 include/triSYCL/vendor/triSYCL/scope.hpp File Reference

```
#include "triSYCL/vendor/triSYCL/scope/device.hpp"
#include "triSYCL/vendor/triSYCL/scope/platform.hpp"
#include "triSYCL/vendor/triSYCL/scope/queue.hpp"
```

Include dependency graph for scope.hpp:



Macros

- #define SYCL_VENDOR_TRISYCL_CONCEPTUAL_API 1

  This is an extension providing a conceptual API for devices, platforms...
- #define SYCL_VENDOR_TRISYCL_SCOPE 1

  This is an extension providing scope to devices & platforms.

13.213.1 Macro Definition Documentation

13.213.1.1 SYCL_VENDOR_TRISYCL_CONCEPTUAL_API

```
#define SYCL_VENDOR_TRISYCL_CONCEPTUAL_API 1
```

This is an extension providing a conceptual API for devices, platforms...

Definition at line 17 of file scope.hpp.

13.213.1.2 SYCL_VENDOR_TRISYCL_SCOPE

```
#define SYCL_VENDOR_TRISYCL_SCOPE 1
```

This is an extension providing scope to devices & platforms.

Definition at line 20 of file scope.hpp.

13.214 scope.hpp

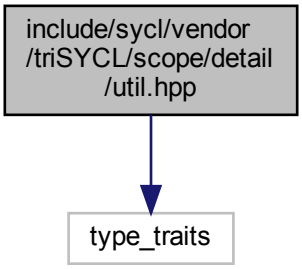
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_HPP
00003
00004 /** \file An extension adding conceptual scope storage to SYCL objects
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include "triSYCL/vendor/triSYCL/scope/device.hpp"
00013 #include "triSYCL/vendor/triSYCL/scope/platform.hpp"
00014 #include "triSYCL/vendor/triSYCL/scope/queue.hpp"
00015
00016 /// This is an extension providing a conceptual API for devices, platforms...
00017 #define SYCL_VENDOR_TRISYCL_CONCEPTUAL_API 1
00018
00019 /// This is an extension providing scope to devices & platforms
00020 #define SYCL_VENDOR_TRISYCL_SCOPE 1
00021
00022 /*
00023 # Some Emacs stuff:
00024 ### Local Variables:
00025 ### ispell-local-dictionary:  "american"
00026 ### eval:  (flyspell-prog-mode)
00027 ### End:
00028 */
00029
00030 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_HPP
```

13.215 include/sycl/vendor/triSYCL/scope/detail/util.hpp File Reference

`#include <type_traits>`

Include dependency graph for util.hpp:



Classes

- struct trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, StorageType >
- struct trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, typename ScopedType::storage_type >

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope
- namespace trisycl::vendor::trisycl::scope::detail

Typedefs

- template<typename ScopedType >
  using trisycl::vendor::trisycl::scope::detail::storage_type_trait_t = typename storage_type_trait< ScopedType >::type

  Retrieve the type of the scoped storage of the object or `void` *if none.*

13.216 util.hpp

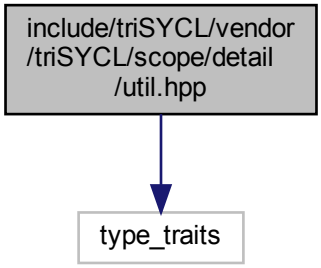
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DETAIL_UTIL_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DETAIL_UTIL_HPP
00003
00004 /** \file Some supporting content for conceptual scope storage
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <type_traits>
00013
00014 namespace trisycl::vendor::trisycl::scope::detail {
00015
00016 template <typename ScopedType, typename StorageType = void>
00017 struct storage_type_trait {
00018   using type = StorageType;
00019 };
00020
00021
00022 template <typename ScopedType>
00023 struct storage_type_trait<ScopedType, typename ScopedType::storage_type> {
00024   using type = typename ScopedType::storage_type;
00025 };
00026
00027
00028 /// Retrieve the type of the scoped storage of the object or \c void if none
00029 template <typename ScopedType>
00030 using storage_type_trait_t = typename storage_type_trait<ScopedType>::type;
00031
00032 }
00033
00034
00035 /*
00036 # Some Emacs stuff:
00037 ### Local Variables:
00038 ### ispell-local-dictionary:  "american"
00039 ### eval:  (flyspell-prog-mode)
00040 ### End:
00041 */
00042
00043 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DETAIL_UTIL_HPP
```

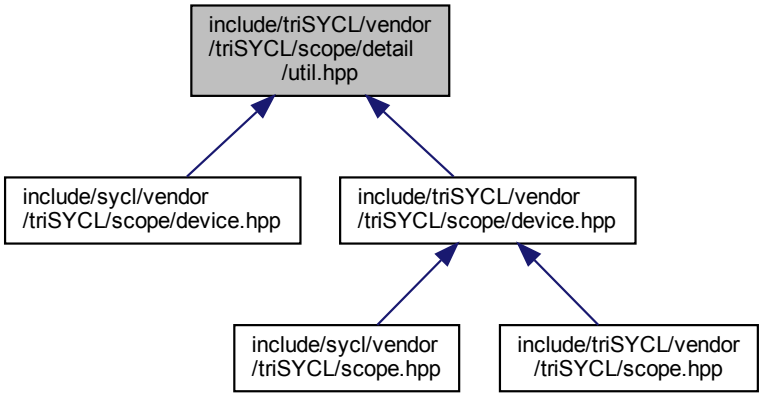
13.217 include/triSYCL/vendor/triSYCL/scope/detail/util.hpp File Reference

```
#include <type_traits>
```

Include dependency graph for util.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, StorageType >
- struct trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, typename ScopedType::storage_type >

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope
- namespace trisycl::vendor::trisycl::scope::detail

13.218 util.hpp

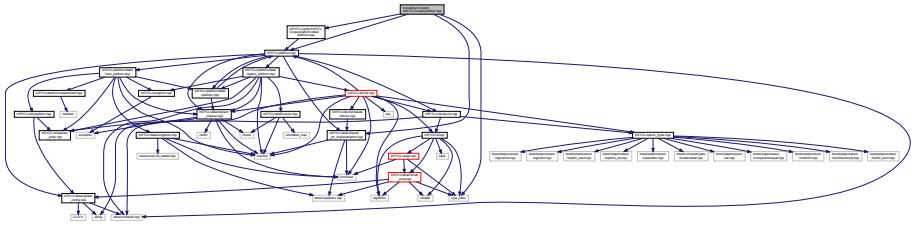
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DETAIL_UTIL_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DETAIL_UTIL_HPP
00003
00004 /** \file Some supporting content for conceptual scope storage
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <type_traits>
00013
00014 namespace trisycl::vendor::trisycl::scope::detail {
00015
00016 template <typename ScopedType, typename StorageType = void>
00017 struct storage_type_trait {
00018   using type = StorageType;
00019 };
00020
00021
00022 template <typename ScopedType>
00023 struct storage_type_trait<ScopedType, typename ScopedType::storage_type> {
00024   using type = typename ScopedType::storage_type;
00025 };
00026
00027
00028 /// Retrieve the type of the scoped storage of the object or \c void if none
00029 template <typename ScopedType>
00030 using storage_type_trait_t = typename storage_type_trait<ScopedType>::type;
00031
00032 }
00033
00034
00035 /*
00036 # Some Emacs stuff:
00037 ### Local Variables:
00038 ### ispell-local-dictionary:  "american"
00039 ### eval:  (flyspell-prog-mode)
00040 ### End:
00041 */
00042
00043 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DETAIL_UTIL_HPP
```

13.219 include/sycl/vendor/triSYCL/scope/platform.hpp File Reference

```
#include <type_traits>
#include "triSYCL/detail/shared_ptr_implementation.hpp"
#include "triSYCL/platform.hpp"
#include "triSYCL/vendor/triSYCL/scope/platform/detail/platform.hpp"
```

Include dependency graph for platform.hpp:



Classes

- struct trisycl::vendor::trisycl::scope::empty_platform_scope

  An empty platform scope to make compilation errors clearer. More...

- class trisycl::vendor::trisycl::scope::platform< PlatformStorage >

  A conceptual platform implementing some platform-scoped storage. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope

Macros

- #define SYCL_VENDOR_TRISYCL_PLATFORM_SCOPE 1

  This is an extension providing scope storage for platforms.

13.219.1 Macro Definition Documentation

13.219.1.1 SYCL_VENDOR_TRISYCL_PLATFORM_SCOPE

```
#define SYCL_VENDOR_TRISYCL_PLATFORM_SCOPE 1
```

This is an extension providing scope storage for platforms.

Definition at line 27 of file platform.hpp.

13.220 platform.hpp

Go to the documentation of this file.

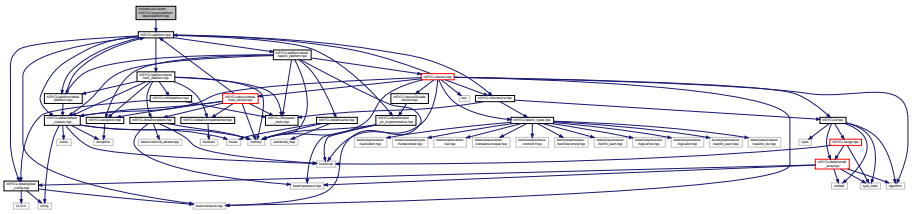
```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_PLATFORM_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_PLATFORM_HPP
00003
00004 /** \file The SYCL conceptual platform with a platform scope storage
00005
00006
00007 \todo Add some caching or prevent multiple creation of storage on
00008 the platform ?  Or is this a feature to have different scopes on
00009 the same platform?
00010
00011 \todo The device and platform are very similar.  Use a common
00012 mix-in to implement both of them?
00013
00014 Ronan at Keryell point FR
00015
00016 This file is distributed under the University of Illinois Open Source
00017 License.  See LICENSE.TXT for details.
00018 */
00019
00020 #include <type_traits>
00021
00022 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00023 #include "triSYCL/platform.hpp"
00024 #include "triSYCL/vendor/triSYCL/scope/platform/detail/platform.hpp"
00025
00026 /// This is an extension providing scope storage for platforms
00027 #define SYCL_VENDOR_TRISYCL_PLATFORM_SCOPE 1
00028
00029 namespace trisycl::vendor::trisycl::scope {
00030
00031 /** \addtogroup vendor_trisycl_scope triSYCL extension for storage scopes
00032 @{
00033 */
```

```
00034
00035 /// An empty platform scope to make compilation errors clearer
00036 struct empty_platform_scope {
00037 };
00038
00039
00040 /// A conceptual platform implementing some platform-scoped storage
00041 template <typename PlatformStorage = empty_platform_scope>
00042 class platform
00043   /* Use the underlying platform implementation that can be shared in the
00044 SYCL model */
00045   :  public
00046     ::trisycl::detail::shared_ptr_implementation<platform<PlatformStorage>,
00047                                                  detail::platform
00048                                                  <PlatformStorage>> {
00049
00050   using spi =
00051     ::trisycl::detail::shared_ptr_implementation<platform<PlatformStorage>,
00052                                                  detail::platform
00053                                                  <PlatformStorage>>;
00054
00055   // Allows the comparison operation to access the implementation
00056   friend spi;
00057
00058 public:
00059
00060   // Make the implementation member directly accessible in this class
00061   using spi::implementation;
00062
00063 /// The type of the scoped storage in the platform
00064   using storage_type = PlatformStorage;
00065
00066 /// The platform scope storage type
00067   using platform_scope_type = PlatformStorage;
00068
00069 /// True if there is some storage in the platform
00070   static constexpr bool has_some_storage_p =
00071     not std::is_same_v<PlatformStorage, empty_platform_scope>;
00072
00073 /** Construct the platform with some platform-scoped storage on top
00074 of a SYCL platform
00075
00076 \param[in] p is the real platform to use
00077 */
00078   platform(const ::trisycl::platform &p) :
00079     spi {
00080       new detail::platform<PlatformStorage> { p }
00081     } {}
00082
00083
00084 /// By default use the default (host) platform
00085   platform() :  platform(::trisycl::platform {}) {}
00086
00087
00088 /// It is still copyable
00089   platform(const platform&) = default;
00090
00091
00092 /// It is still movable
00093   platform(platform&&) = default;
00094
00095
00096 /// Access to the platform-scoped storage
00097   auto& get_storage()const {
00098     return implementation->get_storage();
00099   }
00100
00101
00102 /** Add a conversion to \c trisycl::platform& so it can be used as
00103 a normal platform */
00104     operator ::trisycl::platform&() const {
00105       return implementation->get_underlying_platform();
00106     }
00107
00108 };
00109
00110 /// @} to end the vendor_trisycl_scope Doxygen group
00111
00112 }
00113
00114
00115 /*
00116 # Some Emacs stuff:
00117 ### Local Variables:
00118 ### ispell-local-dictionary:  "american"
00119 ### eval:  (flyspell-prog-mode)
00120 ### End:
```

```
00121 */
00122
00123 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_PLATFORM_HPP
```

13.221 include/sycl/vendor/triSYCL/scope/platform/detail/platform.hpp File Reference

`#include "triSYCL/platform.hpp"`
Include dependency graph for platform.hpp:



Classes

- class trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >

  A conceptual platform implementing some platform-scoped storage. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope
- namespace trisycl::vendor::trisycl::scope::detail

13.222 platform.hpp

Go to the documentation of this file.

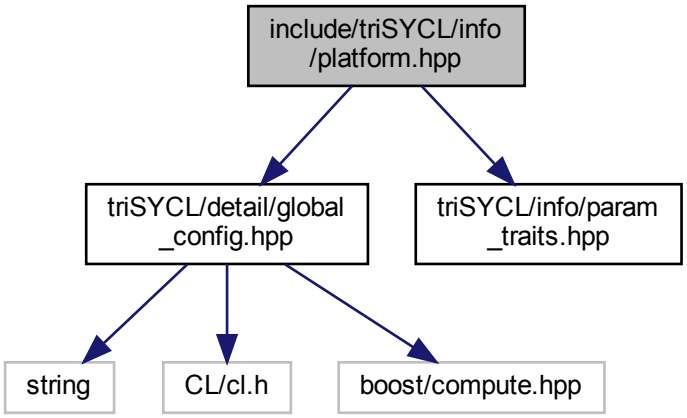
```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_PLATFORM_DETAIL_PLATFORM_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_PLATFORM_DETAIL_PLATFORM_HPP
00003
00004 /** \file Implementation details of the SYCL conceptual platform with a
00005 platform scope storage
00006
00007 \todo The device and platform are very similar.  Use a common
00008 mix-in to implement both of them?
00009
00010 Ronan at Keryell point FR
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include "triSYCL/platform.hpp"
00017
00018 namespace trisycl::vendor::trisycl::scope::detail {
00019
00020 /** \addtogroup vendor_trisycl_scope triSYCL extension for storage scopes
```

```
00021 @{
00022 */
00023
00024 /// A conceptual platform implementing some platform-scoped storage
00025 template <typename PlatformStorage>
00026 class platform {
00027 /// The storage-less platform behind the scene
00028   ::trisycl::platform p;
00029
00030 /** The platform-scoped storage default-initialized
00031
00032 \todo For now it is allocated on the host in this CPU emulation
00033 but a device compiler and run time can create this on a real
00034 device
00035 */
00036   PlatformStorage scope_storage;
00037
00038 public:
00039
00040 /** Construct the platform with some platform-scoped storage on top
00041 of a SYCL platform
00042
00043 \param[in] p is the real platform to use
00044 */
00045   platform(const ::trisycl::platform &p) :  p { p } {}
00046
00047 /// Get the platform behind the curtain
00048   auto get_underlying_platform() {
00049     return p;
00050   }
00051
00052
00053 /// Access to the platform-scoped storage
00054   auto& get_storage() {
00055     return scope_storage;
00056   }
00057
00058 };
00059
00060 /// @} to end the vendor_trisycl_scope Doxygen group
00061
00062 }
00063
00064
00065 /*
00066 # Some Emacs stuff:
00067 ### Local Variables:
00068 ### ispell-local-dictionary:  "american"
00069 ### eval:  (flyspell-prog-mode)
00070 ### End:
00071 */
00072
00073 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_PLATFORM_DETAIL_PLATFORM_HPP
```

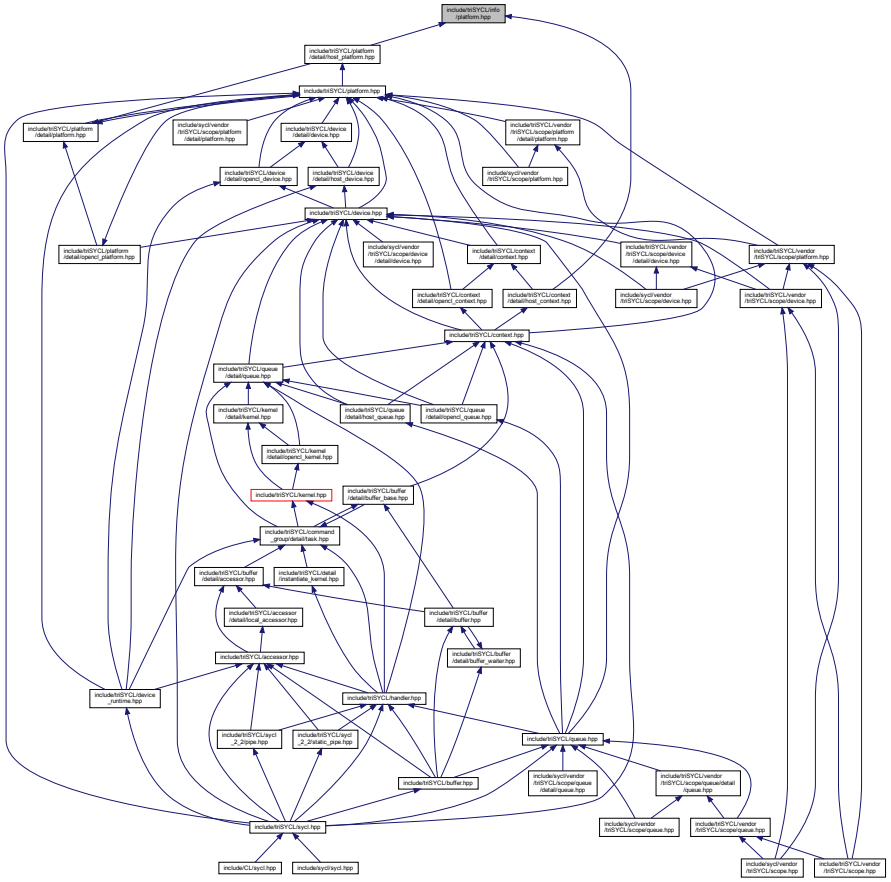
13.223 include/triSYCL/info/platform.hpp File Reference

```
#include "triSYCL/detail/global_config.hpp"
#include "triSYCL/info/param_traits.hpp"
```

Include dependency graph for platform.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::info

Enumerations

- enum class trisycl::info::platform : unsigned int { trisycl::info::platform::TRISYCL_OPENCL_ONLY =(= CL_PLATFORM_PROFILE) , trisycl::info::platform::TRISYCL_OPENCL_O =(= CL_PLATFORM_VERSION) , trisycl::info::platform::TRISYCL_OPENCL_ONLY =(= CL_PLATFORM_↩ NAME) , trisycl::info::platform::TRISYCL_OPENCL_ONLY =(= CL_PLATFORM_VENDOR) , trisycl::info::platform::TRISYCL_OPENCL_ONLY =(= CL_PLATFORM_EXTENSIONS) }

  Platform information descriptors.

13.224 platform.hpp

Go to the documentation of this file.

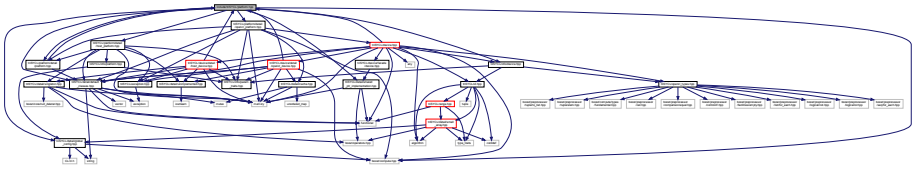
```
00001 #ifndef TRISYCL_SYCL_INFO_PLATFORM_HPP
00002 #define TRISYCL_SYCL_INFO_PLATFORM_HPP
00003
00004 /** \file The OpenCL SYCL platform information parameters
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include "triSYCL/detail/global_config.hpp"
00013 #include "triSYCL/info/param_traits.hpp"
00014
00015 namespace trisycl {
00016
00017 /** \addtogroup execution Platforms, contexts, devices and queues
00018 @{
00019 */
00020 namespace info {
00021
00022 /** Platform information descriptors
00023
00024 A SYCL platform can be queried for all of the following information
00025 using the get_info function.
00026
00027 In this implementation, the values are mapped to OpenCL values to
00028 avoid further remapping later when OpenCL is used
00029 */
00030 enum class platform :  unsigned int {
00031 /** Returns the profile name (as a string_class) supported by the
00032 implementation.
00033
00034 Can be either FULL PROFILE or EMBEDDED PROFILE.
00035 */
00036   profile TRISYCL_OPENCL_ONLY(= CL_PLATFORM_PROFILE),
00037
00038 /** Returns the OpenCL software driver version string in the form major
00039 number.minor number (as a string_class)
00040 */
00041   version TRISYCL_OPENCL_ONLY(= CL_PLATFORM_VERSION),
00042
00043 /** Returns the name of the platform (as a string_class)
00044 */
00045   name TRISYCL_OPENCL_ONLY(= CL_PLATFORM_NAME),
00046
00047 /** Returns the string provided by the platform vendor (as a string_class)
00048 */
00049   vendor TRISYCL_OPENCL_ONLY(= CL_PLATFORM_VENDOR),
00050
00051 /** Returns a space-separated list of extension names supported by the
00052 platform (as a string_class)
00053 */
00054   extensions TRISYCL_OPENCL_ONLY(= CL_PLATFORM_EXTENSIONS),
```

```
00055
00056 #if CL_SYCL_LANGUAGE_VERSION >= 220 && defined(CL_VERSION_2_1)
00057 /** Returns the resolution of the host timer in nanoseconds as used by
00058 clGetDeviceAndHostTimer
00059 */
00060   host_timer_resolution
00061     TRISYCL_OPENCL_ONLY(= CL_PLATFORM_HOST_TIMER_RESOLUTION)
00062 #endif
00063 };
00064
00065
00066 /** Query the return type for get_info() on platform parameter type
00067
00068 This defines the meta-function
00069 \code
00070 param_traits<info::platform x, string_class>::type == string_class
00071 \endcode
00072
00073 for all x, which means that get_info() returns always a string_class
00074 when asked about platform info.
00075 */
00076 TRISYCL_INFO_PARAM_TRAITS_ANY_T(info::platform, string_class)
00077 TRISYCL_INFO_PARAM_TRAITS(info::platform::extensions, vector_class<string_class>)
00078 #if CL_SYCL_LANGUAGE_VERSION >= 220 && defined(CL_VERSION_2_1)
00079 ///  get_info<host_timer_resolution>() return a cl_ulong
00080 #ifdef TRISYCL_OPENCL
00081 TRISYCL_INFO_PARAM_TRAITS(info::platform::host_timer_resolution, cl_ulong)
00082 #else
00083 TRISYCL_INFO_PARAM_TRAITS(info::platform::host_timer_resolution,
00084                           unsigned long int)
00085 #endif
00086 #endif
00087 }
00088 }
00089
00090 /*
00091 # Some Emacs stuff:
00092 ### Local Variables:
00093 ### ispell-local-dictionary:  "american"
00094 ### eval:  (flyspell-prog-mode)
00095 ### End:
00096 */
00097
00098 #endif // TRISYCL_SYCL_INFO_PLATFORM_HPP
```

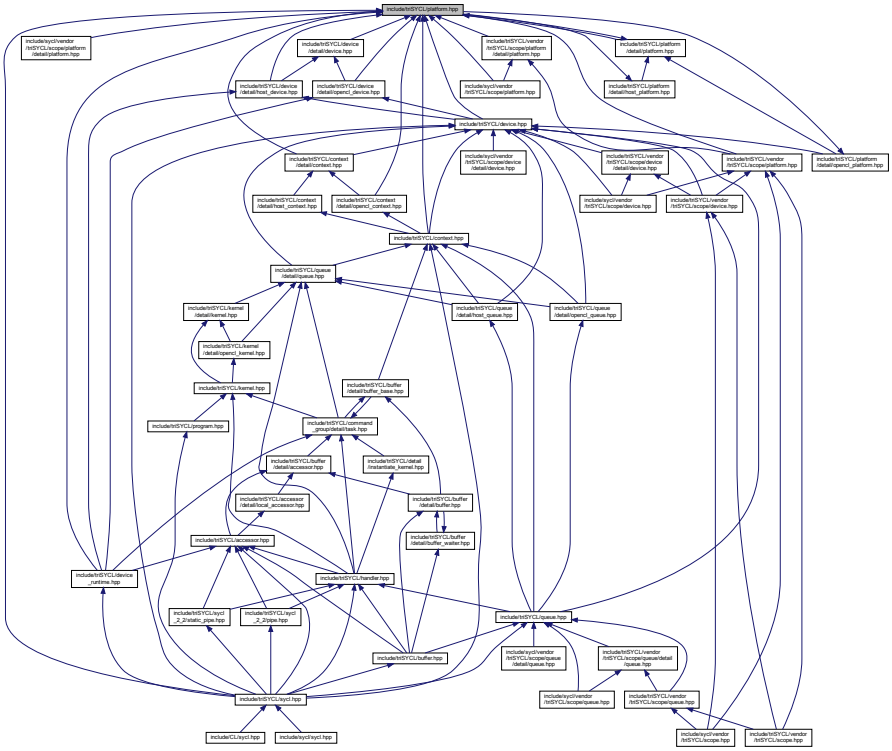
13.225 include/triSYCL/platform.hpp File Reference

```
#include <boost/compute.hpp>
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/detail/global_config.hpp"
#include "triSYCL/detail/shared_ptr_implementation.hpp"
#include "triSYCL/info/device.hpp"
#include "triSYCL/platform/detail/host_platform.hpp"
#include "triSYCL/platform/detail/opencl_platform.hpp"
#include "triSYCL/platform/detail/platform.hpp"
```

Include dependency graph for platform.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::platform

  Abstract the OpenCL platform. More...
- struct std::hash< trisycl::platform >

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace std

Macros

- #define PLATFORM_GET_INFO_STRING(X)

Functions

- template<> auto trisycl::platform::get_info< info::platform::extensions > () const

13.226 platform.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_PLATFORM_HPP
00002 #define TRISYCL_SYCL_PLATFORM_HPP
00003
00004 /** \file The OpenCL SYCL platform
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #ifdef TRISYCL_OPENCL
00013 #include <boost/compute.hpp>
00014 #endif
00015
00016 #include "triSYCL/detail/default_classes.hpp"
00017 #include "triSYCL/detail/global_config.hpp"
00018
00019 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00020 #include "triSYCL/info/device.hpp"
00021 #include "triSYCL/platform/detail/host_platform.hpp"
00022 #ifdef TRISYCL_OPENCL
00023 #include "triSYCL/platform/detail/opencl_platform.hpp"
00024 #endif
00025 #include "triSYCL/platform/detail/platform.hpp"
00026
00027 namespace trisycl {
00028
00029 class device_selector;
00030 class device;
00031
00032 /** \addtogroup execution Platforms, contexts, devices and queues
00033 @{
00034 */
00035
00036 /** Abstract the OpenCL platform
00037
00038 \todo triSYCL Implementation
00039 */
00040 class platform
00041   /* Use the underlying platform implementation that can be shared in the
00042 SYCL model */
00043   :  public detail::shared_ptr_implementation<platform, detail::platform> {
00044
00045   // Allows the comparison operation to access the implementation
00046   friend shared_ptr_implementation;
00047
00048 public:
00049
00050   // Make the implementation member directly accessible in this class
00051   using shared_ptr_implementation::implementation;
00052
00053 /** Default constructor for platform which is the host platform
00054
00055 Returns errors via the SYCL exception class.
00056 */
00057   platform() :
00058     shared_ptr_implementation { detail::host_platform::instance() } {}
00059
00060
00061 #ifdef TRISYCL_OPENCL
00062 /** Construct a platform class instance using cl_platform_id of the
00063 OpenCL device
00064
00065 Return synchronous errors via the SYCL exception class.
00066
00067 Retain a reference to the OpenCL platform.
00068 */
00069   explicit platform(cl_platform_id platform_id)
00070     :  platform { boost::compute::platform { platform_id } } {}
00071
00072
00073 /** Construct a platform class instance using a boost::compute::platform
00074
00075 This is a triSYCL extension for boost::compute interoperation.
00076
00077 Return synchronous errors via the SYCL exception class.
00078 */
00079   platform(const boost::compute::platform &p)
00080     :  shared_ptr_implementation { detail::opencl_platform::instance(p) } {}
00081 #endif
00082
```

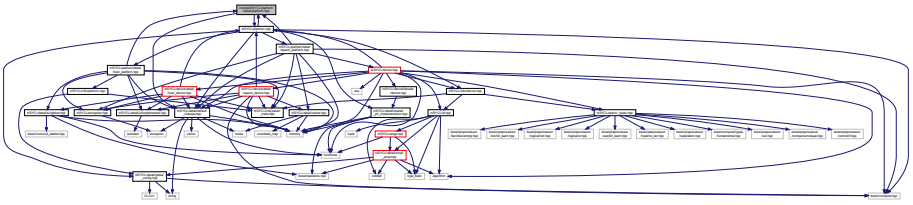
```
00083
00084 /**  Construct a platform object from the device selected by a device
00085 selector of the user's choice
00086
00087 Returns errors via the SYCL exception class.
00088 */
00089   explicit platform(const device_selector &dev_selector);
00090
00091 #ifdef TRISYCL_OPENCL
00092 /** Returns the cl_platform_id of the underlying OpenCL platform
00093
00094 If the platform is not a valid OpenCL platform, for example if it is
00095 the SYCL host, an exception is thrown
00096
00097 \todo Define a SYCL exception for this
00098 */
00099   cl_platform_id get()const {
00100     return implementation->get();
00101   }
00102
00103
00104 /** Return the underlying Boost.Compute platform if it is an
00105 OpenCL platform
00106
00107 This is a triSYCL extension
00108 */
00109   const boost::compute::platform get_boost_compute()const {
00110     return implementation->get_boost_compute();
00111   }
00112 #endif
00113
00114
00115 /// Get the list of all the platforms available to the application
00116   static vector_class<platform> get_platforms() {
00117     // Start with the default platform
00118     vector_class<platform> platforms { {} };
00119
00120 #ifdef TRISYCL_OPENCL
00121     // Then add all the OpenCL platforms
00122     for (const auto &d :  boost::compute::system::platforms())
00123       platforms.emplace_back(d);
00124 #endif
00125
00126     return platforms;
00127   }
00128
00129
00130 /** Get the OpenCL information about the requested parameter
00131
00132 \todo Add to the specification
00133 */
00134   template <info::platform param>
00135   inline auto get_info() const;
00136
00137 /// Test if an extension is available on the platform
00138   bool has_extension(const string_class &extension)const {
00139     return implementation->has_extension(extension);
00140   }
00141
00142
00143 /// Test if this platform is a host platform
00144   bool is_host()const {
00145     return implementation->is_host();
00146   }
00147
00148
00149 /** Get all the available devices for this platform
00150
00151 \param[in] device_type is the device type to filter the selection
00152 or \c info::device_type::all by default to return all the
00153 devices
00154
00155 \return the device list
00156 */
00157   vector_class<device>
00158   get_devices(info::device_type device_type = info::device_type::all) const;
00159 };
00160
00161 template<>
00162 inline auto platform::get_info<info::platform::extensions>()const {
00163   return implementation->get_extension_strings();
00164 }
00165
00166 #define PLATFORM_GET_INFO_STRING(X)             \
00167 template<>                                      \
00168 inline auto platform::get_info<info::platform::X>() const { \
00169 return implementation->get_info_string(info::platform::X);    \
```

```
00170 }
00171
00172 PLATFORM_GET_INFO_STRING(profile)
00173 PLATFORM_GET_INFO_STRING(version)
00174 PLATFORM_GET_INFO_STRING(name)
00175 PLATFORM_GET_INFO_STRING(vendor)
00176
00177 /// @} to end the execution Doxygen group
00178
00179 }
00180
00181
00182 /* Inject a custom specialization of std::hash to have the buffer
00183 usable into an unordered associative container
00184
00185 \todo Add this to the spec
00186 */
00187 namespace std {
00188
00189 template <> struct hash<trisycl::platform> {
00190
00191   auto operator()(const trisycl::platform &p)const {
00192     // Forward the hashing to the implementation
00193     return p.hash();
00194   }
00195
00196 };
00197
00198 }
00199
00200 /*
00201 # Some Emacs stuff:
00202 ### Local Variables:
00203 ### ispell-local-dictionary:  "american"
00204 ### eval:  (flyspell-prog-mode)
00205 ### End:
00206 */
00207
00208 #endif // TRISYCL_SYCL_PLATFORM_HPP
```

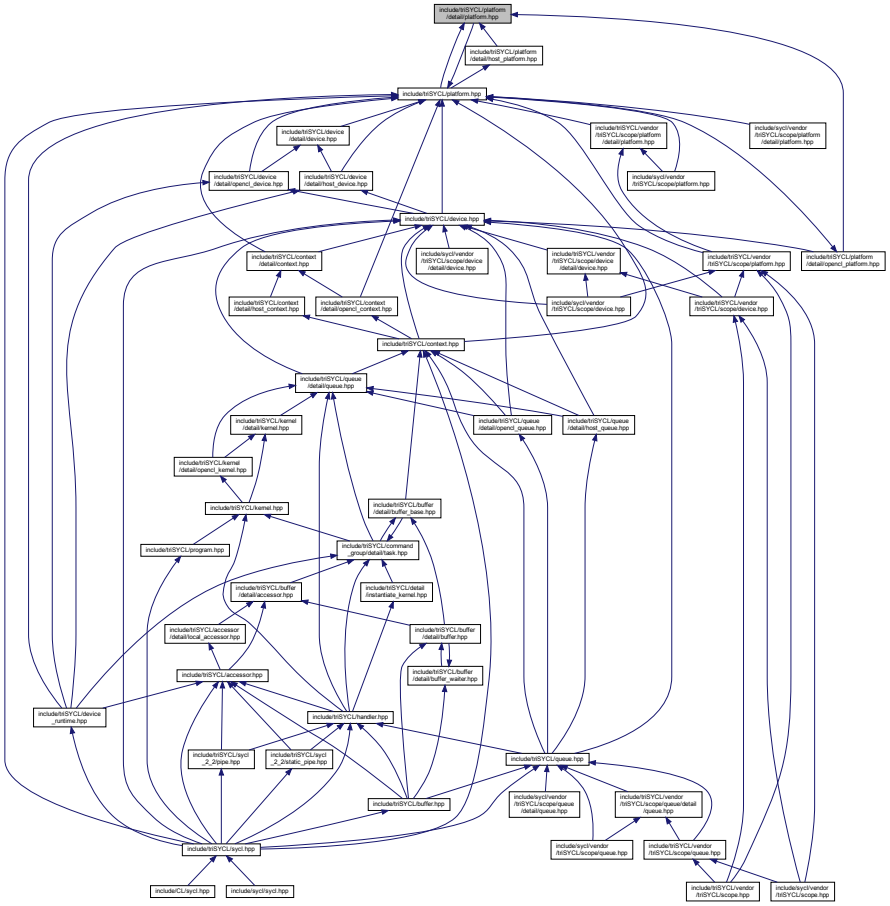
13.227 include/triSYCL/platform/detail/platform.hpp File Reference

```
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/platform.hpp"
```

Include dependency graph for platform.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::platform

  An abstract class representing various models of SYCL platforms. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.228 platform.hpp

Go to the documentation of this file.

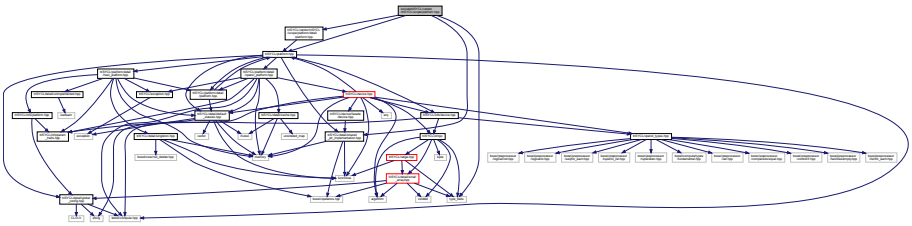
```
00001 #ifndef TRISYCL_SYCL_PLATFORM_DETAIL_PLATFORM_HPP
00002 #define TRISYCL_SYCL_PLATFORM_DETAIL_PLATFORM_HPP
00003
00004 /** \file The OpenCL SYCL abstract platform
00005
```

```
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include "triSYCL/detail/default_classes.hpp"
00013
00014 #include "triSYCL/platform.hpp"
00015
00016 namespace trisycl {
00017
00018 class device;
00019 class device_selector;
00020
00021 namespace detail {
00022
00023 /** \addtogroup execution Platforms, contexts, devices and queues
00024 @{
00025 */
00026
00027 /// An abstract class representing various models of SYCL platforms
00028 class platform {
00029
00030 public:
00031
00032 #ifdef TRISYCL_OPENCL
00033 /// Return the cl_platform_id of the underlying OpenCL platform
00034   virtual cl_platform_id get() const = 0;
00035
00036
00037 /// Return the underlying Boost.Compute platform, if any
00038   virtual const boost::compute::platform &get_boost_compute() const = 0;
00039 #endif
00040
00041
00042 /// Return true if the platform is a SYCL host platform
00043   virtual bool is_host() const = 0;
00044
00045
00046 /// Query the platform for OpenCL string info::platform info
00047   virtual string_class get_info_string(info::platform param) const = 0;
00048
00049   virtual vector_class<string_class> get_extension_strings() const = 0;
00050
00051 /// Specify whether a specific extension is supported on the platform.
00052   virtual bool has_extension(const string_class &extension) const = 0;
00053
00054 /** Get all the available devices for this platform
00055
00056 \param[in] device_selector is used to filter the selection.
00057
00058 \return the device list
00059 */
00060   virtual vector_class<device>
00061   get_devices(const device_selector &device_selector) const = 0;
00062
00063   // Virtual to call the real destructor
00064   virtual ~platform() {}
00065
00066 };
00067
00068 /// @} to end the execution Doxygen group
00069
00070 }
00071 }
00072
00073 /*
00074 # Some Emacs stuff:
00075 ### Local Variables:
00076 ### ispell-local-dictionary:  "american"
00077 ### eval:  (flyspell-prog-mode)
00078 ### End:
00079 */
00080
00081 #endif // TRISYCL_SYCL_PLATFORM_DETAIL_PLATFORM_HPP
```

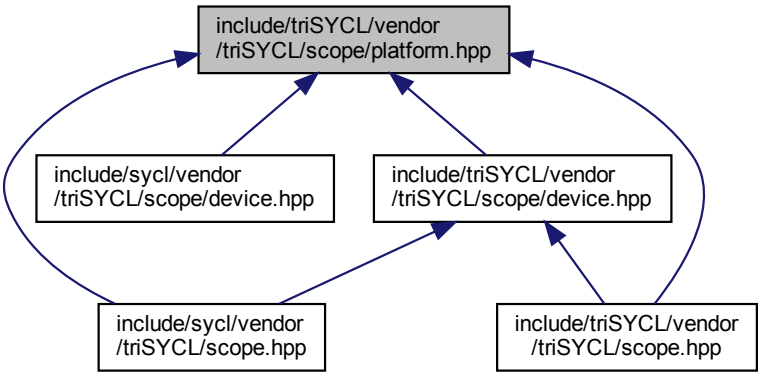
13.229 include/triSYCL/vendor/triSYCL/scope/platform.hpp File Reference

```
#include <type_traits>
#include "triSYCL/detail/shared_ptr_implementation.hpp"
#include "triSYCL/platform.hpp"
#include "triSYCL/vendor/triSYCL/scope/platform/detail/platform.hpp"
```

Include dependency graph for platform.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::trisycl::scope::empty_platform_scope

  An empty platform scope to make compilation errors clearer. More...
- class trisycl::vendor::trisycl::scope::platform< PlatformStorage >

  A conceptual platform implementing some platform-scoped storage. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope

Macros

- #define SYCL_VENDOR_TRISYCL_PLATFORM_SCOPE 1

  This is an extension providing scope storage for platforms.

13.229.1 Macro Definition Documentation

13.229.1.1 SYCL_VENDOR_TRISYCL_PLATFORM_SCOPE

```
#define SYCL_VENDOR_TRISYCL_PLATFORM_SCOPE 1
```

This is an extension providing scope storage for platforms.

Definition at line 27 of file platform.hpp.

13.230 platform.hpp

Go to the documentation of this file.

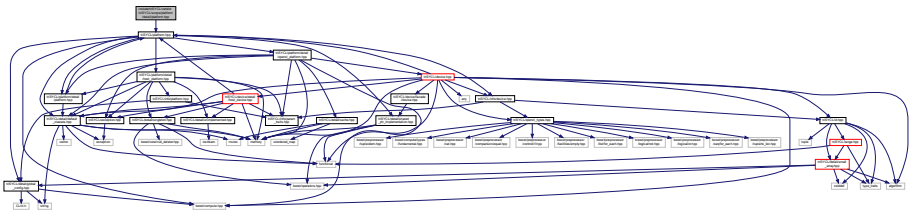
```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_PLATFORM_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_PLATFORM_HPP
00003
00004 /** \file The SYCL conceptual platform with a platform scope storage
00005
00006
00007 \todo Add some caching or prevent multiple creation of storage on
00008 the platform ?  Or is this a feature to have different scopes on
00009 the same platform?
00010
00011 \todo The device and platform are very similar.  Use a common
00012 mix-in to implement both of them?
00013
00014 Ronan at Keryell point FR
00015
00016 This file is distributed under the University of Illinois Open Source
00017 License.  See LICENSE.TXT for details.
00018 */
00019
00020 #include <type_traits>
00021
00022 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00023 #include "triSYCL/platform.hpp"
00024 #include "triSYCL/vendor/triSYCL/scope/platform/detail/platform.hpp"
00025
00026 /// This is an extension providing scope storage for platforms
00027 #define SYCL_VENDOR_TRISYCL_PLATFORM_SCOPE 1
00028
00029 namespace trisycl::vendor::trisycl::scope {
00030
00031 /** \addtogroup vendor_trisycl_scope triSYCL extension for storage scopes
00032 @{
00033 */
00034
00035 /// An empty platform scope to make compilation errors clearer
00036 struct empty_platform_scope {
00037 };
00038
00039
00040 /// A conceptual platform implementing some platform-scoped storage
00041 template <typename PlatformStorage = empty_platform_scope>
00042 class platform
00043   /* Use the underlying platform implementation that can be shared in the
00044 SYCL model */
00045   :  public
00046     ::trisycl::detail::shared_ptr_implementation<platform<PlatformStorage>,
00047                                                  detail::platform
00048                                                  <PlatformStorage>> {
```

```
00049
00050   using spi =
00051     ::trisycl::detail::shared_ptr_implementation<platform<PlatformStorage>,
00052                                                  detail::platform
00053                                                  <PlatformStorage»;
00054
00055   // Allows the comparison operation to access the implementation
00056   friend spi;
00057
00058 public:
00059
00060   // Make the implementation member directly accessible in this class
00061   using spi::implementation;
00062
00063 /// The type of the scoped storage in the platform
00064   using storage_type = PlatformStorage;
00065
00066 /// The platform scope storage type
00067   using platform_scope_type = PlatformStorage;
00068
00069 /// True if there is some storage in the platform
00070   static constexpr bool has_some_storage_p =
00071     not std::is_same_v<PlatformStorage, empty_platform_scope>;
00072
00073 /** Construct the platform with some platform-scoped storage on top
00074 of a SYCL platform
00075
00076 \param[in] p is the real platform to use
00077 */
00078   platform(const ::trisycl::platform &p) :
00079     spi {
00080       new detail::platform<PlatformStorage> { p }
00081     } {}
00082
00083
00084 /// By default use the default (host) platform
00085   platform() :  platform(::trisycl::platform {}) {}
00086
00087
00088 /// It is still copyable
00089   platform(const platform&) = default;
00090
00091
00092 /// It is still movable
00093   platform(platform&&) = default;
00094
00095
00096 /// Access to the platform-scoped storage
00097   auto& get_storage()const {
00098     return implementation->get_storage();
00099   }
00100
00101
00102 /** Add a conversion to \c trisycl::platform& so it can be used as
00103 a normal platform */
00104     operator ::trisycl::platform&() const {
00105       return implementation->get_underlying_platform();
00106     }
00107
00108 };
00109
00110 /// @} to end the vendor_trisycl_scope Doxygen group
00111
00112 }
00113
00114
00115 /*
00116 # Some Emacs stuff:
00117 ### Local Variables:
00118 ### ispell-local-dictionary:  "american"
00119 ### eval:  (flyspell-prog-mode)
00120 ### End:
00121 */
00122
00123 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_PLATFORM_HPP
```

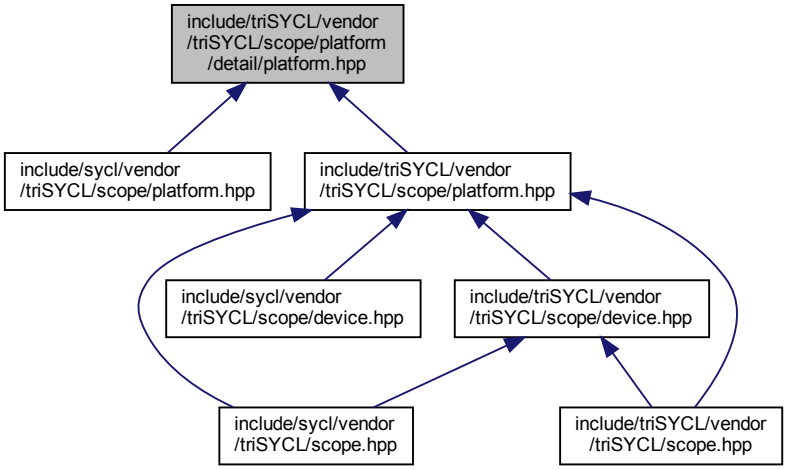
13.231 include/triSYCL/vendor/triSYCL/scope/platform/detail/platform.hpp File Reference

```
#include "triSYCL/platform.hpp"
```

Include dependency graph for platform.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >

  A conceptual platform implementing some platform-scoped storage. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope
- namespace trisycl::vendor::trisycl::scope::detail

13.232 platform.hpp

Go to the documentation of this file.

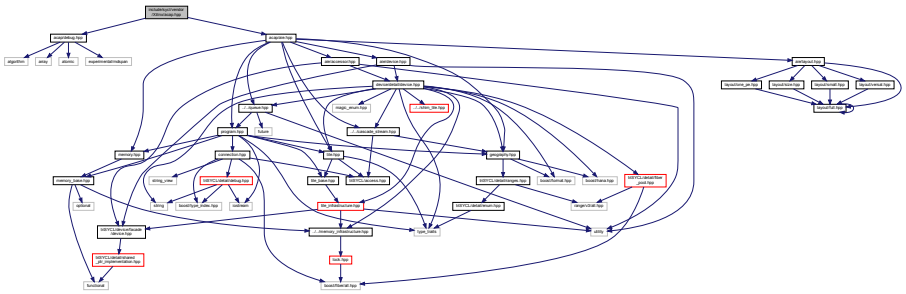
```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_PLATFORM_DETAIL_PLATFORM_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_PLATFORM_DETAIL_PLATFORM_HPP
00003
00004 /** \file Implementation details of the SYCL conceptual platform with a
00005 platform scope storage
00006
00007 \todo The device and platform are very similar.  Use a common
00008 mix-in to implement both of them?
00009
00010 Ronan at Keryell point FR
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include "triSYCL/platform.hpp"
00017
00018 namespace trisycl::vendor::trisycl::scope::detail {
00019
00020 /** \addtogroup vendor_trisycl_scope triSYCL extension for storage scopes
00021 @{
00022 */
00023
00024 /// A conceptual platform implementing some platform-scoped storage
00025 template <typename PlatformStorage>
00026 class platform {
00027 /// The storage-less platform behind the scene
00028   ::trisycl::platform p;
00029
00030 /** The platform-scoped storage default-initialized
00031
00032 \todo For now it is allocated on the host in this CPU emulation
00033 but a device compiler and run time can create this on a real
00034 device
00035 */
00036   PlatformStorage scope_storage;
00037
00038 public:
00039
00040 /** Construct the platform with some platform-scoped storage on top
00041 of a SYCL platform
00042
00043 \param[in] p is the real platform to use
00044 */
00045   platform(const ::trisycl::platform &p) :  p { p } {}
00046
00047 /// Get the platform behind the curtain
00048   auto get_underlying_platform() {
00049     return p;
00050   }
00051
00052
00053 /// Access to the platform-scoped storage
00054   auto& get_storage() {
00055     return scope_storage;
00056   }
00057
00058 };
00059
00060 /// @} to end the vendor_trisycl_scope Doxygen group
00061
00062 }
00063
00064
00065 /*
00066 # Some Emacs stuff:
00067 ### Local Variables:
00068 ### ispell-local-dictionary:  "american"
00069 ### eval:  (flyspell-prog-mode)
00070 ### End:
00071 */
00072
00073 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_PLATFORM_DETAIL_PLATFORM_HPP
```

13.233 include/sycl/vendor/Xilinx/acap.hpp File Reference

Experimenting with CGRA & processor arrays such as ACAP/AI Engine.

```
#include "acap/debug.hpp"
#include "acap/aie.hpp"
```

Include dependency graph for acap.hpp:



13.233.1 Detailed Description

Experimenting with CGRA & processor arrays such as ACAP/AI Engine.

Definition in file acap.hpp.

13.234 acap.hpp

Go to the documentation of this file.

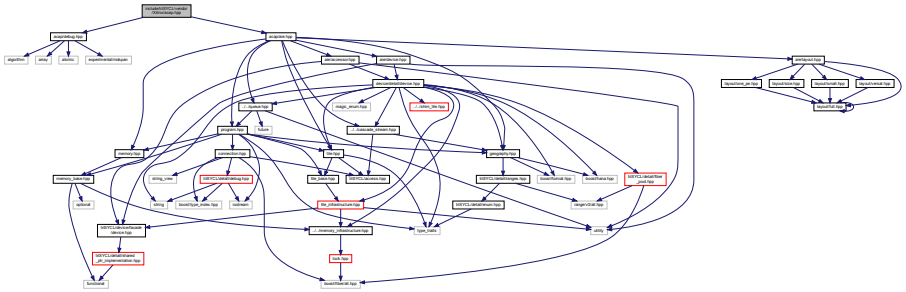
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_HPP
00003
00004 /** \file
00005
00006 Experimenting with CGRA & processor arrays such as ACAP/AI Engine
00007 */
00008
00009 #include "acap/debug.hpp"
00010 #include "acap/aie.hpp"
00011
00012 /*
00013 # Some Emacs stuff:
00014 ### Local Variables:
00015 ### ispell-local-dictionary:  "american"
00016 ### eval:  (flyspell-prog-mode)
00017 ### End:
00018 */
00019
00020 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_HPP
```

13.235 include/triSYCL/vendor/Xilinx/acap.hpp File Reference

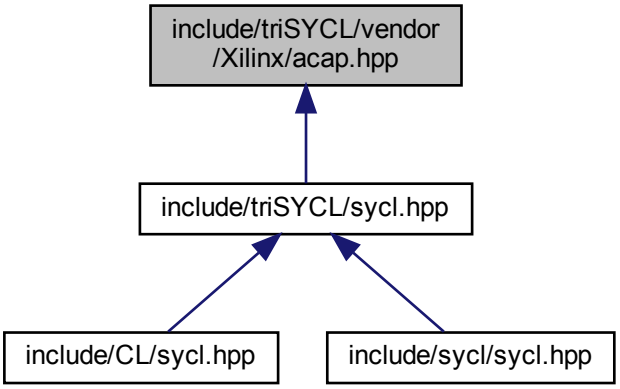
Experimenting with CGRA & processor arrays such as ACAP/AI Engine.

```
#include "acap/debug.hpp"
#include "acap/aie.hpp"
```

Include dependency graph for acap.hpp:



This graph shows which files directly or indirectly include this file:



13.235.1 Detailed Description

Experimenting with CGRA & processor arrays such as ACAP/AI Engine.

Definition in file acap.hpp.

13.236 acap.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_HPP
00003
00004 /** \file
00005
00006 Experimenting with CGRA & processor arrays such as ACAP/AI Engine
00007 */
00008
00009 #include "acap/debug.hpp"
00010 #include "acap/aie.hpp"
00011
```

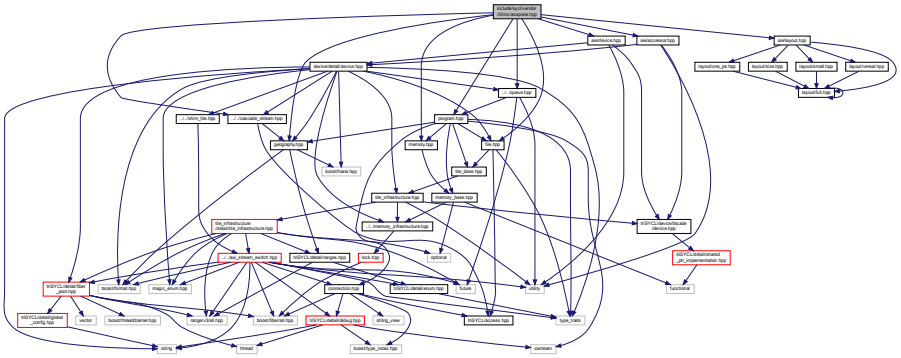
```
00012 /*
00013 # Some Emacs stuff:
00014 ### Local Variables:
00015 ### ispell-local-dictionary:  "american"
00016 ### eval:  (flyspell-prog-mode)
00017 ### End:
00018 */
00019
00020 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_HPP
```

13.237 include/sycl/vendor/Xilinx/acap/aie.hpp File Reference

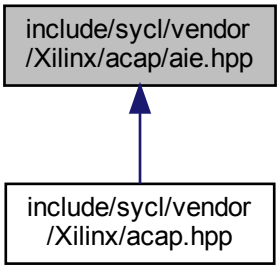
Experimenting with CGRA & processor arrays such as ACAP/AI Engine.

```
#include "aie/accessor.hpp"
#include "aie/cascade_stream.hpp"
#include "aie/device.hpp"
#include "aie/geography.hpp"
#include "aie/layout.hpp"
#include "aie/memory.hpp"
#include "aie/program.hpp"
#include "aie/queue.hpp"
#include "aie/tile.hpp"
```

Include dependency graph for aie.hpp:



This graph shows which files directly or indirectly include this file:



13.237.1 Detailed Description

Experimenting with CGRA & processor arrays such as ACAP/AI Engine.

Definition in file aie.hpp.

13.238 aie.hpp

Go to the documentation of this file.

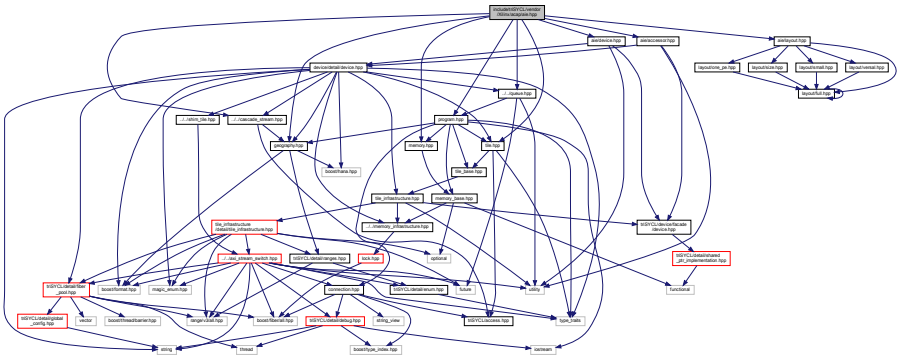
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_HPP
00003
00004 /** \file
00005
00006 Experimenting with CGRA & processor arrays such as ACAP/AI Engine
00007 */
00008
00009 #include "aie/accessor.hpp"
00010 #include "aie/cascade_stream.hpp"
00011 #include "aie/device.hpp"
00012 #include "aie/geography.hpp"
00013 #include "aie/layout.hpp"
00014 #include "aie/memory.hpp"
00015 #include "aie/program.hpp"
00016 #include "aie/queue.hpp"
00017 #include "aie/tile.hpp"
00018
00019 /*
00020 # Some Emacs stuff:
00021 ### Local Variables:
00022 ### ispell-local-dictionary:  "american"
00023 ### eval:  (flyspell-prog-mode)
00024 ### End:
00025 */
00026
00027 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_HPP
```

13.239 include/triSYCL/vendor/Xilinx/acap/aie.hpp File Reference

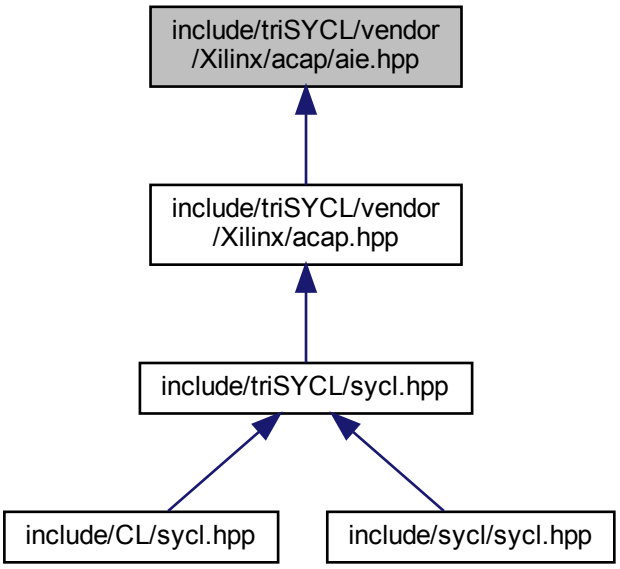
Experimenting with CGRA & processor arrays such as ACAP/AI Engine.

```
#include "aie/accessor.hpp"
#include "aie/cascade_stream.hpp"
#include "aie/device.hpp"
#include "aie/geography.hpp"
#include "aie/layout.hpp"
#include "aie/memory.hpp"
#include "aie/program.hpp"
#include "aie/queue.hpp"
#include "aie/tile.hpp"
```

Include dependency graph for aie.hpp:



This graph shows which files directly or indirectly include this file:



13.239.1 Detailed Description

Experimenting with CGRA & processor arrays such as ACAP/AI Engine.

Definition in file aie.hpp.

13.240 aie.hpp

Go to the documentation of this file.

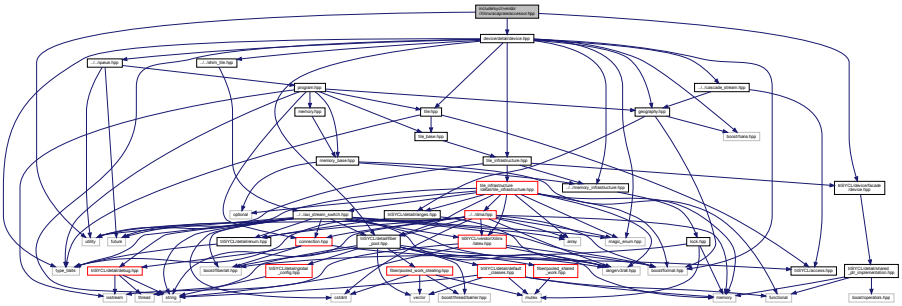
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_HPP
00003
00004 /** \file
00005
00006 Experimenting with CGRA & processor arrays such as ACAP/AI Engine
00007 */
00008
00009 #include "aie/accessor.hpp"
00010 #include "aie/cascade_stream.hpp"
00011 #include "aie/device.hpp"
00012 #include "aie/geography.hpp"
00013 #include "aie/layout.hpp"
00014 #include "aie/memory.hpp"
00015 #include "aie/program.hpp"
00016 #include "aie/queue.hpp"
00017 #include "aie/tile.hpp"
00018
00019 /*
00020 # Some Emacs stuff:
00021 ### Local Variables:
00022 ### ispell-local-dictionary:  "american"
00023 ### eval:  (flyspell-prog-mode)
00024 ### End:
00025 */
00026
00027 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_HPP
```

13.241 include/sycl/vendor/Xilinx/acap/aie/accessor.hpp File Reference

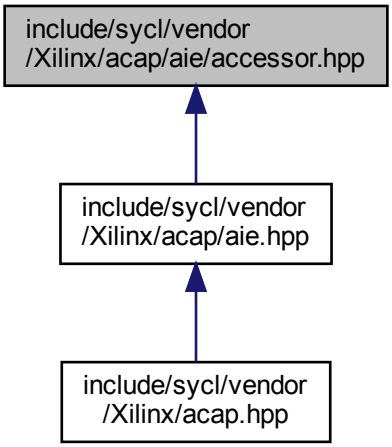
SYCL accessor for AIE tile device.

```
#include <utility>
#include "device/detail/device.hpp"
#include "triSYCL/device/facade/device.hpp"
```

Include dependency graph for accessor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::accessor< Buffer >

  Create a SYCL-like device view of an AI Engine CGRA with some layout.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.241.1 Detailed Description

SYCL accessor for AIE tile device.

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file accessor.hpp.

13.242 accessor.hpp

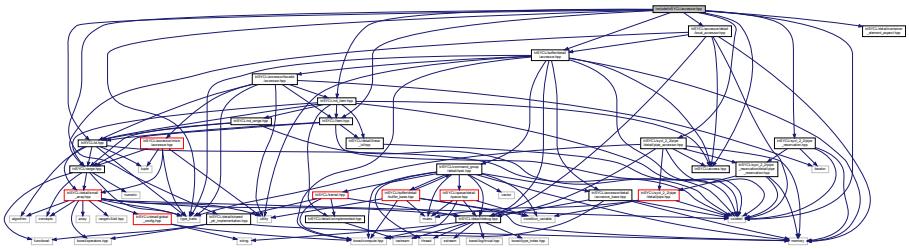
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ACCESSOR_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ACCESSOR_HPP
00003
00004 /** \file
00005
00006 SYCL accessor for AIE tile device
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <utility>
00013
00014 #include "device/detail/device.hpp"
00015 #include "triSYCL/device/facade/device.hpp"
00016
00017 /// \ingroup aie
00018 /// @{
00019
00020 namespace trisycl::vendor::xilinx::acap::aie {
00021
00022 /** Create a SYCL-like device view of an AI Engine CGRA with some layout
00023
00024 \param Layout is the layout description of the machine to
00025 instantiate with the physical size
00026 */
00027 template <typename Buffer>
00028 class accessor :  public ::trisycl::accessor<typename Buffer::value_type,
00029                                               Buffer::rank(),
00030                                               access::mode::read_write,
00031                                               access::target::host_buffer> {
00032
00033   using base = ::trisycl::accessor<typename Buffer::value_type,
00034                                    Buffer::rank(),
00035                                    access::mode::read_write,
00036                                    access::target::host_buffer>;
00037  public:
00038
00039   using typename base::accessor;
00040   accessor(Buffer& b, auto& cgh)
00041     :  base { b } {}
00042 };
00043
00044 /// @} End the aie Doxygen group
00045
00046 } // namespace trisycl::vendor::xilinx::acap::aie
00047
00048 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ACCESSOR_HPP
```

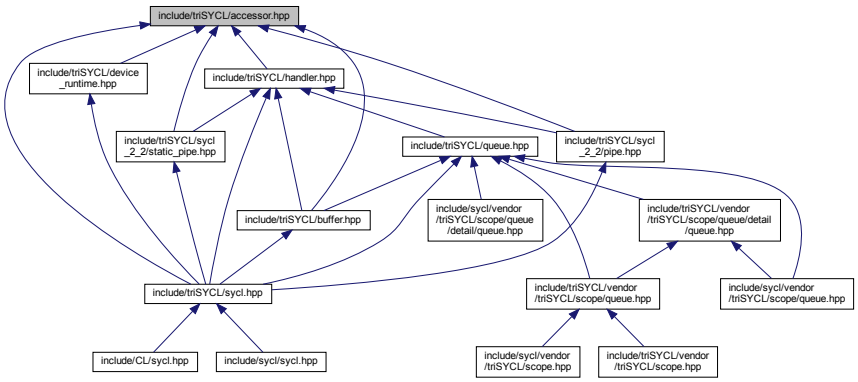
13.243 include/triSYCL/accessor.hpp File Reference

```
#include <cstddef>
#include <type_traits>
#include "triSYCL/access.hpp"
#include "triSYCL/accessor/detail/local_accessor.hpp"
#include "triSYCL/buffer/detail/accessor.hpp"
#include "triSYCL/detail/container_element_aspect.hpp"
#include "triSYCL/detail/shared_ptr_implementation.hpp"
#include "triSYCL/id.hpp"
#include "triSYCL/item.hpp"
#include "triSYCL/nd_item.hpp"
#include "triSYCL/sycl_2_2/pipe_reservation.hpp"
#include "triSYCL/sycl_2_2/pipe/detail/pipe_accessor.hpp"
```

Include dependency graph for accessor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::accessor< DataType, Dimensions, AccessMode, Target >

  The accessor abstracts the way buffer or pipe data are accessed inside a kernel in a multidimensional variable length array way. More...
- class trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >

  The pipe accessor abstracts the way pipe data are accessed inside a kernel. More...
- class trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >

  The pipe accessor abstracts the way pipe data are accessed inside a kernel. More...
- class trisycl::host_accessor< DataType, Dimensions, AccessMode >

  SYCL 2020 host_accessor. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::sycl_2_2

Functions

- template<typename Accessor >
  static auto & trisycl::get_pipe_detail (Accessor &a)

  Top-level function to break circular dependencies on the the types to get the pipe implementation.

13.244 accessor.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_ACCESSOR_HPP
00002 #define TRISYCL_SYCL_ACCESSOR_HPP
00003
00004 /** \file The SYCL accessor<>
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <cstddef>
00013 #include <type_traits>
00014
00015 #include "triSYCL/access.hpp"
00016 #include "triSYCL/accessor/detail/local_accessor.hpp"
00017 #include "triSYCL/buffer/detail/accessor.hpp"
00018 #include "triSYCL/detail/container_element_aspect.hpp"
00019 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00020 #include "triSYCL/id.hpp"
00021 #include "triSYCL/item.hpp"
00022 #include "triSYCL/nd_item.hpp"
00023 #include "triSYCL/sycl_2_2/pipe_reservation.hpp"
00024 #include "triSYCL/sycl_2_2/pipe/detail/pipe_accessor.hpp"
00025
00026 namespace trisycl {
00027
00028 template <typename T, int Dimensions, typename Allocator>
00029 class buffer;
00030
00031 namespace sycl_2_2 {
00032
00033 template <typename T>
00034 class pipe;
00035
00036 }
00037
00038 class handler;
00039
00040 /** \addtogroup data Data access and storage in SYCL
00041 @{
00042 */
00043
00044 /** The accessor abstracts the way buffer or pipe data are accessed
00045 inside a kernel in a multidimensional variable length array way.
00046
00047 \todo Implement it for images according so section 3.3.4.5
00048 */
00049 template <typename DataType, int Dimensions,
00050           access::mode AccessMode =
00051               (std::is_const_v<DataType> ?  access::mode::read
00052                                          :  access::mode::read_write),
00053           access::target Target = access::target::global_buffer>
00054 class accessor
00055     :  public detail::shared_ptr_implementation<
00056           accessor<DataType, Dimensions, AccessMode, Target>,
00057           detail::accessor<DataType, Dimensions, AccessMode, Target>>
00058     , public detail::container_element_aspect<DataType> {
```

```
00059
00060  public:
00061
00062 /// \todo in the specification:  store the dimension for user request
00063   static constexpr auto dimensionality = Dimensions;
00064
00065  private:
00066
00067   using accessor_detail = typename detail::accessor<DataType,
00068                                                     Dimensions,
00069                                                     AccessMode,
00070                                                     Target>;
00071
00072   // The type encapsulating the implementation
00073   using implementation_t = typename accessor::shared_ptr_implementation;
00074
00075   // Allows the comparison operation to access the implementation
00076   friend implementation_t;
00077
00078  public:
00079 /// Introspect the \c access_mode
00080   auto static constexpr access_mode() { return AccessMode; }
00081
00082 /// Introspect the \c target
00083   auto static constexpr target() { return Target; }
00084
00085   // Make the implementation member directly accessible in this class
00086   using implementation_t::implementation;
00087
00088 /** Construct a buffer accessor from a buffer using a command group
00089 handler object from the command group scope
00090
00091 Constructor only available for global_buffer or constant_buffer
00092 target.
00093
00094 access_target defines the form of access being obtained.
00095
00096 \todo Add template allocator type in all the accessor
00097 constructors in the specification or just use a more opaque
00098 Buffer type?
00099
00100 \todo fix specification where access mode should be target
00101 instead
00102 */
00103   template <typename Allocator>
00104   accessor(buffer<DataType, Dimensions, Allocator> &target_buffer,
00105            handler &command_group_handler) :  implementation_t {
00106     new detail::accessor<DataType, Dimensions, AccessMode, Target> {
00107       target_buffer.implementation->implementation, command_group_handler }
00108   } {
00109     static_assert(Target == access::target::global_buffer
00110                   || Target == access::target::constant_buffer,
00111                   "access target should be global_buffer or constant_buffer "
00112                   "when a handler is used");
00113     // Now the implementation is created, register it
00114     implementation->register_accessor();
00115   }
00116
00117
00118 /** Construct a buffer accessor from a buffer
00119
00120 Constructor only available for host_buffer target.
00121
00122 access_target defines the form of access being obtained.
00123 */
00124   template <typename Allocator>
00125   accessor(buffer<DataType, Dimensions, Allocator> &target_buffer)
00126     :  implementation_t {
00127     new detail::accessor<DataType, Dimensions, AccessMode, Target> {
00128       target_buffer.implementation->implementation }
00129   } {
00130     static_assert(Target == access::target::host_buffer,
00131                   "without a handler, access target should be host_buffer");
00132   }
00133
00134
00135 /** Construct a buffer accessor from a buffer given a specific range for
00136 access permissions and an offset that provides the starting point
00137 for the access range using a command group handler object from the
00138 command group scope
00139
00140 This accessor limits the processing of the buffer to the [offset,
00141 offset+range[ for every dimension.  Any other parts of the buffer
00142 will be unaffected.
00143
00144 Constructor only available for access modes global_buffer, and
00145 constant_buffer (see Table "Buffer accessor constructors").
```

```
00146 access_target defines the form of access being obtained.
00147
00148 This accessor is recommended for discard-write and discard read
00149 write access modes, when the unaffected parts of the processing
00150 should be retained.
00151 */
00152   template <typename Allocator>
00153   accessor(buffer<DataType, Dimensions, Allocator> &target_buffer,
00154            handler &command_group_handler,
00155            const range<Dimensions> &offset,
00156            const range<Dimensions> &range) {
00157     TRISYCL_UNIMPL;
00158   }
00159
00160
00161 /** Construct an accessor of dimension Dimensions with elements of type
00162 DataType using the passed range to specify the size in each
00163 dimension
00164
00165 It needs as a parameter a command group handler object from the
00166 command group scope.  Constructor only available if AccessMode is
00167 local, see Table 3.25.
00168 */
00169   accessor(const range<Dimensions> &allocation_size,
00170            handler &command_group_handler)
00171     :  implementation_t { new detail::accessor<DataType,
00172                                                Dimensions,
00173                                                AccessMode,
00174                                                access::target::local> {
00175       allocation_size, command_group_handler
00176         }
00177   }
00178   {
00179     static_assert(Target == access::target::local,
00180                   "This accessor constructor requires "
00181                   "access target be local");
00182   }
00183
00184
00185 /** Return a range object representing the size of the buffer in
00186 terms of number of elements in each dimension as passed to the
00187 constructor
00188
00189 \todo Move on
00190 https://cvs.khronos.org/bugzilla/show_bug.cgi?id=15564 and
00191 https://cvs.khronos.org/bugzilla/show_bug.cgi?id=14404
00192 */
00193   auto get_range()const {
00194     /* Interpret the shape which is a pointer to the first element as an
00195 array of Dimensions elements so that the range<Dimensions>
00196 constructor is happy with this collection
00197
00198 \todo Add also a constructor in range<> to accept a const
00199 std::size_t *?
00200 */
00201     return implementation->get_range();
00202   }
00203
00204
00205 /** Returns the total number of elements behind the accessor
00206
00207 Equal to get_range()[0] * ...  * get_range()[Dimensions-1].
00208
00209 \todo Move on
00210 https://cvs.khronos.org/bugzilla/show_bug.cgi?id=15564 and
00211 https://cvs.khronos.org/bugzilla/show_bug.cgi?id=14404
00212 */
00213   auto get_count()const {
00214     return implementation->get_count();
00215   }
00216
00217
00218 /** Returns the size of the underlying buffer storage in bytes
00219
00220 \todo It is incompatible with buffer get_size() in the spec
00221
00222 \todo Move on
00223 https://cvs.khronos.org/bugzilla/show_bug.cgi?id=15564 and
00224 https://cvs.khronos.org/bugzilla/show_bug.cgi?id=14404
00225 */
00226   auto get_size()const {
00227     return implementation->get_size();
00228   }
00229
00230 /** Use the accessor with integers à la [i1][i2][i3] or C++23 [i1, i2,...]
00231
00232 \return decltype(auto) to return either a reference to the final
```

```
00233 element when the indexing has been fully resolved or a proxy
00234 object to handle the remaining [] */
00235   template <std::integral...  T> decltype(auto) operator[](T...  indices) {
00236     return (*implementation)[indices...];
00237   }
00238
00239 /** Use the accessor with integers à la [i1][i2][i3] or C++23 [i1, i2,...]
00240
00241 \return decltype(auto) to return either a reference to the final
00242 element when the indexing has been fully resolved or a proxy
00243 object to handle the remaining [] */
00244   template <std::integral...  T> decltype(auto) operator[](T...  indices)const {
00245     return (*implementation)[indices...];
00246   }
00247
00248 /// To use the accessor with [id<>]
00249   auto& operator[](const id<dimensionality>& index) {
00250     return (*implementation)[index];
00251   }
00252
00253 /// To use the accessor with [id<>]
00254   auto& operator[](const id<dimensionality>& index)const {
00255     return (*implementation)[index];
00256   }
00257
00258
00259 /// To use an accessor with [item<>]
00260   auto& operator[](const item<dimensionality>& index) {
00261     return (*this)[index.get_id()];
00262   }
00263
00264
00265 /// To use an accessor with [item<>]
00266   auto& operator[](const item<dimensionality>& index)const {
00267     return (*this)[index.get_id()];
00268   }
00269
00270
00271 /** To use an accessor with an [nd_item<>]
00272
00273 \todo Add in the specification because used by HPC-GPU slide 22
00274 */
00275   auto& operator[](const nd_item<dimensionality>& index) {
00276     return (*this)[index.get_global_id()];
00277   }
00278
00279 /** To use an accessor with an [nd_item<>]
00280
00281 \todo Add in the specification because used by HPC-GPU slide 22
00282 */
00283   auto& operator[](const nd_item<dimensionality>& index)const {
00284     return (*this)[index.get_global_id()];
00285   }
00286
00287
00288 /** Get the first element of the accessor
00289
00290 Useful with an accessor on a scalar for example.
00291
00292 \todo Add in the specification
00293 */
00294   typename accessor_detail::reference operator*() {
00295     return **implementation;
00296   }
00297
00298
00299 /** Get the first element of the accessor
00300
00301 Useful with an accessor on a scalar for example.
00302
00303 \todo Add in the specification?
00304
00305 \todo Add the concept of 0-dim buffer and accessor for scalar
00306 and use an implicit conversion to value_type reference to access
00307 the value with the accessor?
00308 */
00309   typename accessor_detail::reference operator*()const {
00310     return **implementation;
00311   }
00312
00313
00314 /** Get the pointer to the start of the data
00315
00316 \todo Should it be named data() instead?  */
00317   auto
00318   get_pointer()const {
00319     return implementation->get_pointer();
```

```
00320   }
00321
00322 /** Forward all the iterator functions to the implementation
00323
00324 \todo Add these functions to the specification
00325
00326 \todo The fact that the lambda capture make a const copy of the
00327 accessor is not yet elegantly managed...  The issue is that
00328 begin()/end() dispatch is made according to the accessor
00329 constness and not from the array member constness...
00330
00331 \todo try to solve it by using some enable_if on array
00332 constness?
00333
00334 \todo The issue is that the end may not be known if it is
00335 implemented by a raw OpenCL cl_mem...  So only provide on the
00336 device the iterators related to the start?  Actually the accessor
00337 needs to know a part of the shape to have the multidimentional
00338 addressing.  So this only require a size_t more...
00339
00340 \todo Factor out these in a template helper
00341 */
00342
00343
00344   // iterator begin() { return array.begin(); }
00345   typename accessor_detail::iterator begin()const {
00346     return implementation->begin();
00347   }
00348
00349
00350   // iterator end() { return array.end(); }
00351   typename accessor_detail::iterator end()const {
00352     return implementation->end();
00353   }
00354
00355   typename accessor_detail::const_iterator cbegin()const {
00356     return implementation->cbegin();
00357   }
00358
00359
00360   typename accessor_detail::const_iterator cend()const {
00361     return implementation->cend();
00362   }
00363
00364
00365   typename accessor_detail::reverse_iterator rbegin()const {
00366     return implementation->rbegin();
00367   };
00368
00369
00370   typename accessor_detail::reverse_iterator rend()const {
00371     return implementation->rend();
00372   }
00373
00374
00375   typename accessor_detail::const_reverse_iterator crbegin()const {
00376     return implementation->rbegin();
00377   }
00378
00379
00380   typename accessor_detail::const_reverse_iterator crend()const {
00381     return implementation->rend();
00382   }
00383 };
00384
00385 /// @} End the execution Doxygen group
00386
00387
00388 /** \addtogroup old_data Data access and storage in old version of SYCL
00389 @{
00390 */
00391
00392 /** The pipe accessor abstracts the way pipe data are accessed inside
00393 a kernel
00394
00395 This is a proposal for the now abandoned SYCL 2.2 provisional specification.
00396 This is still here for historical reasons.
00397
00398 A specialization for an non-blocking pipe
00399 */
00400 template <typename DataType,
00401           access::mode AccessMode>
00402 class accessor<DataType, 1, AccessMode, access::target::pipe> :
00403     public detail::sycl_2_2::pipe_accessor<DataType,
00404                                            AccessMode,
00405                                            access::target::pipe> {
00406 public:
```

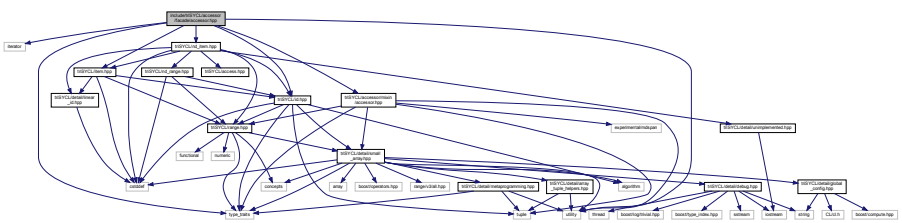
```
00407
00408   using accessor_detail =
00409     detail::sycl_2_2::pipe_accessor<DataType, AccessMode, access::target::pipe>;
00410   // Inherit of the constructors to have accessor constructor from detail
00411   using accessor_detail::accessor_detail;
00412
00413 /** Construct a pipe accessor from a pipe using a command group
00414 handler object from the command group scope
00415
00416 access_target defines the form of access being obtained.
00417 */
00418   accessor(sycl_2_2::pipe<DataType> &p, handler &command_group_handler)
00419     :  accessor_detail { p.implementation, command_group_handler } { }
00420
00421 /// Make a reservation inside the pipe
00422   sycl_2_2::pipe_reservation<accessor> reserve(std::size_t size)const {
00423     return accessor_detail::reserve(size);
00424   }
00425
00426
00427 /// Get the underlying pipe implementation
00428   auto &get_pipe_detail() {
00429     return accessor_detail::get_pipe_detail();
00430   }
00431
00432 };
00433
00434
00435 /** The pipe accessor abstracts the way pipe data are accessed inside
00436 a kernel
00437
00438 This is a proposal for the now abandoned SYCL 2.2 provisional specification.
00439 This is still here for historical reasons.
00440
00441 A specialization for a blocking pipe
00442 */
00443 template <typename DataType,
00444           access::mode AccessMode>
00445 class accessor<DataType, 1, AccessMode, access::target::blocking_pipe> :
00446     public detail::sycl_2_2::pipe_accessor<DataType,
00447                                            AccessMode,
00448                                            access::target::blocking_pipe> {
00449 public:
00450
00451   using accessor_detail =
00452     detail::sycl_2_2::pipe_accessor<DataType,
00453                                     AccessMode,
00454                                     access::target::blocking_pipe>;
00455   // Inherit of the constructors to have accessor constructor from detail
00456   using accessor_detail::accessor_detail;
00457
00458 /** Construct a pipe accessor from a pipe using a command group
00459 handler object from the command group scope
00460
00461 access_target defines the form of access being obtained.
00462 */
00463   accessor(sycl_2_2::pipe<DataType> &p, handler &command_group_handler)
00464     :  accessor_detail { p.implementation, command_group_handler } { }
00465
00466
00467 /** Construct a pipe accessor from a pipe outside of a normal
00468 kernel, for example in host code
00469 */
00470   accessor(sycl_2_2::pipe<DataType> &p)
00471     :  accessor_detail { p.implementation } { }
00472
00473
00474 /// Make a reservation inside the pipe
00475   sycl_2_2::pipe_reservation<accessor> reserve(std::size_t size)const {
00476     return accessor_detail::reserve(size);
00477   }
00478
00479
00480 /// Get the underlying pipe implementation
00481   auto &get_pipe_detail() {
00482     return accessor_detail::get_pipe_detail();
00483   }
00484
00485 };
00486
00487 /** SYCL 2020 host_accessor
00488
00489 \param[in] DataType is the value type of the elements
00490
00491 \param[in] Dimensions is the rank of the accessor
00492
00493 \param[in] AccessMode precises whether the access is done in read
```

```
00494 or write mode.  By default this is read+write
00495 */
00496 template <typename DataType,
00497           int Dimensions,
00498           access::mode AccessMode = access::mode::read_write>
00499 class host_accessor :  public accessor<DataType,
00500                                        Dimensions,
00501                                        AccessMode,
00502                                        access::target::host_buffer> {
00503   using base = accessor<DataType,
00504                         Dimensions,
00505                         AccessMode,
00506                         access::target::host_buffer>;
00507  public:
00508   using typename base::accessor;
00509
00510 /// Create an accessor to a buffer to access from the host
00511   template <typename Allocator>
00512   host_accessor(buffer<DataType, Dimensions, Allocator>& target_buffer)
00513     :  base { target_buffer } {}
00514 };
00515
00516 /** Top-level function to break circular dependencies on the the types
00517 to get the pipe implementation */
00518 template <typename Accessor>
00519 static inline auto &get_pipe_detail(Accessor &a) {
00520   return a.get_pipe_detail();
00521 }
00522
00523 /// @} End the data Doxygen group
00524
00525 }
00526
00527 /*
00528 # Some Emacs stuff:
00529 ### Local Variables:
00530 ### ispell-local-dictionary:  "american"
00531 ### eval:  (flyspell-prog-mode)
00532 ### End:
00533 */
00534
00535 #endif // TRISYCL_SYCL_ACCESSOR_HPP
```

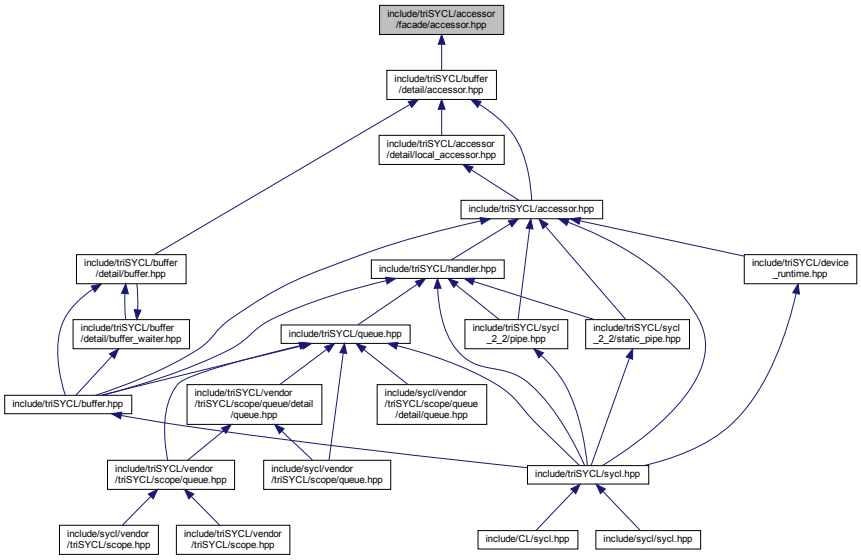
13.245 include/triSYCL/accessor/facade/accessor.hpp File Reference

```
#include <iterator>
#include <type_traits>
#include <utility>
#include "triSYCL/accessor/mixin/accessor.hpp"
#include "triSYCL/id.hpp"
#include "triSYCL/item.hpp"
#include "triSYCL/nd_item.hpp"
```

Include dependency graph for accessor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::facade::accessor< AccessorMixIn >

  SYCL accessor facade providing multi-dimensional access features. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::facade

13.246 accessor.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_ACCESSOR_FACADE_ACCESSOR_HPP
00002 #define TRISYCL_SYCL_ACCESSOR_FACADE_ACCESSOR_HPP
00003
00004 /** \file A SYCL accessor façade to expose to the end user the
00005 expected accessor interface on top of low-level SYCL accessor
00006 concepts
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <iterator>
00013 #include <type_traits>
00014 #include <utility>
00015
00016 #include "triSYCL/accessor/mixin/accessor.hpp"
00017 #include "triSYCL/id.hpp"
00018 #include "triSYCL/item.hpp"
00019 #include "triSYCL/nd_item.hpp"
00020
00021 namespace trisycl::facade {
00022
```

```
00023 /** \addtogroup execution Platforms, contexts, accessors and queues
00024 @{
00025 */
00026
00027 /// SYCL accessor facade providing multi-dimensional access features
00028 template <typename AccessorMixIn> class accessor :  public AccessorMixIn {
00029
00030   using mixin = AccessorMixIn;
00031
00032  public:
00033   using iterator = typename mixin::pointer;
00034   using const_iterator = typename mixin::const_pointer;
00035   using reverse_iterator = typename std::reverse_iterator<iterator>;
00036   using const_reverse_iterator = typename std::reverse_iterator<const_iterator>;
00037
00038 /** Use the accessor with integers à la [i1][i2][i3] or C++23 [i1, i2,...]
00039
00040 \return decltype(auto) to return either a reference to the final
00041 element when the indexing has been fully resolved or a proxy
00042 object to handle the remaining [] */
00043   template <std::integral...  T> decltype(auto) operator[](T...  indices) {
00044     if constexpr (sizeof...(T) == 1)
00045       /* The [][][] case uses a proxy object to track all the [index]
00046 and aggregate them to resolve the indexing */
00047       return
00048           typename mixin::template track_index<1> { mixin::access }[indices...];
00049     else
00050       // Or just the C++23 [i1, i2,...]  case
00051       return mixin::access[indices...];
00052   }
00053
00054 /** Use the accessor with integers à la [i1][i2][i3] or C++23 [i1, i2,...]
00055
00056 \return decltype(auto) to return either a reference to the final
00057 element when the indexing has been fully resolved or a proxy
00058 object to handle the remaining [] */
00059   template <std::integral...  T> decltype(auto) operator[](T...  indices)const {
00060     if constexpr (sizeof...(T) == 1)
00061       /* The [][][] case uses a proxy object to track all the [index]
00062 and aggregate them to resolve the indexing */
00063       return
00064           typename mixin::template track_index<1> { mixin::access }[indices...];
00065     else
00066       // Or just the C++23 [i1, i2,...]  case
00067       return mixin::access[indices...];
00068   }
00069
00070 /// To use the accessor with [id<>]
00071   decltype(auto) operator[](const id<mixin::rank()>& index) {
00072     return mixin::tuple_indexed_access(index);
00073   }
00074
00075 /// To use the accessor with [id<>]
00076   decltype(auto) operator[](const id<mixin::rank()>& index) const {
00077     return mixin::tuple_indexed_access(index);
00078   }
00079
00080 /// To use an accessor with [item<>]
00081   decltype(auto) operator[](const item<mixin::rank()>& index) {
00082     return (*this)[index.get()];
00083   }
00084
00085 /// To use an accessor with [item<>]
00086   decltype(auto) operator[](const item<mixin::rank()>& index) const {
00087     return (*this)[index.get()];
00088   }
00089
00090 /** To use an accessor with an [nd_item<>]
00091
00092 \todo Add to the specification because used by HPC-GPU slide 22
00093 */
00094   decltype(auto) operator[](const nd_item<mixin::rank()>& index) {
00095     return (*this)[index.get_global()];
00096   }
00097
00098 /** To use an accessor with an [nd_item<>]
00099
00100 \todo Add to the specification because used by HPC-GPU slide 22
00101 */
00102   decltype(auto) operator[](const nd_item<mixin::rank()>& index) const {
00103     return (*this)[index.get_global()];
00104   }
00105
00106 /** Get the first element of the accessor
00107
00108 Useful with an accessor on a scalar for example.
00109
```

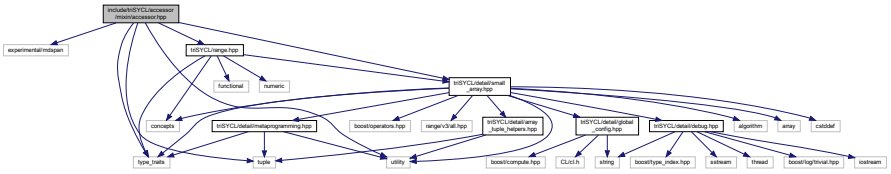
```
00110 \todo Add to the specification
00111 */
00112   typename mixin::reference operator*() { return *mixin::data(); }
00113
00114 /** Get the first element of the accessor
00115
00116 Useful with an accessor on a scalar for example.
00117
00118 \todo Add to the specification?
00119
00120 \todo Add the concept of 0-dim buffer and accessor for scalar
00121 and use an implicit conversion to value_type reference to access
00122 the value with the accessor?
00123 */
00124   typename mixin::reference operator*()const { return *mixin::data(); }
00125
00126 /** Return the pointer to the data
00127
00128 \todo Implement the various pointer address spaces
00129 */
00130   auto get_pointer() { return mixin::data(); }
00131
00132 /** Forward all the iterator functions to the implementation
00133
00134 \todo Add these functions to the specification
00135
00136 \todo The fact that the lambda capture make a const copy of the
00137 accessor is not yet elegantly managed...  The issue is that
00138 begin()/end() dispatch is made according to the accessor
00139 constness and not from the array member constness...
00140
00141 \todo try to solve it by using some enable_if on array
00142 constness?
00143
00144 \todo The issue is that the end may not be known if it is
00145 implemented by a raw OpenCL cl_mem...  So only provide on the
00146 device the iterators related to the start?  Actually the accessor
00147 needs to know a part of the shape to have the multidimentional
00148 addressing.  So this only require a size_t more...
00149
00150 \todo Factor out these in a template helper
00151
00152 \todo Do we need this in detail::accessor too or only in accessor?
00153 */
00154
00155   iterator begin() { return mixin::data(); }
00156
00157   iterator end() { return mixin::data() + mixin::get_count(); }
00158
00159   const_iterator cbegin() { return mixin::data(); }
00160
00161   const_iterator cend() { return mixin::data() + mixin::get_count(); }
00162
00163   reverse_iterator rbegin() { return std::reverse_iterator(end()); }
00164
00165   reverse_iterator rend() {
00166     return std::reverse_iterator(begin());
00167     ;
00168   }
00169
00170   const_reverse_iterator crbegin()const {
00171     return std::reverse_iterator(cend());
00172   }
00173
00174   const_reverse_iterator crend()const {
00175     return std::reverse_iterator(cbegin());
00176   }
00177 };
00178
00179 /// @} to end the Doxygen group
00180
00181 } // namespace trisycl::facade
00182
00183 #endif // TRISYCL_SYCL_ACCESSOR_FACADE_ACCESSOR_HPP
```

13.247 include/triSYCL/accessor/mixin/accessor.hpp File Reference

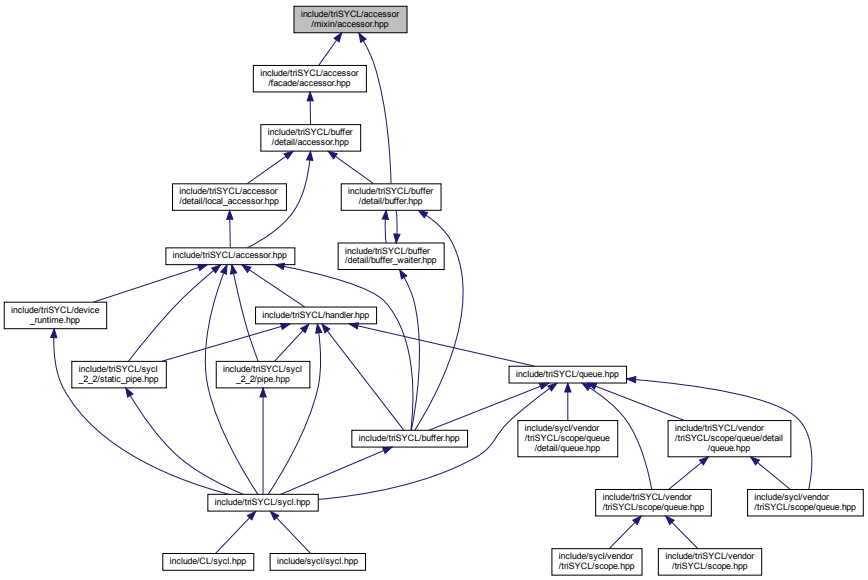
```
#include <experimental/mdspan>
#include <tuple>
#include <type_traits>
```

```
#include <utility>
#include "triSYCL/range.hpp"
#include "triSYCL/detail/small_array.hpp"
```

Include dependency graph for accessor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::mixin::accessor< T, Dimensions >

  SYCL accessor mixin providing multi-dimensional access features. More...
- struct trisycl::mixin::accessor< T, Dimensions >::track_index< N >

  Proxy object to transform an expression like accessor[i1][i2][i3] into the implementation mdpsan(i1,i2,i3) one index at a time.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::mixin

13.248 accessor.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_ACCESSOR_MIXIN_ACCESSOR_HPP
00002 #define TRISYCL_SYCL_ACCESSOR_MIXIN_ACCESSOR_HPP
00003
00004 /** \file A SYCL accessor mixin to implement more easily a low-level
00005 SYCL accessor concept on top of some concrete storage
00006
00007 This file is distributed under the University of Illinois Open Source
00008 License.  See LICENSE.TXT for details.
00009 */
00010
00011 #include <experimental/mdspan>
00012 #include <tuple>
00013 #include <type_traits>
00014 #include <utility>
00015
00016 #include "triSYCL/range.hpp"
00017 #include "triSYCL/detail/small_array.hpp"
00018
00019 namespace trisycl::mixin {
00020
00021 /** \addtogroup execution Platforms, contexts, accessors and queues
00022 @{
00023 */
00024
00025 /// SYCL accessor mixin providing multi-dimensional access features
00026 template <typename T, int Dimensions> class accessor {
00027
00028  public:
00029 /** Extension to SYCL: provide pieces of STL container interface
00030 from mdspan */
00031   using element_type = T;
00032 /** Even if the buffer is read-only use a non-const type so at least
00033 the current implementation can copy the data too */
00034   using value_type = std::remove_cv_t<element_type>;
00035
00036   // Useful to handle buffers initialized from const values
00037   using non_const_pointer = value_type*;
00038
00039 /** Get the number of dimensions of the buffer
00040
00041 Name inspired from ISO C++ P0009 mdspan papers
00042 */
00043   static auto constexpr rank() { return Dimensions; }
00044
00045  protected:
00046 /// The memory lay-out of a buffer is a dynamic multidimensional array
00047   using mdspan =
00048       std::mdspan<element_type, std::dextents<std::size_t, Dimensions»;
00049
00050 /** This is the multi-dimensional interface to the data that may point
00051 to either allocation in the case of storage managed by SYCL itself
00052 or to some other memory location in the case of host memory or
00053 storage<> abstraction use
00054 */
00055   mdspan access;
00056
00057 /** Cast a SYCL range/id-like into a mdspan index array, which is an
00058 array of std::size_t into an array of std::ptrdiff_t
00059 */
00060   template <typename BasicType, typename FinalType>
00061   const std::array<typename mdspan::size_type, rank()>&
00062   extents_cast(const detail::small_array<BasicType, FinalType, rank()>& sa) {
00063     return reinterpret_cast<
00064         const std::array<typename mdspan::size_type, rank()>&>(sa);
00065   }
00066
00067 /// Set later the mdspan associated to this accessor
00068   void set_access(const mdspan& a) { access = a; }
00069
00070  public:
00071 /// Pointer type to element
00072   using pointer = typename mdspan::accessor_type::data_handle_type;
00073
00074 /// Pointer type to const element
00075   using const_pointer = const element_type*;
00076
00077 /// Reference type to the elements
00078   using reference = typename mdspan::reference;
00079
00080 /// Used by the local accessor hack on top of host accessor
00081   accessor() = default;
00082
```

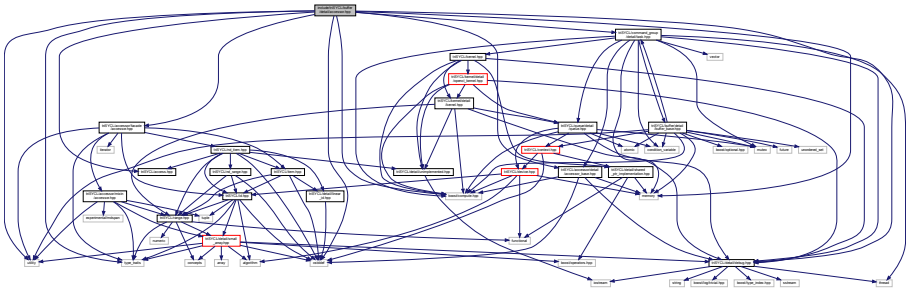
```
00083 /// Create an accessor of dimensions r on top of data storage
00084   accessor(pointer data, const range<rank()>& r)
00085       :  access { data, extents_cast(r) } {}
00086
00087 /// Create an accessor from another mdspan
00088   accessor(const mdspan& m)
00089       :  access { m } {}
00090
00091 /// Update the accessor to target somewhere else
00092   void update(pointer data, const range<rank()>& r) {
00093     access = mdspan { data, extents_cast(r) };
00094   }
00095
00096 /** Return a range object representing the size of the buffer in
00097 terms of number of elements in each dimension as passed to the
00098 constructor
00099
00100 \todo Cache it since it is const?
00101 */
00102   auto get_range()const {
00103     range<Dimensions> r;
00104     for (std::size_t i = 0; i < Dimensions; ++i)
00105       r[i] = access.extent(i);
00106     return r;
00107   }
00108
00109 /** Returns the total number of elements in the buffer
00110
00111 Equal to get_range()[0] * ...  * get_range()[Dimensions-1].
00112
00113 \todo Move these kinds of functions into a mixin between buffers
00114 and accessors?
00115
00116 \todo Cache it since it is const?
00117 */
00118   std::size_t get_count()const {
00119     return access.mapping().required_span_size();
00120   }
00121
00122 /** Returns the size of the buffer storage in bytes
00123
00124 \todo rename to something else.  In
00125 http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/p0122r0.pdf
00126 it is named bytes() for example
00127
00128 \todo Cache it since it is const?
00129 */
00130   std::size_t get_size()const { return get_count() * sizeof(value_type); }
00131
00132 /// Get the underlying storage
00133   auto data() { return access.data_handle(); }
00134
00135 /** Access to an mdspan element with indices implementing a tuple
00136 interface
00137
00138 \param[inout] some_mdspan is the mdspan to access
00139
00140 \param[in] tuple_like_indices are the indices to use
00141
00142 \return a reference to the element
00143 */
00144   static decltype(auto)
00145   tuple_indexed_mdspan_access(auto&& some_mdspan,
00146                               const auto& tuple_like_indices) {
00147     // Otherwise use the C++23 mdspan[i1, i2,...]  indexing syntax
00148     return std::invoke(
00149         [&](auto...  i) -> decltype(auto) { return some_mdspan[i...]; },
00150         tuple_like_indices);
00151   }
00152
00153 /** Access to an element with indices implementing a tuple
00154 interface
00155
00156 \param[in] tuple_like_indices are the indices to use
00157
00158 \return a reference to the accessor element
00159 */
00160   decltype(auto) tuple_indexed_access(const auto& tuple_like_indices) {
00161     return tuple_indexed_mdspan_access(access, tuple_like_indices);
00162   }
00163
00164 /** Proxy object to transform an expression like
00165 accessor[i1][i2][i3] into the implementation mdpsan(i1,i2,i3)
00166 one index at a time.
00167
00168 It gathers intermediate [index] to finally call the mdspan
00169 indexing operator once they are all available
```

```
00170
00171 \parameter[in] N is the number of indices which can be stored in
00172 this proxy
00173 */
00174   template <std::size_t N> struct track_index {
00175 /// Keep a reference to the mdspan to eventually resolve the
00176 /// indexing
00177     mdspan& mds;
00178
00179 /// The list of indices in the order of [i1][i2][i3]...
00180     std::array<std::size_t, N> indices;
00181
00182 /// Construct the initial tracking object from the accessor
00183     track_index(accessor& a)
00184         :  mds { a.access } {}
00185
00186 /// Create a tracking object from an mdspan and a list of indices
00187     template <typename...  Index>
00188     track_index(mdspan& m, Index&&...  inds)
00189         :  mds { m }
00190         // Typically there will be N - 1 indices in the array of size N
00191         , indices { std::forward<Index>(inds)...  } {}
00192
00193 /// The individual indexing operator
00194     decltype(auto) operator[](std::size_t index) {
00195       // Keep track of the new index in the last element
00196       indices[N - 1] = index;
00197       if constexpr (N == accessor::rank())
00198         // If we have accumulated all the indices, call the mdspan
00199         // indexing function
00200         return tuple_indexed_mdspan_access(mds, indices);
00201       else
00202         // Otherwise return a tracker from the current indices and
00203         // with 1 more room for the next index
00204         return std::make_from_tuple<track_index<N + 1»(
00205             std::tuple_cat(std::forward_as_tuple(mds), indices));
00206     }
00207   };
00208 };
00209
00210 /// @} to end the Doxygen group
00211
00212 } // namespace trisycl::mixin
00213
00214 #endif // TRISYCL_SYCL_ACCESSOR_MIXIN_ACCESSOR_HPP
```

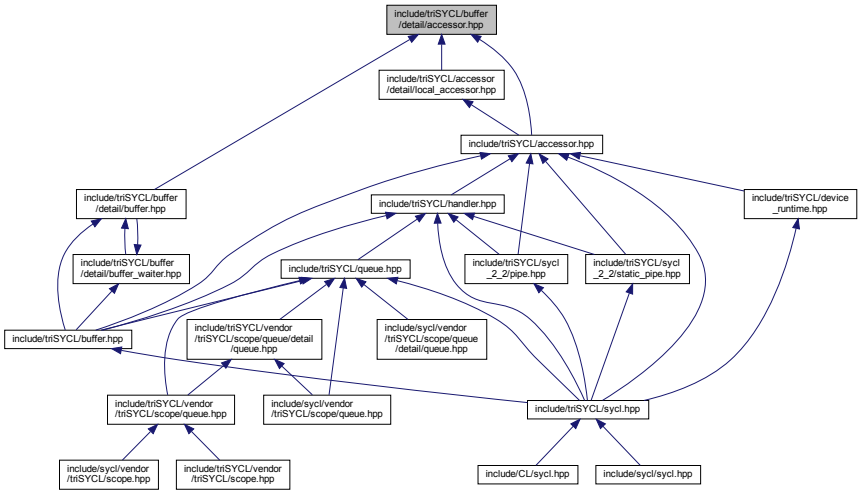
13.249 include/triSYCL/buffer/detail/accessor.hpp File Reference

```
#include <cstddef>
#include <memory>
#include <type_traits>
#include <utility>
#include <boost/compute.hpp>
#include "triSYCL/access.hpp"
#include "triSYCL/accessor/detail/accessor_base.hpp"
#include "triSYCL/accessor/facade/accessor.hpp"
#include "triSYCL/command_group/detail/task.hpp"
#include "triSYCL/detail/debug.hpp"
```

Include dependency graph for accessor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::accessor< T, Dimensions, Mode, Target >

  The buffer accessor abstracts the way buffer data are accessed inside a kernel in a multidimensional variable length array way. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.250 accessor.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_BUFFER_DETAIL_ACCESSOR_HPP
00002 #define TRISYCL_SYCL_BUFFER_DETAIL_ACCESSOR_HPP
00003
00004 /** \file The SYCL buffer accessor<> detail behind the scene
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <cstddef>
00013 #include <memory>
00014 #include <type_traits>
00015 #include <utility>
00016
00017 #ifdef TRISYCL_OPENCL
00018 #include <boost/compute.hpp>
00019 #endif
00020
00021 #include "triSYCL/access.hpp"
00022 #include "triSYCL/accessor/detail/accessor_base.hpp"
```

```
00023 #include "triSYCL/accessor/facade/accessor.hpp"
00024 #include "triSYCL/command_group/detail/task.hpp"
00025 #include "triSYCL/detail/debug.hpp"
00026
00027 namespace trisycl {
00028
00029 class handler;
00030
00031 namespace detail {
00032
00033 // Forward declaration of detail::buffer for use in accessor
00034 template <typename T, int Dimensions> class buffer;
00035
00036 /** \addtogroup data Data access and storage in SYCL
00037 @{
00038 */
00039
00040 /** The buffer accessor abstracts the way buffer data are accessed
00041 inside a kernel in a multidimensional variable length array way.
00042
00043 This implementation relies on std::mdspan to provide this nice
00044 syntax and behavior.
00045
00046 Right now the aim of this class is just to access to the buffer in
00047 a read-write mode, even if capturing the accessor from a lambda
00048 make it const (since in examples we have lambda with [=] without
00049 mutable lambda).
00050
00051 \todo Use the access::mode
00052 */
00053 template <typename T, int Dimensions, access::mode Mode,
00054           access::target Target /* = access::global_buffer */>
00055 class accessor
00056     :  public detail::accessor_base
00057     , public facade::accessor<mixin::accessor<T, Dimensions>»
00058     , public std::enable_shared_from_this<accessor<T, Dimensions, Mode, Target>»
00059     , public detail::debug<accessor<T, Dimensions, Mode, Target>» {
00060 /** Keep a reference to the accessed buffer
00061
00062 Beware that it owns the buffer, which means that the accessor
00063 has to be destroyed to release the buffer and potentially
00064 unblock a kernel at the end of its execution
00065 */
00066   std::shared_ptr<detail::buffer<T, Dimensions>» buf;
00067
00068 /// Where most of the user-facing interface dwells
00069   using facade = facade::accessor<mixin::accessor<T, Dimensions>>;
00070
00071  public:
00072 #ifdef TODO
00073 /** Inherit the iterator types from the implementation
00074
00075 \todo Add iterators to accessors in the specification
00076 */
00077   using const_iterator = typename array_view_type::const_iterator;
00078   using reverse_iterator = typename array_view_type::reverse_iterator;
00079   using const_reverse_iterator =
00080       typename array_view_type::const_reverse_iterator;
00081 #endif
00082
00083 /// Used by the local accessor hack on top of host accessor
00084   accessor() = default;
00085
00086 /** Construct a host accessor from an existing buffer
00087
00088 \todo fix the specification to rename target that shadows
00089 template parm
00090 */
00091   accessor(std::shared_ptr<detail::buffer<T, Dimensions>> target_buffer)
00092       :  facade { target_buffer->access }
00093       , buf { target_buffer } {
00094     target_buffer->template track_access_mode<Mode>();
00095     TRISYCL_DUMP_T("Create a host accessor write = " « is_write_access());
00096     static_assert(Target == access::target::host_buffer,
00097                   "without a handler, access target should be host_buffer");
00098     /* The host needs to wait for all the producers of the buffer to
00099 have finished */
00100     buf->wait();
00101
00102 #ifdef TRISYCL_OPENCL
00103     /* For the host context, we are obligated to update the buffer state
00104 during the accessors creation, otherwise we have no way of knowing
00105 if a buffer was modified on the host.  This is only true because
00106 host accessors are blocking
00107 */
00108     trisycl::context ctx;
00109     buf->update_buffer_state(ctx, Mode, facade::get_size(), facade::data());
```

```
00110 #endif
00111   }
00112
00113 /** Construct a device accessor from an existing buffer
00114
00115 \todo fix the specification to rename target that shadows
00116 template parm
00117 */
00118   accessor(std::shared_ptr<detail::buffer<T, Dimensions>> target_buffer,
00119            handler& command_group_handler)
00120       :  facade { target_buffer->access }
00121       , buf { target_buffer } {
00122     target_buffer->template track_access_mode<Mode>();
00123     TRISYCL_DUMP_T("Create a kernel accessor write = " « is_write_access());
00124     static_assert(Target == access::target::global_buffer ||
00125                           Target == access::target::constant_buffer,
00126                   "access target should be global_buffer or constant_buffer "
00127                   "when a handler is used");
00128     // Register the buffer to the task dependencies
00129     task = buffer_add_to_task(buf, &command_group_handler, is_write_access());
00130   }
00131
00132 /** Register the accessor once a \c std::shared_ptr is created on it
00133
00134 This is to be called from outside once the object is created.  It
00135 has been tried directly inside the contructor, but calling \c
00136 shared_from_this() from the constructor dead-lock with
00137 libstdc++6
00138
00139 \todo Double-check with the C++ committee on this issue.
00140 */
00141   void register_accessor() {
00142     if (!task->get_queue()->is_host()) {
00143       // To keep alive this accessor in the following lambdas
00144       auto acc = this->shared_from_this();
00145       // Attach the accessor to the task and get its order
00146       set_order(task->register_accessor(acc));
00147 #ifdef TRISYCL_OPENCL
00148       /* Before running the kernel, make sure the cl_mem behind this
00149 accessor is up-to-date on the device if needed and pass it to
00150 the kernel */
00151       task->add_prelude([=] { acc->copy_in_cl_buffer(); });
00152       // After running the kernel, deal with some copy-back if needed
00153       task->add_postlude([=] {
00154         /* Even if this function does nothing, it is required to
00155 have the capture of acc to keep the accessor alive across
00156 the kernel execution up to the execution postlude */
00157         acc->copy_back_cl_buffer();
00158       });
00159 #endif
00160     }
00161   }
00162
00163 /// Get the buffer used to create the accessor
00164   detail::buffer<T, Dimensions>& get_buffer() { return *buf; }
00165
00166 /** Test if the accessor has a read access right
00167
00168 \todo Strangely, it is not really constexpr because it is not a
00169 static method...
00170
00171 \todo to move in the access::mode enum class and add to the
00172 specification ?
00173 */
00174   constexpr bool is_read_access()const {
00175     return Mode == access::mode::read || Mode == access::mode::read_write ||
00176            Mode == access::mode::discard_read_write;
00177   }
00178
00179 /** Test if the accessor has a write access right
00180
00181 \todo Strangely, it is not really constexpr because it is not a
00182 static method...
00183
00184 \todo to move in the access::mode enum class and add to the
00185 specification ?
00186 */
00187   constexpr bool is_write_access()const {
00188     return Mode == access::mode::write || Mode == access::mode::read_write ||
00189            Mode == access::mode::discard_write ||
00190            Mode == access::mode::discard_read_write;
00191   }
00192
00193  private:
00194 #ifdef TRISYCL_OPENCL
00195   // The following function are used from handler
00196   friend handler;
```

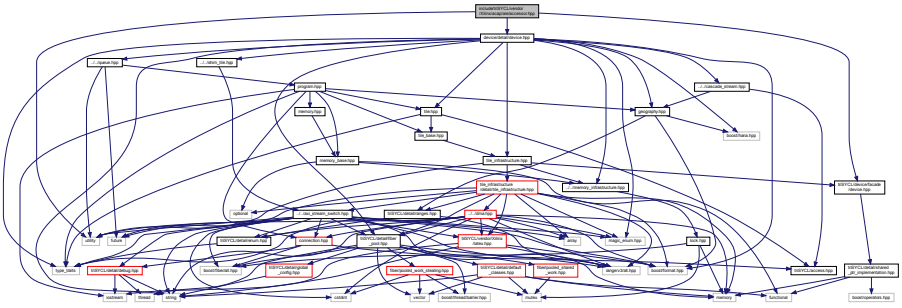
```
00197
00198 /// Get the boost::compute::buffer or throw if unset
00199   boost::compute::buffer get_cl_buffer()const override {
00200     // This throws if not set
00201     auto ctx = task->get_queue()->get_context();
00202     return buf->get_cl_buffer(ctx);
00203   }
00204
00205 /** Lazily associate a CL buffer to the SYCL buffer and copy data in it
00206 if required, updates the state of the data in the buffer across contexts
00207 */
00208   void copy_in_cl_buffer() {
00209     /* Create the OpenCL buffer and copy in it the data from the host if
00210 the buffer doesn't already exists or if the data is not up to date
00211 */
00212     auto ctx = task->get_queue()->get_context();
00213     buf->update_buffer_state(ctx, Mode, facade::get_size(), facade::data());
00214   }
00215
00216 /// Does nothing
00217   void copy_back_cl_buffer() {
00218     /* The copy back is handled by the host accessor and the buffer destructor.
00219 We don't need to systematically transfer the data after the
00220 kernel execution
00221
00222 \todo Figure out what to do with this function
00223 */
00224   }
00225 #endif
00226
00227 protected:
00228 /// Set later the current buffer associated to this accessor
00229   void set_buffer(std::shared_ptr<detail::buffer<T, Dimensions>> b) {
00230     buf = b;
00231   }
00232 };
00233
00234 /// @} End the data Doxygen group
00235
00236 } // namespace detail
00237 } // namespace trisycl
00238
00239 /*
00240 # Some Emacs stuff:
00241 ### Local Variables:
00242 ### ispell-local-dictionary:  "american"
00243 ### eval:  (flyspell-prog-mode)
00244 ### End:
00245 */
00246
00247 #endif // TRISYCL_SYCL_BUFFER_DETAIL_ACCESSOR_HPP
```

13.251 include/triSYCL/vendor/Xilinx/acap/aie/accessor.hpp File Reference

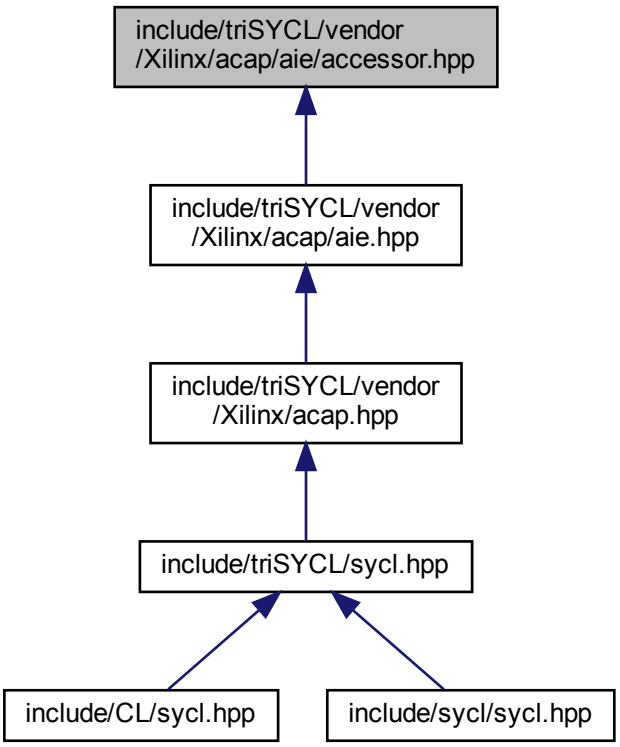
SYCL accessor for AIE tile device.

```
#include <utility>
#include "device/detail/device.hpp"
#include "triSYCL/device/facade/device.hpp"
```

Include dependency graph for accessor.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::accessor< Buffer >

  Create a SYCL-like device view of an AI Engine CGRA with some layout.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.251.1 Detailed Description

SYCL accessor for AIE tile device.

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file accessor.hpp.

13.252 accessor.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ACCESSOR_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ACCESSOR_HPP
00003
00004 /** \file
00005
00006 SYCL accessor for AIE tile device
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <utility>
00013
00014 #include "device/detail/device.hpp"
00015 #include "triSYCL/device/facade/device.hpp"
00016
00017 /// \ingroup aie
00018 /// @{
00019
00020 namespace trisycl::vendor::xilinx::acap::aie {
00021
00022 /** Create a SYCL-like device view of an AI Engine CGRA with some layout
00023
00024 \param Layout is the layout description of the machine to
00025 instantiate with the physical size
00026 */
00027 template <typename Buffer>
00028 class accessor :  public ::trisycl::accessor<typename Buffer::value_type,
00029                                               Buffer::rank(),
00030                                               access::mode::read_write,
00031                                               access::target::host_buffer> {
00032
00033   using base = ::trisycl::accessor<typename Buffer::value_type,
00034                                    Buffer::rank(),
00035                                    access::mode::read_write,
00036                                    access::target::host_buffer>;
00037  public:
00038
00039   using typename base::accessor;
00040   accessor(Buffer& b, auto& cgh)
00041     :  base { b } {}
00042 };
00043
00044 /// @} End the aie Doxygen group
00045
00046 } // namespace trisycl::vendor::xilinx::acap::aie
00047
00048 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ACCESSOR_HPP
```

13.253 include/sycl/vendor/Xilinx/acap/aie/axi_stream_switch.hpp File Reference

Model of an AXI stream switch.

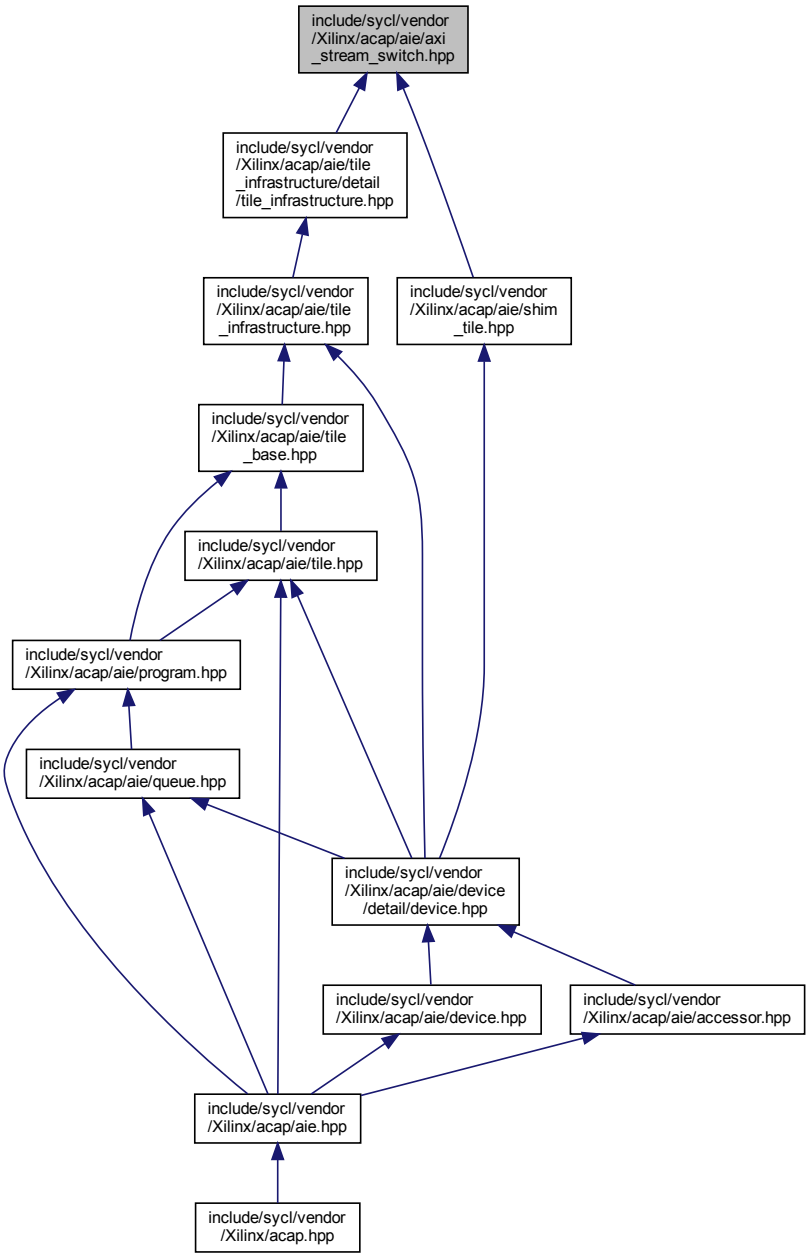
```
#include <array>
#include <cstdint>
#include <future>
#include <memory>
#include <string>
#include <type_traits>
#include <utility>
#include <boost/fiber/all.hpp>
#include <boost/format.hpp>
#include "magic_enum.hpp"
#include <range/v3/all.hpp>
#include "connection.hpp"
#include "triSYCL/detail/debug.hpp"
```

```
#include "triSYCL/detail/enum.hpp"
#include "triSYCL/detail/fiber_pool.hpp"
#include "triSYCL/vendor/Xilinx/latex.hpp"
```

Include dependency graph for axi_stream_switch.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >

  An AXI stream switch parameterized with some geographic layout. More...
- class trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion

  A router input port with routing skills.
- struct trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input_port
- struct trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::output_port

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.253.1 Detailed Description

Model of an AXI stream switch.

Based on Math Engine (ME) Architecture Specification, Revision v2.01 March 2019:

- section "4.2 AXI-Stream Tile Interconnect"
- section "6.5.3 Shim AXI-Stream Interconnect"

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file axi_stream_switch.hpp.

13.254 axi_stream_switch.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_AXI_STREAM_SWITCH_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_AXI_STREAM_SWITCH_HPP
00003
00004 /** \file
00005
00006 Model of an AXI stream switch
00007
00008 Based on Math Engine (ME) Architecture Specification, Revision v2.01
00009 March 2019:
00010
00011 - section "4.2 AXI-Stream Tile Interconnect"
00012
00013 - section "6.5.3 Shim AXI-Stream Interconnect"
00014
00015 Ronan dot Keryell at Xilinx dot com
00016
00017 This file is distributed under the University of Illinois Open Source
00018 License.  See LICENSE.TXT for details.
00019 */
00020
```

```
00021 #include <array>
00022 #include <cstdint>
00023 #include <future>
00024 #include <memory>
00025 #include <string>
00026 #include <type_traits>
00027 #include <utility>
00028
00029 #include <boost/fiber/all.hpp>
00030 #include <boost/format.hpp>
00031 #include "magic_enum.hpp"
00032 #include <range/v3/all.hpp>
00033
00034 #include "connection.hpp"
00035 #include "triSYCL/detail/debug.hpp"
00036 #include "triSYCL/detail/enum.hpp"
00037 #include "triSYCL/detail/fiber_pool.hpp"
00038 #include "triSYCL/vendor/Xilinx/latex.hpp"
00039
00040 namespace trisycl::vendor::xilinx::acap::aie {
00041
00042 /// \ingroup aie
00043 /// @{
00044
00045 /** An AXI stream switch parameterized with some geographic layout
00046
00047 \param[in] AxiStreamGeography is the geography class used to
00048 instantiate the AXI stream switch.
00049 */
00050 template <typename AXIStreamGeography>
00051 class axi_stream_switch
00052   :  ::trisycl::detail::debug<axi_stream_switch <AXIStreamGeography» {
00053
00054 public:
00055
00056   using axi_ss_geo = AXIStreamGeography;
00057   using mpl = typename axi_ss_geo::master_port_layout;
00058   using spl = typename axi_ss_geo::slave_port_layout;
00059
00060 /// A router input port with routing skills
00061   class router_minion :  public communicator_port {
00062 public  :
00063 /// Router ingress capacity queue
00064     const int capacity;
00065
00066 /// Ingress packet queue
00067     boost::fibers::buffered_channel<axi_packet> c
00068       { static_cast<std::size_t>(capacity) };
00069
00070 /// The ids of the outputs port the router minion has to forward to.
00071 /// Used by introspection to track current routing configuration
00072     std::vector<axi_stream_switch::mpl> mpl_outputs;
00073
00074 /** The outputs the router minion has to forward to.
00075 This what is used to speed up the routing itself even if the id
00076 could be enough */
00077     std::vector<std::shared_ptr<communicator_port» outputs;
00078
00079 /// To shepherd the routing fibers
00080     std::vector<::trisycl::detail::fiber_pool::future<void» futures;
00081
00082 /// Keep track of the AXI stream switch owning this port for debugging
00083     axi_stream_switch &axi_ss;
00084
00085 /// Wait for the fibers to complete
00086     void join() {
00087       // Wait on all the fibers to finish
00088       for (auto &f :  futures)
00089         // Get the value of the future, to get an exception if any
00090         f.get();
00091     }
00092
00093   public:
00094
00095     using value_type = axi_packet::value_type;
00096
00097 /// Enqueue a packet on the router input
00098 /// \todo Clean up the API to separate data from signaling
00099     void write(const axi_packet &v)override {
00100 /// \todo separate debug from shutdown case
00101       TRISYCL_DUMP_T("router_minion " « this
00102                      « " on tile(" « router_minion::axi_ss.x_coordinate « ',’
00103                      « router_minion::axi_ss.y_coordinate
00104                      « ") on fiber " « boost::this_fiber::get_id()
00105                      « " write data value " « v.data
00106                      « " to buffered_channel " « &c);
00107       c.push(v);
```

```
00108     }
00109
00110
00111 /** Try to enqueue a packet on the router input
00112
00113 \return true if the packet is correctly enqueued
00114 */
00115     bool try_write(const axi_packet &v)override {
00116       return c.try_push(v) != boost::fibers::channel_op_status::full;
00117     }
00118
00119
00120 /// Waiting read to a core input port
00121     value_type read()override {
00122       TRISYCL_DUMP_T("router_minion " « this « " on tile("
00123                      « router_minion::axi_ss.x_coordinate
00124                      « ',' « router_minion::axi_ss.y_coordinate
00125                      « ") on fiber " « boost::this_fiber::get_id()
00126                      « " reading from buffered_channel " « &c « " ...");
00127       return c.value_pop().data;
00128     }
00129
00130
00131 /** Non-blocking read to a core input port
00132
00133 \return true if the value was correctly read
00134 */
00135     bool try_read(value_type &v)override {
00136       axi_packet p;
00137
00138       if (c.try_pop(p) == boost::fibers::channel_op_status::success) {
00139         v = p.data;
00140         return true;
00141       }
00142       return false;
00143     }
00144
00145
00146 /// Connect this routing input to a switch output
00147     void connect_to(axi_stream_switch::mpl port_dest,
00148                     std::shared_ptr<communicator_port> dest) {
00149       TRISYCL_DUMP_T("connect_to router_minion " « this « " on tile("
00150                      « router_minion::axi_ss.x_coordinate
00151                      « ',' « router_minion::axi_ss.y_coordinate
00152                      « ") on fiber " « boost::this_fiber::get_id()
00153                      « " to dest " « &*dest);
00154       mpl_outputs.push_back(port_dest);
00155       outputs.push_back(dest);
00156     }
00157
00158
00159 /// Return a vector of the master ports this router is forwarding
00160     const auto& get_master_port_dests() {
00161       return mpl_outputs;
00162     }
00163
00164
00165 /// Create a router minion on this switch using a fiber executor
00166     router_minion(axi_stream_switch &axi_ss,
00167                   ::trisycl::detail::fiber_pool &fiber_executor,
00168                   int capacity = axi_ss_geo::latency)
00169       :  capacity { capacity }
00170       , axi_ss { axi_ss }
00171     {
00172 /// Use a fiber as a data mover from the input queue
00173       futures.push_back(fiber_executor.submit([&] {
00174          TRISYCL_DUMP_T("router_minion " « this « " on tile("
00175                         « router_minion::axi_ss.x_coordinate
00176                         « ',' « router_minion::axi_ss.y_coordinate
00177                         « ") on fiber " « boost::this_fiber::get_id()
00178                         « " starting with buffered_channel " « &c);
00179           for (;;) {
00180             TRISYCL_DUMP_T("router_minion " « this « " on tile("
00181                            « router_minion::axi_ss.x_coordinate
00182                            « ',' « router_minion::axi_ss.y_coordinate
00183                            « ") on fiber " « boost::this_fiber::get_id()
00184                            « " reading from buffered_channel "
00185                            « &c « " ...");
00186             auto v = c.value_pop();
00187             if (v.shutdown_request)
00188               // End the execution on shutdown packet reception
00189               break;
00190             TRISYCL_DUMP_T("router_minion " « this « " on tile("
00191                            « router_minion::axi_ss.x_coordinate
00192                            « ',' « router_minion::axi_ss.y_coordinate
00193                            « ") on fiber " « boost::this_fiber::get_id()
00194                            « " routing data value " « v.data
```

```
00195                           « " received from buffered_channel " « &c);
00196             // The routing itself is a blocking write on each output
00197             for (auto &o :  outputs) {
00198               TRISYCL_DUMP_T("router_minion " « this « " on tile("
00199                              « router_minion::axi_ss.x_coordinate
00200                              « ',' « router_minion::axi_ss.y_coordinate
00201                              « ") on fiber " « boost::this_fiber::get_id()
00202                              « " forwarding to router_minion " « o.get());
00203                o->write(v);
00204             }
00205           }
00206           TRISYCL_DUMP_T("router_minion " « this « " on tile("
00207                          « router_minion::axi_ss.x_coordinate
00208                          « ',' « router_minion::axi_ss.y_coordinate
00209                          « ") on fiber " « boost::this_fiber::get_id()
00210                          « " shutting down");
00211       }));
00212     }
00213
00214
00215 /// Destructor handling the correct infrastructure shutdown
00216     ~router_minion()override {
00217       // Send a special packet to shutdown the routing process
00218       write(axi_packet::shutdown);
00219       // Wait for shutdown to avoid calling std::terminate on destruction...
00220       join();
00221     }
00222   };
00223
00224
00225 /// Get the router_minion behind a communicator_port
00226   static router_minion *
00227   get_as_router_minion(communicator_port * rp) {
00228 #ifdef NDEBUG
00229     // No assert mode
00230     auto rm = static_cast<router_minion *>(rp);
00231 #else
00232     // In mode with assertion, check this is really a router_minion
00233     auto rm = dynamic_cast<router_minion *>(rp);
00234     if (!rm) {
00235       throw ::trisycl::runtime_error {
00236         "This is not a routable switch port" };
00237     }
00238 #endif
00239     return rm;
00240   }
00241
00242
00243 /** Validate the user port number and translate it to the physical
00244 port number
00245
00246 \param[in] user_port is the logical port number, starting to 0
00247
00248 \param[in] physical_port_min is the physical lower port number to map to
00249
00250 \param[in] physical_port_max is the physical higher port number
00251 to map to
00252
00253 \throws ::trisycl::runtime_error if the port number is invalid
00254
00255 \return the physical port number in the switch corresponding to
00256 the logical port
00257
00258 \todo This makes sense only for core tile
00259 */
00260   static auto inline translate_port = [] (int user_port,
00261                                           auto physical_port_min,
00262                                           auto physical_port_max,
00263                                           const auto& error_message) {
00264     // Cast to int since the physical port might be the enum types
00265     auto port_min = static_cast<int>(physical_port_min);
00266     auto port_max = static_cast<int>(physical_port_max);
00267     auto last_user_port = port_max - port_min;
00268     if (user_port < 0 || user_port > last_user_port)
00269       throw ::trisycl::runtime_error {
00270         (boost::format {
00271           "%1%:  %2% is not a valid port number between 0 and %3%" }
00272            % error_message % user_port % last_user_port).str() };
00273     // Return the value as the requested enum port
00274     return static_cast<decltype(physical_port_min)>(port_min + user_port);
00275   };
00276
00277
00278 /** Access the input connection behind an input port
00279
00280 \param[in] p is the input port number
00281
```

```
00282 \throws ::trisycl::runtime_error if the port number is invalid
00283 */
00284   auto &
00285   in_connection(spl sp) {
00286     using u_t = std::underlying_type_t<decltype(sp)>;
00287     auto port = static_cast<u_t>(sp);
00288     // \todo replace by std::ssize() in the future
00289     if (port >=  static_cast<std::ptrdiff_t>(std::size(input_ports)))
00290       throw ::trisycl::runtime_error {
00291         (boost::format {
00292           "in_connection:  %1% is not a valid port number between 0 and %2%" }
00293           % port % (std::size(input_ports) - 1)).str() };
00294     return input_ports[port];
00295   }
00296
00297
00298 /** Access the output connection behind an output port
00299
00300 \param[in] p is the output port number
00301
00302 \throws ::trisycl::runtime_error if the port number is invalid
00303 */
00304   auto &out_connection(mpl mp) {
00305     using u_t = std::underlying_type_t<decltype(mp)>;
00306     auto port = static_cast<u_t>(mp);
00307     // \todo replace by std::ssize() in the future
00308     if (port >= static_cast<std::ptrdiff_t>(std::size(output_ports)))
00309       throw ::trisycl::runtime_error {
00310         (boost::format {
00311           "out_connection:  %1% is not a valid port number between 0 and %2%" }
00312           % port % (std::size(output_ports) - 1)).str() };
00313     return output_ports[port];
00314   }
00315
00316
00317 /** Configure a connection of the shim AXI stream switch
00318
00319 \param[in] sp is the input port number
00320
00321 \param[in] mp is the output port number
00322
00323 \throws ::trisycl::runtime_error if the port number is invalid
00324 */
00325   void connect(spl sp, mpl mp) {
00326     TRISYCL_DUMP_T("connect " « this
00327                    « " input:  " « magic_enum::enum_name(sp)
00328                    « " to output:  " « magic_enum::enum_name(mp));
00329     get_as_router_minion(in_connection(sp).get())
00330       ->connect_to(mp, out_connection(mp));
00331   }
00332
00333   // \todo To deprecate ?
00334   using data_type = std::uint32_t;
00335   static constexpr auto stream_latency = 4;
00336
00337   struct input_port {
00338     ::trisycl::sycl_2_2::static_pipe<data_type, stream_latency> stream;
00339     bool enabled;
00340   };
00341
00342   struct output_port {
00343     ::trisycl::sycl_2_2::static_pipe<data_type, stream_latency> stream;
00344   };
00345
00346 /** The input ports used to send information to the switch
00347
00348 Use a shared pointer to represent the interconnect between
00349 switches in a shortcut way.
00350 */
00351   std::array<std::shared_ptr<communicator_port>,
00352              axi_ss_geo::nb_slave_port> input_ports;
00353
00354 /** The input ports used to send information to the switch
00355
00356 Use a shared pointer to represent the interconnect between
00357 switches in a shortcut way.
00358 */
00359   std::array<std::shared_ptr<communicator_port>,
00360              axi_ss_geo::nb_master_port> output_ports;
00361
00362 /// Keep the horizontal coordinate for debug purpose
00363   int x_coordinate;
00364
00365 /// Keep the vertical coordinate for debug purpose
00366   int y_coordinate;
00367
00368
```

```
00369 /** Get the input router port of the AXI stream switch
00370
00371 \param p is the slave_port_layout for the stream
00372 */
00373   auto & input(spl p) {
00374     // No index validation required because of type safety
00375     return input_ports[::trisycl::detail::underlying_value(p)];
00376   }
00377
00378
00379 /** Get the input router port of the AXI stream switch
00380
00381 \param p is the master_port_layout for the stream
00382 */
00383   auto & output(mpl p) {
00384     // No index validation required because of type safety
00385     return output_ports[::trisycl::detail::underlying_value(p)];
00386   }
00387
00388
00389 #if 0
00390 /** Map the user receive DMA port number to the AXI stream switch port
00391
00392 \param[in] port is the user port to use
00393 */
00394   static auto translate_rx_dma_port(int port) {
00395     return axi_ss_t::translate_port(port, spl::dma_0, spl::dma_last,
00396                                     "The receive DMA port is out of range");
00397   }
00398
00399
00400 /** Map the user transmit DMA port number to the AXI stream switch port
00401
00402 \param[in] port is the user port to use
00403 */
00404   static auto translate_tx_dma_port(int port) {
00405     return axi_ss_t::translate_port(port, mpl::dma_0, mpl::dma_last,
00406                                     "The transmit DMA port is out of range");
00407   }
00408 #endif
00409
00410
00411 /// Connect the internals of the AXI stream switch
00412   axi_stream_switch() {
00413     // \todo DMA
00414
00415     // \todo CTRL
00416   }
00417
00418
00419 /// Start the tile infrastructure associated to the AIE device
00420   void start(int x, int y, ::trisycl::detail::fiber_pool &fiber_executor) {
00421     x_coordinate = x;
00422     y_coordinate = y;
00423     // Add a router worker on all switch input
00424     for (auto& p :  input_ports) {
00425       auto port_enum =
00426         magic_enum::enum_cast<spl>(std::distance(input_ports.begin(), &p));
00427 #ifdef TRISYCL_DEBUG
00428       auto port_name = magic_enum::enum_name(port_enum.value());
00429 #endif
00430       TRISYCL_DUMP_T("axi_stream_switch " « this
00431                      « " (" « x_coordinate « ',' « y_coordinate
00432                      « ") start initialize input:  " « port_name);
00433       // If the port comes from the FIFO, use a FIFO-featured
00434       // router_minion instead
00435       if (ranges::find(axi_ss_geo::s_fifo_range, port_enum)
00436           != std::end(axi_ss_geo::s_fifo_range))
00437         /* A special AXI switch routing interface which provides a
00438 FIFO.  Since a router_minion has already some internal FIFO
00439 behavior, some deeper FIFO behavior can actually be
00440 implemented with a specific router_minion with a deeper
00441 FIFO internally, to avoid having 2 active components, the
00442 FIFO + the router_minion */
00443         p = std::make_shared<router_minion>(*this,
00444                                             fiber_executor,
00445                                             axi_ss_geo::fifo_depth);
00446       else
00447         p = std::make_shared<router_minion>(*this, fiber_executor);
00448     }
00449     // Since the FIFO behavior is actually implemented with a
00450     // router_minion with a deeper FIFO internally, the "input" of the
00451     // "FIFO" can be just directly connected to this specific
00452     // router_minion
00453     for (auto [o, i] :  ranges::views::zip(axi_ss_geo::m_fifo_range,
00454                                            axi_ss_geo::s_fifo_range))
00455       output(o) = input(i);
```

```
00456   }
00457
00458
00459 /** Compute the edge angle for a port name according a list of
00460 description associating a port name prefix and an associated
00461 angle in degree
00462 */
00463   int
00464   tikz_edge_angle(const std::string_view& port_name,
00465                   const std::initializer_list<std::tuple<std::string_view,
00466                                                          int»& desc)const {
00467     for (auto& [prefix, angle] :  desc)
00468       if (port_name.starts_with(prefix))
00469         return angle;
00470     assert(0 && "Unknown port name");
00471     return 0;
00472   }
00473
00474
00475 /// Display the internal switch connections with routing arrows
00476 /// perpendicular to the borders
00477   void display_routing_configuration(latex::context& c)const {
00478     for (const auto& [i, p] :    ranges::views::enumerate(input_ports)) {
00479       auto input_port = static_cast<spl>(i);
00480       auto rm = get_as_router_minion(p.get());
00481       for (auto output_port :  rm->get_master_port_dests()) {
00482         auto input_port_name = magic_enum::enum_name(input_port);
00483         // Compute the angle of the starting arrow from the input
00484         auto input_angle = tikz_edge_angle(input_port_name,
00485                                            { { "east", 180 }, ///< To the left
00486                                              { "north", -90 }, ///< Downwards
00487                                              { "west", 0 }, ///< To the right
00488                                              { "dma", 90 }, ///< Upwards
00489                                              { "fifo", 90 },
00490                                              { "me", 90 },
00491                                              { "tile_ctrl", 90 },
00492                                              { "south", 90 } });
00493         auto output_port_name = magic_enum::enum_name(output_port);
00494         // Compute the angle of the ending arrow to the output
00495         auto output_angle = tikz_edge_angle(output_port_name,
00496                                             { { "east", 180 }, ///< To the left
00497                                               { "north", -90 }, ///< Downwards
00498                                               { "dma", 0 }, ///< To the right
00499                                               { "fifo", 0 },
00500                                               { "me", 0 },
00501                                               { "tile_ctrl", 0 },
00502                                               { "west", 0 },
00503                                               { "south", 90 } }); ///< Upwards
00504         // Use specific color for core emission or reception
00505         auto color = input_port_name.starts_with("me") ?  "red"
00506           :  output_port_name.starts_with("me") ?  "teal"
00507           :  input_port_name.starts_with("dma") ?  "magenta"
00508           :  output_port_name.starts_with("dma") ?  "violet" :  "blue";
00509         c.add((boost::format { R"(
00510 \draw[%5%] (node cs:name=S%1%)
00511 to [out=%2%, in=%4%] (node cs:name=M%3%);)" }
00512 % c.clean_node(input_port_name) % input_angle
00513 % c.clean_node(output_port_name) % output_angle
00514 % color).str());
00515 }
00516 }
00517 }
00518
00519
00520 /// Display a border of the AXI stream switch
00521   template <typename FG, typename FL, typename...  EnumRanges>
00522   auto display_border(const std::string_view& node_attribute,
00523                       FG&& global_coordinate_function,
00524                       FL&& local_coordinate_function,
00525                       EnumRanges&&...  enum_ranges)const {
00526     std::string out;
00527     // Index each connection displayed in order
00528     auto i = 0;
00529     boost::hana::for_each(boost::hana::make_tuple(enum_ranges...),
00530       [&] (auto&& r) {
00531         for (auto p :  r) {
00532           auto [x, y] = local_coordinate_function(i);
00533           out += (boost::format { R"(
00534 \coordinate(%3%) at %2%;
00535 \node[%1%] at %2% {%4%};)" }
00536 % node_attribute
00537 % global_coordinate_function(x, y)
00538 // Make a node name from enum name prefixed with master/slave statu7s
00539             % latex::context::clean_node(magic_enum::enum_name(p),
00540                                          is_axi_master(p) ?  "M" :  "S")
00541             // Use the enum name as the port name to display
00542             % magic_enum::enum_name(p)).str();
```

```
00543           ++i;
00544         };
00545       });
00546     return out;
00547   }
00548
00549
00550 /// Compute the size of the graphics representation of the switch
00551   static vec<int, 2> display_size() {
00552     // Combine the lengths of the most packed sides,
00553     // + 1/+ 2 to keep room for labels on each side
00554     return { 1 + ranges::distance(axi_ss_geo::s_me_range)
00555              + ranges::distance(axi_ss_geo::m_dma_range)
00556              + ranges::distance(axi_ss_geo::m_tile_ctrl_range)
00557              + ranges::distance(axi_ss_geo::m_fifo_range)
00558              + ranges::distance(axi_ss_geo::m_south_range)
00559              + ranges::distance(axi_ss_geo::s_south_range),
00560              2 + ranges::distance(axi_ss_geo::s_me_range)
00561              + ranges::distance(axi_ss_geo::s_dma_range)
00562              + ranges::distance(axi_ss_geo::s_tile_ctrl_range)
00563              + ranges::distance(axi_ss_geo::s_fifo_range)
00564              + ranges::distance(axi_ss_geo::m_west_range)
00565              + ranges::distance(axi_ss_geo::s_west_range) };
00566   }
00567
00568
00569 /// Display the AXI stream switch
00570   template <typename GTC>
00571   void display(latex::context& c,
00572                const vec<int, 2>& core_size,
00573                const GTC& get_tikz_coordinate)const {
00574     const auto [ width, height ] = display_size();
00575     // Display the network interfaces on the top of the switch
00576     c.add(display_border
00577           ("rotate=90,anchor=north west", get_tikz_coordinate,
00578            [&, height = height] (auto i) {
00579              return std::tuple
00580                { core_size.x()
00581                    + ranges::distance(axi_ss_geo::s_me_range)
00582                    + ranges::distance(axi_ss_geo::s_dma_range)
00583                    + ranges::distance(axi_ss_geo::s_tile_ctrl_range)
00584                    + ranges::distance(axi_ss_geo::s_fifo_range) + i,
00585                  core_size.y() + height - 1 };
00586            },
00587            axi_ss_geo::m_north_range,
00588            axi_ss_geo::s_north_range));
00589     // Display the network interfaces on the bottom of the switch
00590     c.add(display_border("rotate=90,anchor=north east",
00591                          get_tikz_coordinate,
00592                          [&] (auto i) {
00593                            return std::tuple { core_size.x() + i,
00594                                                core_size.y() };
00595                          },
00596                          ranges::views::reverse(axi_ss_geo::s_me_range),
00597                          axi_ss_geo::s_dma_range,
00598                          axi_ss_geo::s_tile_ctrl_range,
00599                          axi_ss_geo::s_fifo_range,
00600                          axi_ss_geo::s_south_range,
00601                          axi_ss_geo::m_south_range));
00602     // Display the network interfaces on the left of the switch
00603     c.add(display_border("anchor=south east",
00604                          get_tikz_coordinate,
00605                          [&] (auto i) {
00606                            return std::tuple { core_size.x() - 1,
00607                                                core_size.y() + i + 1 };
00608                          },
00609                          ranges::views::reverse(axi_ss_geo::m_me_range),
00610                          axi_ss_geo::m_dma_range,
00611                          axi_ss_geo::m_tile_ctrl_range,
00612                          axi_ss_geo::m_fifo_range,
00613                          axi_ss_geo::m_west_range,
00614                          axi_ss_geo::s_west_range));
00615     // Display the network interfaces on the right of the switch
00616     c.add(display_border
00617           ("anchor=south west",
00618            get_tikz_coordinate,
00619            [&, width = width] (auto i) {
00620              return std::tuple
00621                { core_size.x() + width - 1,
00622                  core_size.y() + i + 1
00623                  + ranges::distance(axi_ss_geo::m_me_range)
00624                  + ranges::distance(axi_ss_geo::m_dma_range)
00625                  + ranges::distance(axi_ss_geo::m_tile_ctrl_range)
00626                  + ranges::distance(axi_ss_geo::m_fifo_range) };
00627            },
00628            axi_ss_geo::s_east_range,
00629            axi_ss_geo::m_east_range));
```

```
00630
00631     display_routing_configuration(c);
00632   }
00633
00634 };
00635
00636 /// @} End the aie Doxygen group
00637
00638 }
00639
00640 /*
00641 # Some Emacs stuff:
00642 ### Local Variables:
00643 ### ispell-local-dictionary:  "american"
00644 ### eval:  (flyspell-prog-mode)
00645 ### End:
00646 */
00647
00648 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_AXI_STREAM_SWITCH_HPP
```

13.255 include/triSYCL/vendor/Xilinx/acap/aie/axi_stream_switch.hpp File Reference

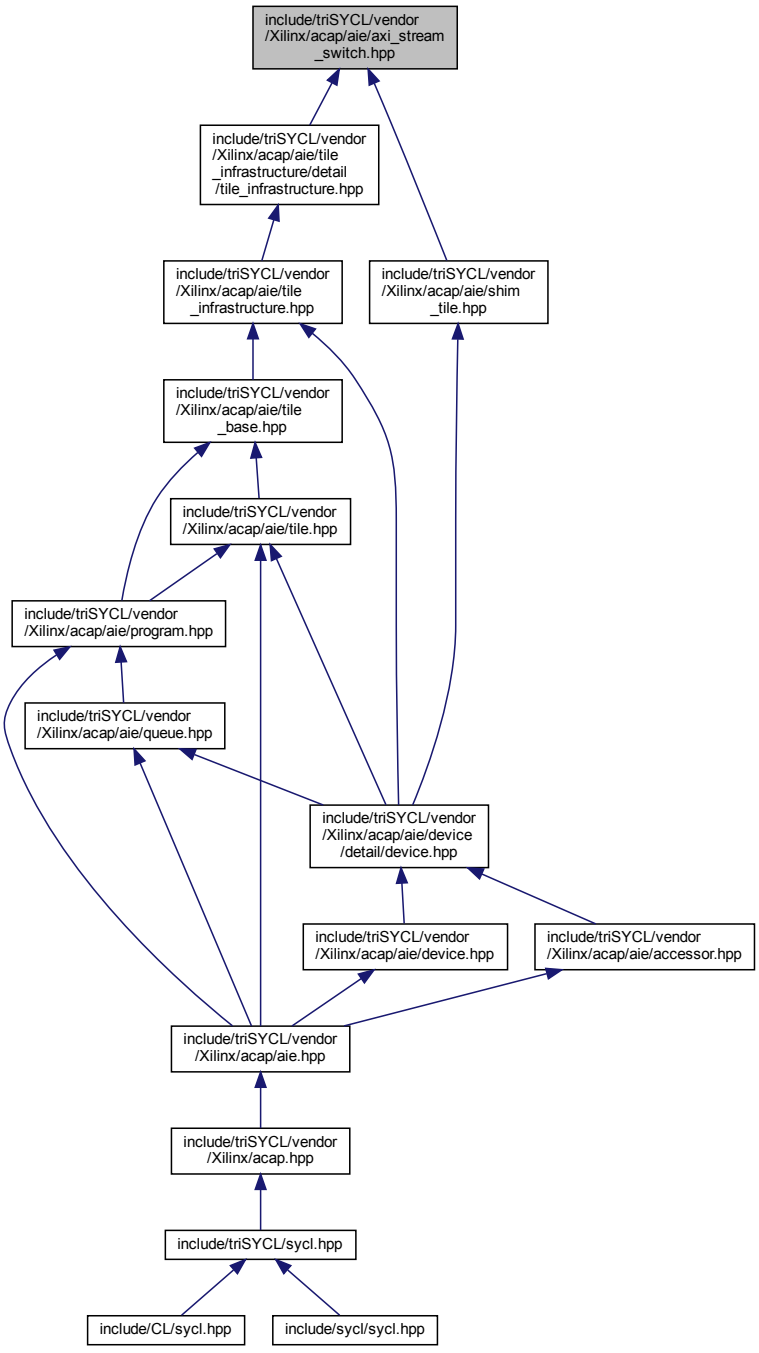
Model of an AXI stream switch.

```
#include <array>
#include <cstdint>
#include <future>
#include <memory>
#include <string>
#include <type_traits>
#include <utility>
#include <boost/fiber/all.hpp>
#include <boost/format.hpp>
#include "magic_enum.hpp"
#include <range/v3/all.hpp>
#include "connection.hpp"
#include "triSYCL/detail/debug.hpp"
#include "triSYCL/detail/enum.hpp"
#include "triSYCL/detail/fiber_pool.hpp"
#include "triSYCL/vendor/Xilinx/latex.hpp"
```

Include dependency graph for axi_stream_switch.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >

  An AXI stream switch parameterized with some geographic layout. More...
- class trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion

  A router input port with routing skills.
- struct trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input_port
- struct trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::output_port

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.255.1 Detailed Description

Model of an AXI stream switch.

Based on Math Engine (ME) Architecture Specification, Revision v2.01 March 2019:

- section "4.2 AXI-Stream Tile Interconnect"
- section "6.5.3 Shim AXI-Stream Interconnect"

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file axi_stream_switch.hpp.

13.256 axi_stream_switch.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_AXI_STREAM_SWITCH_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_AXI_STREAM_SWITCH_HPP
00003
00004 /** \file
00005
00006 Model of an AXI stream switch
00007
00008 Based on Math Engine (ME) Architecture Specification, Revision v2.01
00009 March 2019:
00010
00011 - section "4.2 AXI-Stream Tile Interconnect"
00012
00013 - section "6.5.3 Shim AXI-Stream Interconnect"
00014
00015 Ronan dot Keryell at Xilinx dot com
00016
00017 This file is distributed under the University of Illinois Open Source
00018 License.  See LICENSE.TXT for details.
00019 */
00020
00021 #include <array>
00022 #include <cstdint>
00023 #include <future>
00024 #include <memory>
00025 #include <string>
00026 #include <type_traits>
00027 #include <utility>
00028
00029 #include <boost/fiber/all.hpp>
00030 #include <boost/format.hpp>
00031 #include "magic_enum.hpp"
00032 #include <range/v3/all.hpp>
00033
00034 #include "connection.hpp"
00035 #include "triSYCL/detail/debug.hpp"
00036 #include "triSYCL/detail/enum.hpp"
00037 #include "triSYCL/detail/fiber_pool.hpp"
```

```
00038 #include "triSYCL/vendor/Xilinx/latex.hpp"
00039
00040 namespace trisycl::vendor::xilinx::acap::aie {
00041
00042 /// \ingroup aie
00043 /// @{
00044
00045 /** An AXI stream switch parameterized with some geographic layout
00046
00047 \param[in] AxiStreamGeography is the geography class used to
00048 instantiate the AXI stream switch.
00049 */
00050 template <typename AXIStreamGeography>
00051 class axi_stream_switch
00052   :  ::trisycl::detail::debug<axi_stream_switch <AXIStreamGeography» {
00053
00054 public:
00055
00056   using axi_ss_geo = AXIStreamGeography;
00057   using mpl = typename axi_ss_geo::master_port_layout;
00058   using spl = typename axi_ss_geo::slave_port_layout;
00059
00060 /// A router input port with routing skills
00061   class router_minion :  public communicator_port {
00062 public  :
00063 /// Router ingress capacity queue
00064     const int capacity;
00065
00066 /// Ingress packet queue
00067     boost::fibers::buffered_channel<axi_packet> c
00068       { static_cast<std::size_t>(capacity) };
00069
00070 /// The ids of the outputs port the router minion has to forward to.
00071 /// Used by introspection to track current routing configuration
00072     std::vector<axi_stream_switch::mpl> mpl_outputs;
00073
00074 /** The outputs the router minion has to forward to.
00075 This what is used to speed up the routing itself even if the id
00076 could be enough */
00077     std::vector<std::shared_ptr<communicator_port» outputs;
00078
00079 /// To shepherd the routing fibers
00080     std::vector<::trisycl::detail::fiber_pool::future<void» futures;
00081
00082 /// Keep track of the AXI stream switch owning this port for debugging
00083     axi_stream_switch &axi_ss;
00084
00085 /// Wait for the fibers to complete
00086     void join() {
00087       // Wait on all the fibers to finish
00088       for (auto &f :  futures)
00089         // Get the value of the future, to get an exception if any
00090         f.get();
00091     }
00092
00093   public:
00094
00095     using value_type = axi_packet::value_type;
00096
00097 /// Enqueue a packet on the router input
00098 /// \todo Clean up the API to separate data from signaling
00099     void write(const axi_packet &v)override {
00100 /// \todo separate debug from shutdown case
00101       TRISYCL_DUMP_T("router_minion " « this
00102                      « " on tile(" « router_minion::axi_ss.x_coordinate « ','
00103                      « router_minion::axi_ss.y_coordinate
00104                      « ") on fiber " « boost::this_fiber::get_id()
00105                      « " write data value " « v.data
00106                      « " to buffered_channel " « &c);
00107       c.push(v);
00108     }
00109
00110
00111 /** Try to enqueue a packet on the router input
00112
00113 \return true if the packet is correctly enqueued
00114 */
00115     bool try_write(const axi_packet &v)override {
00116       return c.try_push(v) != boost::fibers::channel_op_status::full;
00117     }
00118
00119
00120 /// Waiting read to a core input port
00121     value_type read()override {
00122       TRISYCL_DUMP_T("router_minion " « this « " on tile("
00123                      « router_minion::axi_ss.x_coordinate
00124                      « ',' « router_minion::axi_ss.y_coordinate
```

```
00125                     « ") on fiber " « boost::this_fiber::get_id()
00126                     « " reading from buffered_channel " « &c « " ...");
00127       return c.value_pop().data;
00128     }
00129
00130
00131 /** Non-blocking read to a core input port
00132
00133 \return true if the value was correctly read
00134 */
00135     bool try_read(value_type &v)override {
00136       axi_packet p;
00137
00138       if (c.try_pop(p) == boost::fibers::channel_op_status::success) {
00139         v = p.data;
00140         return true;
00141       }
00142       return false;
00143     }
00144
00145
00146 /// Connect this routing input to a switch output
00147     void connect_to(axi_stream_switch::mpl port_dest,
00148                     std::shared_ptr<communicator_port> dest) {
00149       TRISYCL_DUMP_T("connect_to router_minion " « this « " on tile("
00150                      « router_minion::axi_ss.x_coordinate
00151                      « ',' « router_minion::axi_ss.y_coordinate
00152                      « ") on fiber " « boost::this_fiber::get_id()
00153                      « " to dest " « &*dest);
00154       mpl_outputs.push_back(port_dest);
00155       outputs.push_back(dest);
00156     }
00157
00158
00159 /// Return a vector of the master ports this router is forwarding
00160     const auto& get_master_port_dests() {
00161       return mpl_outputs;
00162     }
00163
00164
00165 /// Create a router minion on this switch using a fiber executor
00166     router_minion(axi_stream_switch &axi_ss,
00167                   ::trisycl::detail::fiber_pool &fiber_executor,
00168                   int capacity = axi_ss_geo::latency)
00169       :  capacity { capacity }
00170       , axi_ss { axi_ss }
00171     {
00172 /// Use a fiber as a data mover from the input queue
00173       futures.push_back(fiber_executor.submit([&] {
00174          TRISYCL_DUMP_T("router_minion " « this « " on tile("
00175                         « router_minion::axi_ss.x_coordinate
00176                         « ',' « router_minion::axi_ss.y_coordinate
00177                         « ") on fiber " « boost::this_fiber::get_id()
00178                         « " starting with buffered_channel " « &c);
00179           for (;;) {
00180             TRISYCL_DUMP_T("router_minion " « this « " on tile("
00181                            « router_minion::axi_ss.x_coordinate
00182                            « ',' « router_minion::axi_ss.y_coordinate
00183                            « ") on fiber " « boost::this_fiber::get_id()
00184                            « " reading from buffered_channel "
00185                            « &c « " ...");
00186             auto v = c.value_pop();
00187             if (v.shutdown_request)
00188               // End the execution on shutdown packet reception
00189               break;
00190             TRISYCL_DUMP_T("router_minion " « this « " on tile("
00191                            « router_minion::axi_ss.x_coordinate
00192                            « ',' « router_minion::axi_ss.y_coordinate
00193                            « ") on fiber " « boost::this_fiber::get_id()
00194                            « " routing data value " « v.data
00195                            « " received from buffered_channel " « &c);
00196             // The routing itself is a blocking write on each output
00197             for (auto &o :  outputs) {
00198               TRISYCL_DUMP_T("router_minion " « this « " on tile("
00199                              « router_minion::axi_ss.x_coordinate
00200                              « ',' « router_minion::axi_ss.y_coordinate
00201                              « ") on fiber " « boost::this_fiber::get_id()
00202                              « " forwarding to router_minion " « o.get());
00203                o->write(v);
00204             }
00205           }
00206           TRISYCL_DUMP_T("router_minion " « this « " on tile("
00207                          « router_minion::axi_ss.x_coordinate
00208                          « ',' « router_minion::axi_ss.y_coordinate
00209                          « ") on fiber " « boost::this_fiber::get_id()
00210                          « " shutting down");
00211       }));
```

```
00212     }
00213
00214
00215 /// Destructor handling the correct infrastructure shutdown
00216     ~router_minion()override {
00217       // Send a special packet to shutdown the routing process
00218       write(axi_packet::shutdown);
00219       // Wait for shutdown to avoid calling std::terminate on destruction...
00220       join();
00221     }
00222   };
00223
00224
00225 /// Get the router_minion behind a communicator_port
00226   static router_minion *
00227   get_as_router_minion(communicator_port * rp) {
00228 #ifdef NDEBUG
00229     // No assert mode
00230     auto rm = static_cast<router_minion *>(rp);
00231 #else
00232     // In mode with assertion, check this is really a router_minion
00233     auto rm = dynamic_cast<router_minion *>(rp);
00234     if (!rm) {
00235       throw ::trisycl::runtime_error {
00236         "This is not a routable switch port" };
00237     }
00238 #endif
00239     return rm;
00240   }
00241
00242
00243 /** Validate the user port number and translate it to the physical
00244 port number
00245
00246 \param[in] user_port is the logical port number, starting to 0
00247
00248 \param[in] physical_port_min is the physical lower port number to map to
00249
00250 \param[in] physical_port_max is the physical higher port number
00251 to map to
00252
00253 \throws ::trisycl::runtime_error if the port number is invalid
00254
00255 \return the physical port number in the switch corresponding to
00256 the logical port
00257
00258 \todo This makes sense only for core tile
00259 */
00260   static auto inline translate_port = [] (int user_port,
00261                                           auto physical_port_min,
00262                                           auto physical_port_max,
00263                                           const auto& error_message) {
00264     // Cast to int since the physical port might be the enum types
00265     auto port_min = static_cast<int>(physical_port_min);
00266     auto port_max = static_cast<int>(physical_port_max);
00267     auto last_user_port = port_max - port_min;
00268     if (user_port < 0 || user_port > last_user_port)
00269       throw ::trisycl::runtime_error {
00270         (boost::format {
00271           "%1%:  %2% is not a valid port number between 0 and %3%" }
00272           % error_message % user_port % last_user_port).str() };
00273     // Return the value as the requested enum port
00274     return static_cast<decltype(physical_port_min)>(port_min + user_port);
00275   };
00276
00277
00278 /** Access the input connection behind an input port
00279
00280 \param[in] p is the input port number
00281
00282 \throws ::trisycl::runtime_error if the port number is invalid
00283 */
00284   auto &
00285   in_connection(spl sp) {
00286     using u_t = std::underlying_type_t<decltype(sp)>;
00287     auto port = static_cast<u_t>(sp);
00288     // \todo replace by std::ssize() in the future
00289     if (port >=  static_cast<std::ptrdiff_t>(std::size(input_ports)))
00290       throw ::trisycl::runtime_error {
00291         (boost::format {
00292           "in_connection:  %1% is not a valid port number between 0 and %2%" }
00293           % port % (std::size(input_ports) - 1)).str() };
00294     return input_ports[port];
00295   }
00296
00297
00298 /** Access the output connection behind an output port
```

```
00299
00300 \param[in] p is the output port number
00301
00302 \throws ::trisycl::runtime_error if the port number is invalid
00303 */
00304   auto &out_connection(mpl mp) {
00305     using u_t = std::underlying_type_t<decltype(mp)>;
00306     auto port = static_cast<u_t>(mp);
00307     // \todo replace by std::ssize() in the future
00308     if (port >= static_cast<std::ptrdiff_t>(std::size(output_ports)))
00309       throw ::trisycl::runtime_error {
00310         (boost::format {
00311           "out_connection:  %1% is not a valid port number between 0 and %2%" }
00312          % port % (std::size(output_ports) - 1)).str() };
00313     return output_ports[port];
00314   }
00315
00316
00317 /** Configure a connection of the shim AXI stream switch
00318
00319 \param[in] sp is the input port number
00320
00321 \param[in] mp is the output port number
00322
00323 \throws ::trisycl::runtime_error if the port number is invalid
00324 */
00325   void connect(spl sp, mpl mp) {
00326     TRISYCL_DUMP_T("connect " « this
00327                    « " input:  " « magic_enum::enum_name(sp)
00328                    « " to output:  " « magic_enum::enum_name(mp));
00329     get_as_router_minion(in_connection(sp).get())
00330       ->connect_to(mp, out_connection(mp));
00331   }
00332
00333   // \todo To deprecate ?
00334   using data_type = std::uint32_t;
00335   static constexpr auto stream_latency = 4;
00336
00337   struct input_port {
00338     ::trisycl::sycl_2_2::static_pipe<data_type, stream_latency> stream;
00339     bool enabled;
00340   };
00341
00342   struct output_port {
00343     ::trisycl::sycl_2_2::static_pipe<data_type, stream_latency> stream;
00344   };
00345
00346 /** The input ports used to send information to the switch
00347
00348 Use a shared pointer to represent the interconnect between
00349 switches in a shortcut way.
00350 */
00351   std::array<std::shared_ptr<communicator_port>,
00352              axi_ss_geo::nb_slave_port> input_ports;
00353
00354 /** The input ports used to send information to the switch
00355
00356 Use a shared pointer to represent the interconnect between
00357 switches in a shortcut way.
00358 */
00359   std::array<std::shared_ptr<communicator_port>,
00360              axi_ss_geo::nb_master_port> output_ports;
00361
00362 /// Keep the horizontal coordinate for debug purpose
00363   int x_coordinate;
00364
00365 /// Keep the vertical coordinate for debug purpose
00366   int y_coordinate;
00367
00368
00369 /** Get the input router port of the AXI stream switch
00370
00371 \param p is the slave_port_layout for the stream
00372 */
00373   auto & input(spl p) {
00374     // No index validation required because of type safety
00375     return input_ports[::trisycl::detail::underlying_value(p)];
00376   }
00377
00378
00379 /** Get the input router port of the AXI stream switch
00380
00381 \param p is the master_port_layout for the stream
00382 */
00383   auto & output(mpl p) {
00384     // No index validation required because of type safety
00385     return output_ports[::trisycl::detail::underlying_value(p)];
```

```
00386   }
00387
00388
00389 #if 0
00390 /** Map the user receive DMA port number to the AXI stream switch port
00391
00392 \param[in] port is the user port to use
00393 */
00394   static auto translate_rx_dma_port(int port) {
00395     return axi_ss_t::translate_port(port, spl::dma_0, spl::dma_last,
00396                                     "The receive DMA port is out of range");
00397   }
00398
00399
00400 /** Map the user transmit DMA port number to the AXI stream switch port
00401
00402 \param[in] port is the user port to use
00403 */
00404   static auto translate_tx_dma_port(int port) {
00405     return axi_ss_t::translate_port(port, mpl::dma_0, mpl::dma_last,
00406                                     "The transmit DMA port is out of range");
00407   }
00408 #endif
00409
00410
00411 /// Connect the internals of the AXI stream switch
00412   axi_stream_switch() {
00413     // \todo DMA
00414
00415     // \todo CTRL
00416   }
00417
00418
00419 /// Start the tile infrastructure associated to the AIE device
00420   void start(int x, int y, ::trisycl::detail::fiber_pool &fiber_executor) {
00421     x_coordinate = x;
00422     y_coordinate = y;
00423     // Add a router worker on all switch input
00424     for (auto& p :  input_ports) {
00425       auto port_enum =
00426         magic_enum::enum_cast<spl>(std::distance(input_ports.begin(), &p));
00427 #ifdef TRISYCL_DEBUG
00428       auto port_name = magic_enum::enum_name(port_enum.value());
00429 #endif
00430       TRISYCL_DUMP_T("axi_stream_switch " << this
00431                      << " (" << x_coordinate << ',' << y_coordinate
00432                      << ") start initialize input:  " << port_name);
00433       // If the port comes from the FIFO, use a FIFO-featured
00434       // router_minion instead
00435       if (ranges::find(axi_ss_geo::s_fifo_range, port_enum)
00436           != std::end(axi_ss_geo::s_fifo_range))
00437         /* A special AXI switch routing interface which provides a
00438 FIFO.  Since a router_minion has already some internal FIFO
00439 behavior, some deeper FIFO behavior can actually be
00440 implemented with a specific router_minion with a deeper
00441 FIFO internally, to avoid having 2 active components, the
00442 FIFO + the router_minion */
00443         p = std::make_shared<router_minion>(*this,
00444                                             fiber_executor,
00445                                             axi_ss_geo::fifo_depth);
00446       else
00447         p = std::make_shared<router_minion>(*this, fiber_executor);
00448     }
00449     // Since the FIFO behavior is actually implemented with a
00450     // router_minion with a deeper FIFO internally, the "input" of the
00451     // "FIFO" can be just directly connected to this specific
00452     // router_minion
00453     for (auto [o, i] :  ranges::views::zip(axi_ss_geo::m_fifo_range,
00454                                            axi_ss_geo::s_fifo_range))
00455       output(o) = input(i);
00456   }
00457
00458
00459 /** Compute the edge angle for a port name according a list of
00460 description associating a port name prefix and an associated
00461 angle in degree
00462 */
00463   int
00464   tikz_edge_angle(const std::string_view& port_name,
00465                   const std::initializer_list<std::tuple<std::string_view,
00466                                                          int>>& desc)const {
00467     for (auto& [prefix, angle] :  desc)
00468       if (port_name.starts_with(prefix))
00469         return angle;
00470     assert(0 && "Unknown port name");
00471     return 0;
00472   }
```

```
00473
00474
00475 /// Display the internal switch connections with routing arrows
00476 /// perpendicular to the borders
00477   void display_routing_configuration(latex::context& c)const {
00478     for (const auto& [i, p] :    ranges::views::enumerate(input_ports)) {
00479       auto input_port = static_cast<spl>(i);
00480       auto rm = get_as_router_minion(p.get());
00481       for (auto output_port :  rm->get_master_port_dests()) {
00482         auto input_port_name = magic_enum::enum_name(input_port);
00483         // Compute the angle of the starting arrow from the input
00484         auto input_angle = tikz_edge_angle(input_port_name,
00485                                            { { "east", 180 }, ///< To the left
00486                                              { "north", -90 }, ///< Downwards
00487                                              { "west", 0 }, ///< To the right
00488                                              { "dma", 90 }, ///< Upwards
00489                                              { "fifo", 90 },
00490                                              { "me", 90 },
00491                                              { "tile_ctrl", 90 },
00492                                              { "south", 90 } });
00493         auto output_port_name = magic_enum::enum_name(output_port);
00494         // Compute the angle of the ending arrow to the output
00495         auto output_angle = tikz_edge_angle(output_port_name,
00496                                             { { "east", 180 }, ///< To the left
00497                                               { "north", -90 }, ///< Downwards
00498                                               { "dma", 0 }, ///< To the right
00499                                               { "fifo", 0 },
00500                                               { "me", 0 },
00501                                               { "tile_ctrl", 0 },
00502                                               { "west", 0 },
00503                                               { "south", 90 } }); ///< Upwards
00504         // Use specific color for core emission or reception
00505         auto color = input_port_name.starts_with("me") ?  "red"
00506           :  output_port_name.starts_with("me") ?  "teal"
00507           :  input_port_name.starts_with("dma") ?  "magenta"
00508           :  output_port_name.starts_with("dma") ?  "violet" :  "blue";
00509         c.add((boost::format { R"(
00510 \draw[%5%] (node cs:name=S%1%)
00511 to [out=%2%, in=%4%] (node cs:name=M%3%);)" }
00512 % c.clean_node(input_port_name) % input_angle
00513 % c.clean_node(output_port_name) % output_angle
00514 % color).str());
00515 }
00516 }
00517 }
00518
00519
00520 /// Display a border of the AXI stream switch
00521   template <typename FG, typename FL, typename...  EnumRanges>
00522   auto display_border(const std::string_view& node_attribute,
00523                       FG&& global_coordinate_function,
00524                       FL&& local_coordinate_function,
00525                       EnumRanges&&...  enum_ranges)const {
00526     std::string out;
00527     // Index each connection displayed in order
00528     auto i = 0;
00529     boost::hana::for_each(boost::hana::make_tuple(enum_ranges...),
00530       [&] (auto&& r) {
00531         for (auto p :  r) {
00532           auto [x, y] = local_coordinate_function(i);
00533           out += (boost::format { R"(
00534 \coordinate(%3%) at %2%;
00535 \node[%1%] at %2% {%4%};)" }
00536 % node_attribute
00537 % global_coordinate_function(x, y)
00538 // Make a node name from enum name prefixed with master/slave statu7s
00539             % latex::context::clean_node(magic_enum::enum_name(p),
00540                                          is_axi_master(p) ?  "M" :  "S")
00541             // Use the enum name as the port name to display
00542             % magic_enum::enum_name(p)).str();
00543           ++i;
00544         };
00545       });
00546     return out;
00547   }
00548
00549
00550 /// Compute the size of the graphics representation of the switch
00551   static vec<int, 2> display_size() {
00552     // Combine the lengths of the most packed sides,
00553     // + 1/+ 2 to keep room for labels on each side
00554     return { 1 + ranges::distance(axi_ss_geo::s_me_range)
00555              + ranges::distance(axi_ss_geo::m_dma_range)
00556              + ranges::distance(axi_ss_geo::m_tile_ctrl_range)
00557              + ranges::distance(axi_ss_geo::m_fifo_range)
00558              + ranges::distance(axi_ss_geo::m_south_range)
00559              + ranges::distance(axi_ss_geo::s_south_range),
```

```
00560               2 + ranges::distance(axi_ss_geo::s_me_range)
00561               + ranges::distance(axi_ss_geo::s_dma_range)
00562               + ranges::distance(axi_ss_geo::s_tile_ctrl_range)
00563               + ranges::distance(axi_ss_geo::s_fifo_range)
00564               + ranges::distance(axi_ss_geo::m_west_range)
00565               + ranges::distance(axi_ss_geo::s_west_range) };
00566   }
00567
00568
00569 /// Display the AXI stream switch
00570   template <typename GTC>
00571   void display(latex::context& c,
00572                const vec<int, 2>& core_size,
00573                const GTC& get_tikz_coordinate)const {
00574     const auto [ width, height ] = display_size();
00575     // Display the network interfaces on the top of the switch
00576     c.add(display_border
00577           ("rotate=90,anchor=north west", get_tikz_coordinate,
00578            [&, height = height] (auto i) {
00579              return std::tuple
00580                { core_size.x()
00581                    + ranges::distance(axi_ss_geo::s_me_range)
00582                    + ranges::distance(axi_ss_geo::s_dma_range)
00583                    + ranges::distance(axi_ss_geo::s_tile_ctrl_range)
00584                    + ranges::distance(axi_ss_geo::s_fifo_range) + i,
00585                  core_size.y() + height - 1 };
00586            },
00587            axi_ss_geo::m_north_range,
00588            axi_ss_geo::s_north_range));
00589     // Display the network interfaces on the bottom of the switch
00590     c.add(display_border("rotate=90,anchor=north east",
00591                          get_tikz_coordinate,
00592                          [&] (auto i) {
00593                            return std::tuple { core_size.x() + i,
00594                                                core_size.y() };
00595                          },
00596                          ranges::views::reverse(axi_ss_geo::s_me_range),
00597                          axi_ss_geo::s_dma_range,
00598                          axi_ss_geo::s_tile_ctrl_range,
00599                          axi_ss_geo::s_fifo_range,
00600                          axi_ss_geo::s_south_range,
00601                          axi_ss_geo::m_south_range));
00602     // Display the network interfaces on the left of the switch
00603     c.add(display_border("anchor=south east",
00604                          get_tikz_coordinate,
00605                          [&] (auto i) {
00606                            return std::tuple { core_size.x() - 1,
00607                                                core_size.y() + i + 1 };
00608                          },
00609                          ranges::views::reverse(axi_ss_geo::m_me_range),
00610                          axi_ss_geo::m_dma_range,
00611                          axi_ss_geo::m_tile_ctrl_range,
00612                          axi_ss_geo::m_fifo_range,
00613                          axi_ss_geo::m_west_range,
00614                          axi_ss_geo::s_west_range));
00615     // Display the network interfaces on the right of the switch
00616     c.add(display_border
00617           ("anchor=south west",
00618            get_tikz_coordinate,
00619            [&, width = width] (auto i) {
00620              return std::tuple
00621                { core_size.x() + width - 1,
00622                  core_size.y() + i + 1
00623                  + ranges::distance(axi_ss_geo::m_me_range)
00624                  + ranges::distance(axi_ss_geo::m_dma_range)
00625                  + ranges::distance(axi_ss_geo::m_tile_ctrl_range)
00626                  + ranges::distance(axi_ss_geo::m_fifo_range) };
00627            },
00628            axi_ss_geo::s_east_range,
00629            axi_ss_geo::m_east_range));
00630
00631     display_routing_configuration(c);
00632   }
00633
00634 };
00635
00636 /// @} End the aie Doxygen group
00637
00638 }
00639
00640 /*
00641 # Some Emacs stuff:
00642 ### Local Variables:
00643 ### ispell-local-dictionary:  "american"
00644 ### eval:  (flyspell-prog-mode)
00645 ### End:
00646 */
```

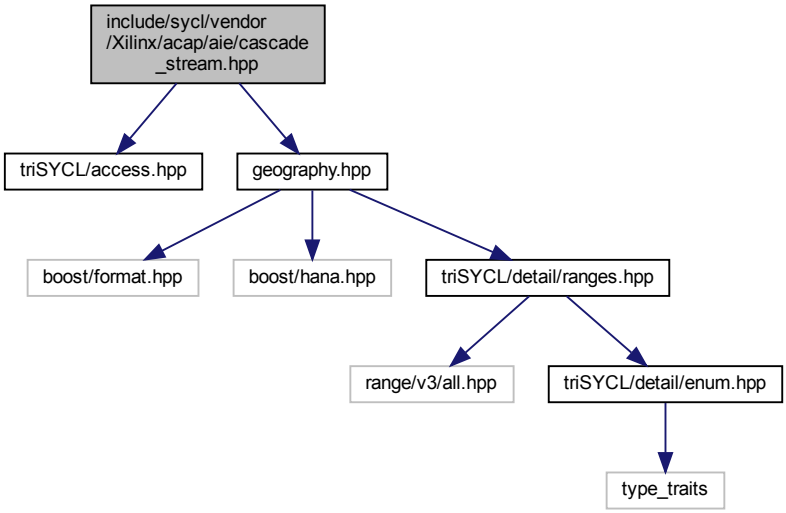
```
00647
00648 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_AXI_STREAM_SWITCH_HPP
```

13.257 include/sycl/vendor/Xilinx/acap/aie/cascade_stream.hpp File Reference

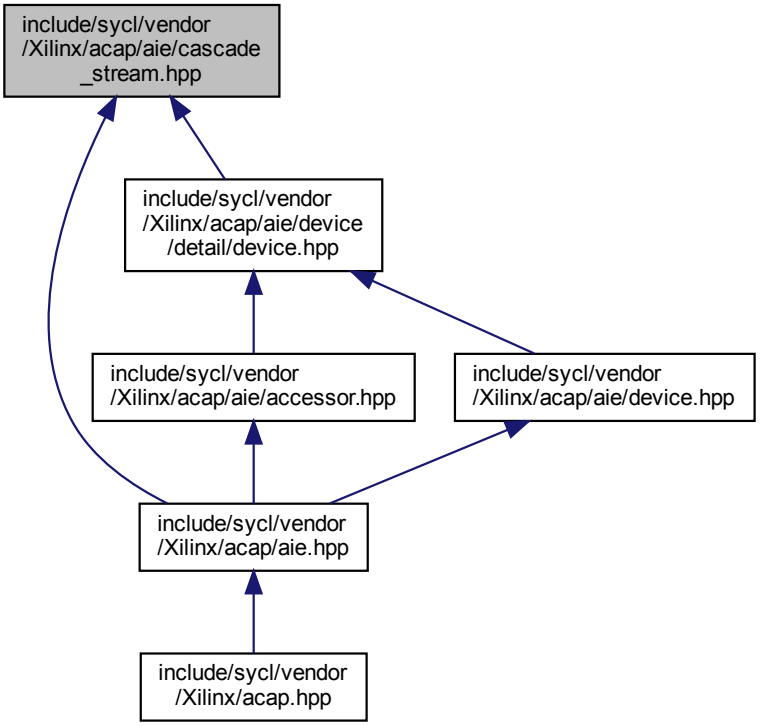
The cascade stream infrastructure between AI Engine tiles.

```
#include "triSYCL/access.hpp"
#include "geography.hpp"
```

Include dependency graph for cascade_stream.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >

  Implementation of the cascade stream infrastructure between AI Engine tiles. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.257.1 Detailed Description

The cascade stream infrastructure between AI Engine tiles.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file cascade_stream.hpp.

13.258 cascade_stream.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_CASCADE_STREAM_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_CASCADE_STREAM_HPP
00003
00004 /** \file
00005
00006 The cascade stream infrastructure between AI Engine tiles
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include "triSYCL/access.hpp"
00015 #include "geography.hpp"
00016
00017 namespace trisycl::vendor::xilinx::acap::aie {
00018
00019 /// \ingroup aie
00020 /// @{
00021
00022 /** Implementation of the cascade stream infrastructure between AI
00023 Engine tiles
00024
00025 Based on Math Engine (ME) Architecture Specification, Revision v2.01
00026
00027 March 2019
00028
00029 3.5.4 Data Movement using Cascade Streams, p.  93
00030
00031 2.14 Device-level Floorplanning Guidelines, 2.14.2 Array Edges, p.  69
00032
00033 4.4 ME Core Interfaces, 4.4.4 Cascade Stream Interface, p.  120
00034 explains there is a FIFO depth of 4
00035
00036 5.9.3.6 Cascade Streams:  explains that there are only blocking
00037 accesses
00038
00039 Direct stream interface:  One cascade stream in, one cascade stream
00040 out (384-bits)
00041
00042 \todo Remove this global infrastructure and distribute it on the
00043 tile_infrastructure itself
00044 */
00045 template <typename Geography>
00046 struct cascade_stream {
00047   using geo = Geography;
00048 /** The pipes for the cascade streams, with 1 spare pipe on each
00049 side of the PE string
00050
00051 \todo Use a data type with 384 bits
00052 */
00053   ::trisycl::sycl_2_2::static_pipe<int, 4>
00054   cascade_stream_pipes[geo::x_size*geo::y_size + 1];
00055
00056   /* Cascade stream layout
00057
00058 On even rows, a tile use cascade_stream_pipes[y][x] as input and
00059 cascade_stream_pipes[y][x + 1] as output
00060
00061 On odd rows the flow goes into the other direction, so a tile
00062 use cascade_stream_pipes[y][x + 1] as input and
00063 cascade_stream_pipes[y][x] as output
00064 */
00065
00066 /** Get a blocking read accessor to the cascade stream input
00067
00068 \param T is the data type used to read from the cascade
00069 stream pipe
00070
00071 \param[in] x is the horizontal tile coordinate
00072
00073 \param[in] y is the vertical tile coordinate
00074 */
00075   template <typename T>
00076   auto get_cascade_stream_in(int x, int y)const {
00077     return cascade_stream_pipes[geo::cascade_linear_id(x, y)]
00078       .template get_access<access::mode::read,
00079                            access::target::blocking_pipe>();
00080   }
00081
00082
```

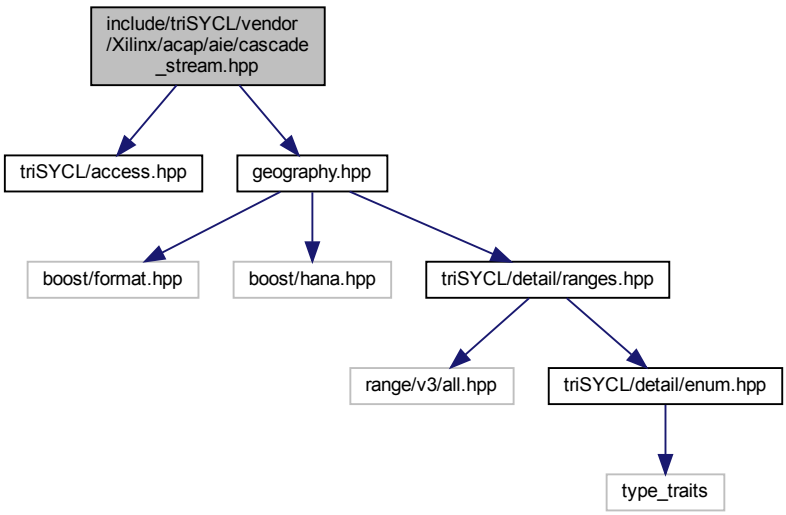
```
00083 /** Get a blocking write accessor to the cascade stream output
00084
00085 \param T is the data type used to write to the cascade
00086 stream pipe
00087
00088 \param[in] x is the horizontal tile coordinate
00089
00090 \param[in] y is the vertical tile coordinate
00091 */
00092   template <typename T>
00093   auto get_cascade_stream_out(int x, int y)const {
00094     // The output is connected to the down-stream neighbour of the cascade
00095     return cascade_stream_pipes[geo::cascade_linear_id(x, y) + 1]
00096       .template get_access<access::mode::write,
00097                            access::target::blocking_pipe>();
00098   }
00099
00100 };
00101
00102 /// @} End the aie Doxygen group
00103
00104 }
00105
00106 /*
00107 # Some Emacs stuff:
00108 ### Local Variables:
00109 ### ispell-local-dictionary:  "american"
00110 ### eval:  (flyspell-prog-mode)
00111 ### End:
00112 */
00113
00114 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_CASCADE_STREAM_HPP
```

13.259 include/triSYCL/vendor/Xilinx/acap/aie/cascade_stream.hpp File Reference

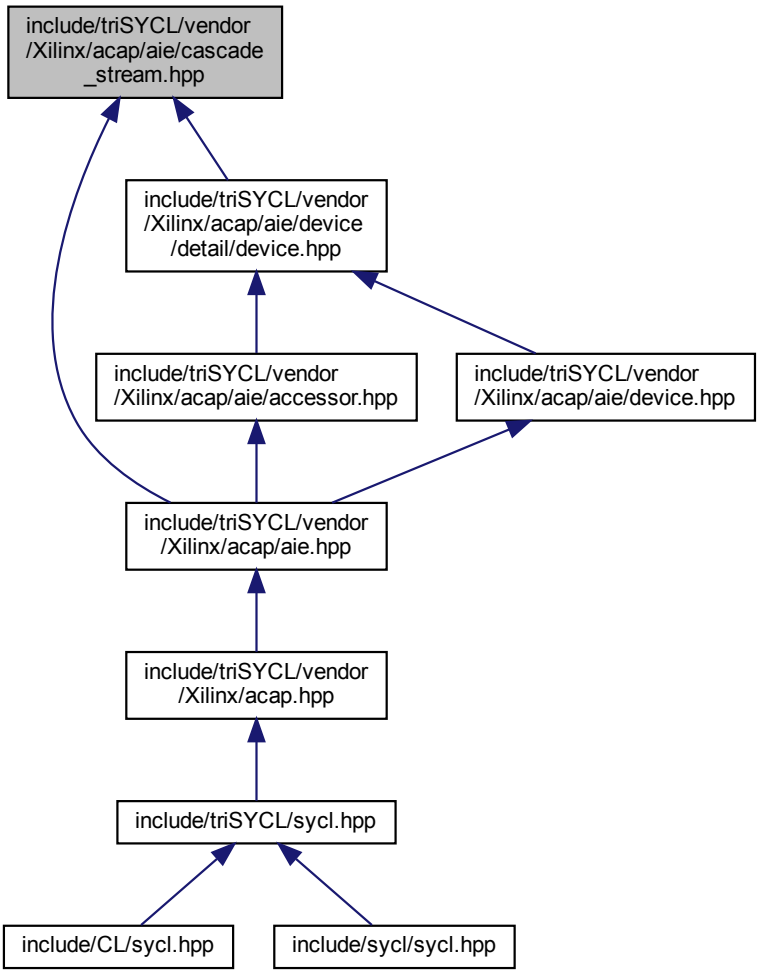
The cascade stream infrastructure between AI Engine tiles.

```
#include "triSYCL/access.hpp"
#include "geography.hpp"
```

Include dependency graph for cascade_stream.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >

  Implementation of the cascade stream infrastructure between AI Engine tiles. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.259.1 Detailed Description

The cascade stream infrastructure between AI Engine tiles.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file cascade_stream.hpp.

13.260 cascade_stream.hpp

Go to the documentation of this file.

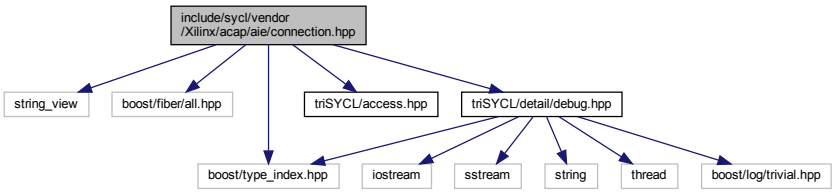
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_CASCADE_STREAM_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_CASCADE_STREAM_HPP
00003
00004 /** \file
00005
00006 The cascade stream infrastructure between AI Engine tiles
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include "triSYCL/access.hpp"
00015 #include "geography.hpp"
00016
00017 namespace trisycl::vendor::xilinx::acap::aie {
00018
00019 /// \ingroup aie
00020 /// @{
00021
00022 /** Implementation of the cascade stream infrastructure between AI
00023 Engine tiles
00024
00025 Based on Math Engine (ME) Architecture Specification, Revision v2.01
00026
00027 March 2019
00028
00029 3.5.4 Data Movement using Cascade Streams, p.  93
00030
00031 2.14 Device-level Floorplanning Guidelines, 2.14.2 Array Edges, p.  69
00032
00033 4.4 ME Core Interfaces, 4.4.4 Cascade Stream Interface, p.  120
00034 explains there is a FIFO depth of 4
00035
00036 5.9.3.6 Cascade Streams:  explains that there are only blocking
00037 accesses
00038
00039 Direct stream interface:  One cascade stream in, one cascade stream
00040 out (384-bits)
00041
00042 \todo Remove this global infrastructure and distribute it on the
00043 tile_infrastructure itself
00044 */
00045 template <typename Geography>
00046 struct cascade_stream {
00047   using geo = Geography;
00048 /** The pipes for the cascade streams, with 1 spare pipe on each
00049 side of the PE string
00050
00051 \todo Use a data type with 384 bits
00052 */
00053   ::trisycl::sycl_2_2::static_pipe<int, 4>
00054   cascade_stream_pipes[geo::x_size*geo::y_size + 1];
00055
00056   /* Cascade stream layout
00057
00058 On even rows, a tile use cascade_stream_pipes[y][x] as input and
00059 cascade_stream_pipes[y][x + 1] as output
00060
00061 On odd rows the flow goes into the other direction, so a tile
00062 use cascade_stream_pipes[y][x + 1] as input and
00063 cascade_stream_pipes[y][x] as output
```

```
00064 */
00065
00066 /** Get a blocking read accessor to the cascade stream input
00067
00068 \param T is the data type used to read from the cascade
00069 stream pipe
00070
00071 \param[in] x is the horizontal tile coordinate
00072
00073 \param[in] y is the vertical tile coordinate
00074 */
00075   template <typename T>
00076   auto get_cascade_stream_in(int x, int y)const {
00077     return cascade_stream_pipes[geo::cascade_linear_id(x, y)]
00078       .template get_access<access::mode::read,
00079                            access::target::blocking_pipe>();
00080   }
00081
00082
00083 /** Get a blocking write accessor to the cascade stream output
00084
00085 \param T is the data type used to write to the cascade
00086 stream pipe
00087
00088 \param[in] x is the horizontal tile coordinate
00089
00090 \param[in] y is the vertical tile coordinate
00091 */
00092   template <typename T>
00093   auto get_cascade_stream_out(int x, int y)const {
00094     // The output is connected to the down-stream neighbour of the cascade
00095     return cascade_stream_pipes[geo::cascade_linear_id(x, y) + 1]
00096       .template get_access<access::mode::write,
00097                            access::target::blocking_pipe>();
00098   }
00099
00100 };
00101
00102 /// @} End the aie Doxygen group
00103
00104 }
00105
00106 /*
00107 # Some Emacs stuff:
00108 ### Local Variables:
00109 ### ispell-local-dictionary:  "american"
00110 ### eval:  (flyspell-prog-mode)
00111 ### End:
00112 */
00113
00114 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_CASCADE_STREAM_HPP
```

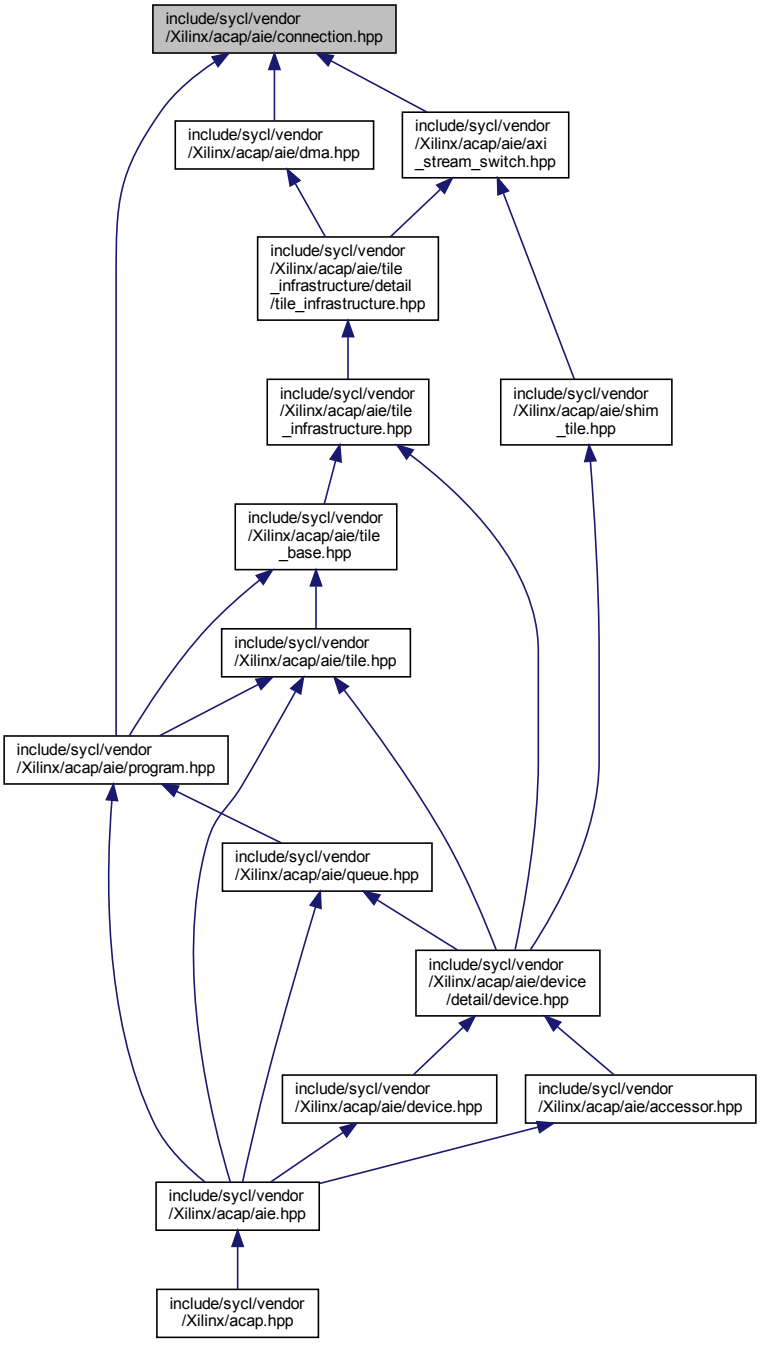
13.261 include/sycl/vendor/Xilinx/acap/aie/connection.hpp File Reference

```
#include <string_view>
#include <boost/fiber/all.hpp>
#include <boost/type_index.hpp>
#include "triSYCL/access.hpp"
#include "triSYCL/detail/debug.hpp"
```

Include dependency graph for connection.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::axi_packet
- struct trisycl::vendor::xilinx::acap::aie::axi_packet::shutdown_t

  Marker type used to signal network shutdown.
- class trisycl::vendor::xilinx::acap::aie::communicator_port

  Abstract interface for a communication port.

- class trisycl::vendor::xilinx::acap::aie::fifo_channel

  A FIFO channel connection.
- class trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >

  A router input port directing to an AIE core input or a BLI input.
- struct trisycl::vendor::xilinx::acap::aie::port

  Abstraction of a communication port.
- struct trisycl::vendor::xilinx::acap::aie::port::shim

  Abstraction of a communication port inside a shim tile.
- struct trisycl::vendor::xilinx::acap::aie::port::tile

  Abstraction of a communication port inside a tile.
- struct trisycl::vendor::xilinx::acap::aie::connection

  A connection between some AXI Stream Switch.
- struct trisycl::vendor::xilinx::acap::aie::connection::base

  Base class of connection port.
- struct trisycl::vendor::xilinx::acap::aie::connection::input

  Abstract an input port.
- struct trisycl::vendor::xilinx::acap::aie::connection::output

  Abstract an output port.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.262 connection.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_CONNECTION_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_CONNECTION_HPP
00003
00004 /** \file Simple connections between AIE AXI stream switch and inside
00005 AXI stream switch
00006
00007 \todo Deprecate \c connection since now we have a more precise routing
00008 involving the AXI stream switches so we should instead use the
00009 same data types.
00010
00011 Ronan dot Keryell at Xilinx dot com
00012
00013 This file is distributed under the University of Illinois Open Source
00014 License.  See LICENSE.TXT for details.
00015 */
00016
00017 #include <string_view>
00018
00019 #include <boost/fiber/all.hpp>
00020 #include <boost/type_index.hpp>
00021
00022 #include "triSYCL/access.hpp"
00023 #include "triSYCL/detail/debug.hpp"
00024
00025 namespace trisycl::vendor::xilinx::acap::aie {
00026
00027 /// \ingroup aie
00028 /// @{
00029
00030 /// Packet moving in AIE network-on-chip (NoC)
```

```
00031 class axi_packet {
00032 /// Marker type used to signal network shutdown
00033   struct shutdown_t {};
00034
00035 public:
00036
00037 /// Payload data type
00038   using value_type = std::uint32_t;
00039
00040 /// Marker value used to construct a packet asking for shutdown
00041   static const shutdown_t shutdown;
00042
00043 /// Network payload
00044   value_type data;
00045
00046 /// Signal end of packet stream
00047   bool tlast = false;
00048
00049 /// Signal a router shutdown
00050   bool shutdown_request = false;
00051
00052 /// Implicit constructor from a data value
00053   axi_packet(const value_type & data) :  data { data } {}
00054
00055 /// Construct a shutdown request packet from axi_packet::shutdown
00056   axi_packet(shutdown_t) :  shutdown_request { true } {}
00057
00058   axi_packet() = default;
00059 };
00060
00061
00062 /** Abstract interface for a communication port
00063
00064 For example a router output is actually implemented as an input
00065 to some other consumer (router, core...).
00066 */
00067 class communicator_port :  ::trisycl::detail::debug<communicator_port> {
00068
00069 public:
00070
00071 /// Enqueue a packet on the communicator input
00072   void virtual write(const axi_packet&) = 0;
00073
00074
00075 /// Alias to enqueue a packet on the communicator input
00076   auto& operator«(const axi_packet::value_type& v) {
00077     write(v);
00078     return *this;
00079   }
00080
00081
00082 /** Try to enqueue a packet to input
00083
00084 \return true if the packet is correctly enqueued
00085 */
00086   bool virtual try_write(const axi_packet&) = 0;
00087
00088
00089 /// Waiting read to a core input port
00090   axi_packet::value_type virtual read() = 0;
00091
00092
00093 /** Non-blocking read from a communicator output port
00094
00095 \return true if the value was correctly read
00096 */
00097   bool virtual try_read(axi_packet::value_type&) = 0;
00098
00099
00100 /// Alias to the waiting read to a communicator output port
00101   template <typename T>
00102   auto& operator»(T& v) {
00103     v = static_cast<T>(read());
00104     return *this;
00105   }
00106
00107
00108 /// Nothing specific to do but need a virtual destructor to avoid slicing
00109   virtual ~communicator_port() = default;
00110 };
00111
00112
00113 /// A FIFO channel connection
00114 /// \todo factorize out router minion
00115 class fifo_channel :  public communicator_port {
00116 /// FIFO capacity queue
00117 /// \todo check with hardware team for the value
```

```
00118   auto static constexpr capacity = 16;
00119
00120 /// Payload data type
00121   using value_type = axi_packet::value_type;
00122
00123 /// The underlying FIFO
00124   boost::fibers::buffered_channel<axi_packet> c { capacity };
00125
00126 public:
00127
00128 /// Enqueue a packet to the FIFO channel
00129   void write(const axi_packet &v)override {
00130     c.push(v);
00131   }
00132
00133
00134 /** Try to enqueue a packet to the FIFO channel
00135
00136 \return true if the packet is correctly enqueued
00137 */
00138   bool try_write(const axi_packet &v)override {
00139     return c.try_push(v) == boost::fibers::channel_op_status::success;
00140   }
00141
00142
00143 /// Waiting read from the FIFO channel
00144   value_type read()override {
00145     return c.value_pop().data;
00146   }
00147
00148
00149 /** Non-blocking read from the FIFO channel
00150
00151 \return true if the value was correctly read
00152 */
00153   bool try_read(value_type &v)override {
00154     axi_packet p;
00155     if (c.try_pop(p) == boost::fibers::channel_op_status::success) {
00156       v = p.data;
00157       return true;
00158     }
00159     return false;
00160   }
00161
00162 };
00163
00164
00165 /// A router input port directing to an AIE core input or a BLI input
00166 /// \todo factorize with fifo_port
00167 template <typename AXIStreamSwich>
00168 class port_receiver :  public communicator_port {
00169 /// Router ingress capacity queue
00170 /// \todo check with hardware team for the value
00171   auto static constexpr capacity = 2;
00172
00173 /// Payload data type
00174   using value_type = axi_packet::value_type;
00175
00176   /* boost::fibers::unbuffered_channel has no try_push() function, so
00177 use a buffered version for now
00178
00179 \todo open a GitHub issue on Boost.Fiber
00180 */
00181   boost::fibers::buffered_channel<axi_packet> c { capacity };
00182
00183 /// Keep track of the AXI stream switch owning this port for debugging
00184   AXIStreamSwich& axi_ss;
00185
00186 /// The component model during debug
00187   std::string_view component_name;
00188
00189 public:
00190
00191   port_receiver(AXIStreamSwich& axi_ss, std::string_view component_name)
00192     :  axi_ss { axi_ss }, component_name { component_name } {}
00193
00194
00195 /// Enqueue a packet (coming from the switch) to the core input
00196   void write(const axi_packet &v)override {
00197     TRISYCL_DUMP_T(component_name « ' ' « this « " on tile("
00198                    « axi_ss.x_coordinate
00199                    « ',' « axi_ss.y_coordinate
00200                    « ") on fiber " « boost::this_fiber::get_id()
00201                    « " write data value " « v.data
00202                    « " to buffered_channel " « &c);
00203     c.push(v);
00204   }
```

```
00205
00206
00207 /** Try to enqueue a packet to the core input
00208
00209 \return true if the packet is correctly enqueued
00210 */
00211   bool try_write(const axi_packet &v)override {
00212     return c.try_push(v) == boost::fibers::channel_op_status::success;
00213   }
00214
00215
00216 /// Waiting read by a tile program on a core input port from the switch
00217   value_type read()override {
00218     TRISYCL_DUMP_T(component_name « ' ' « this « " on tile("
00219                    « axi_ss.x_coordinate
00220                    « ',' « axi_ss.y_coordinate
00221                    « ") on fiber " « boost::this_fiber::get_id()
00222                    « " reading from buffered_channel " « &c « "...");
00223     return c.value_pop().data;
00224   }
00225
00226
00227 /** Non-blocking read to a core input port
00228
00229 \return true if the value was correctly read
00230 */
00231   bool try_read(value_type &v)override {
00232     axi_packet p;
00233
00234     if (c.try_pop(p) == boost::fibers::channel_op_status::success) {
00235       v = p.data;
00236       return true;
00237     }
00238     return false;
00239   }
00240
00241 };
00242
00243
00244 /// Abstraction of a communication port
00245 struct port {
00246 /// Abstraction of a communication port inside a shim tile
00247   struct shim {
00248     const int x;
00249     const int port;
00250   };
00251 /// Abstraction of a communication port inside a tile
00252   struct tile {
00253     const int x;
00254     const int y;
00255     const int port;
00256   };
00257 };
00258
00259 /// A connection between some AXI Stream Switch
00260 struct connection {
00261 /// The concrete communication media with some type erasure
00262   std::shared_ptr<std::any> p;
00263
00264 /** Create a connection
00265
00266 \param[in] pipe is the communication media to be used
00267 */
00268   template <typename Pipe>
00269   connection(Pipe && pipe)
00270     :  p { new std::any { std::forward<Pipe>(pipe) } } {}
00271
00272
00273 /// Get the input port of this connection
00274   auto in() {
00275     return input { *this };
00276   }
00277
00278
00279 /// Get the output port of this connection
00280   auto out() {
00281     return output { *this };
00282   }
00283
00284
00285 /// Base class of connection port
00286   struct base {
00287 /// Keep track of the communication media
00288     std::shared_ptr<std::any> p;
00289
00290
00291 /// Construct a port from a connection
```

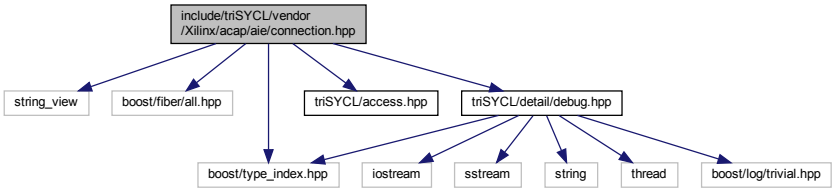
```
00292     base(const connection &c)
00293       :  p { c.p } {}
00294
00295     base() = default;
00296
00297
00298 /** Access the type erased pipe as type T
00299
00300 \param[in] T is the type of the data transmitted by the
00301 connection
00302
00303 \throws trisycl::runtime_error if the connection has not the
00304 correct type
00305 */
00306   template <typename T>
00307     auto pipe_of() {
00308       try {
00309         return std::any_cast<::trisycl::sycl_2_2::static_pipe<T, 4»(*p);
00310       } catch (std::bad_any_cast &) {
00311         throw ::trisycl::runtime_error {
00312           "The current connection is not of type "s
00313           + boost::typeindex::type_id<T>().pretty_name() };
00314       }
00315     }
00316
00317   };
00318
00319
00320 /// Abstract an input port
00321   struct input :  base {
00322 /// Recycle the base constructors
00323     using base::base;
00324
00325 /** Get the input port
00326
00327 \param[in] InputT is the data type to be used in the transfers
00328
00329 \param[in] Target specifies if the connection is blocking or
00330 not
00331
00332 \throws trisycl::runtime_error if the input is not connected
00333 */
00334     template <typename InputT, access::target Target>
00335     auto in() {
00336       if (!p)
00337         throw ::trisycl::runtime_error { "This input is not connected" };
00338       return
00339         pipe_of<InputT>().template get_access<access::mode::read, Target>();
00340     }
00341   };
00342
00343
00344 /// Abstract an output port
00345   struct output :  base {
00346 /// Recycle the base constructors
00347     using base::base;
00348
00349 /** Get the output port
00350
00351 \param[in] InputT is the data type to be used in the transfers
00352
00353 \param[in] Target specifies if the connection is blocking or
00354 not
00355
00356 \throws trisycl::runtime_error if the output is not connected
00357 */
00358     template <typename OutputT, access::target Target>
00359     auto out() {
00360       if (!p)
00361         throw ::trisycl::runtime_error { "This output is not connected" };
00362       return
00363         pipe_of<OutputT>().template get_access<access::mode::write, Target>();
00364     }
00365   };
00366 };
00367
00368 /// @} End the aie Doxygen group
00369
00370 }
00371
00372 /*
00373 # Some Emacs stuff:
00374 ### Local Variables:
00375 ### ispell-local-dictionary:  "american"
00376 ### eval:  (flyspell-prog-mode)
00377 ### End:
00378 */
```

```
00379
00380 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_CONNECTION_HPP
```

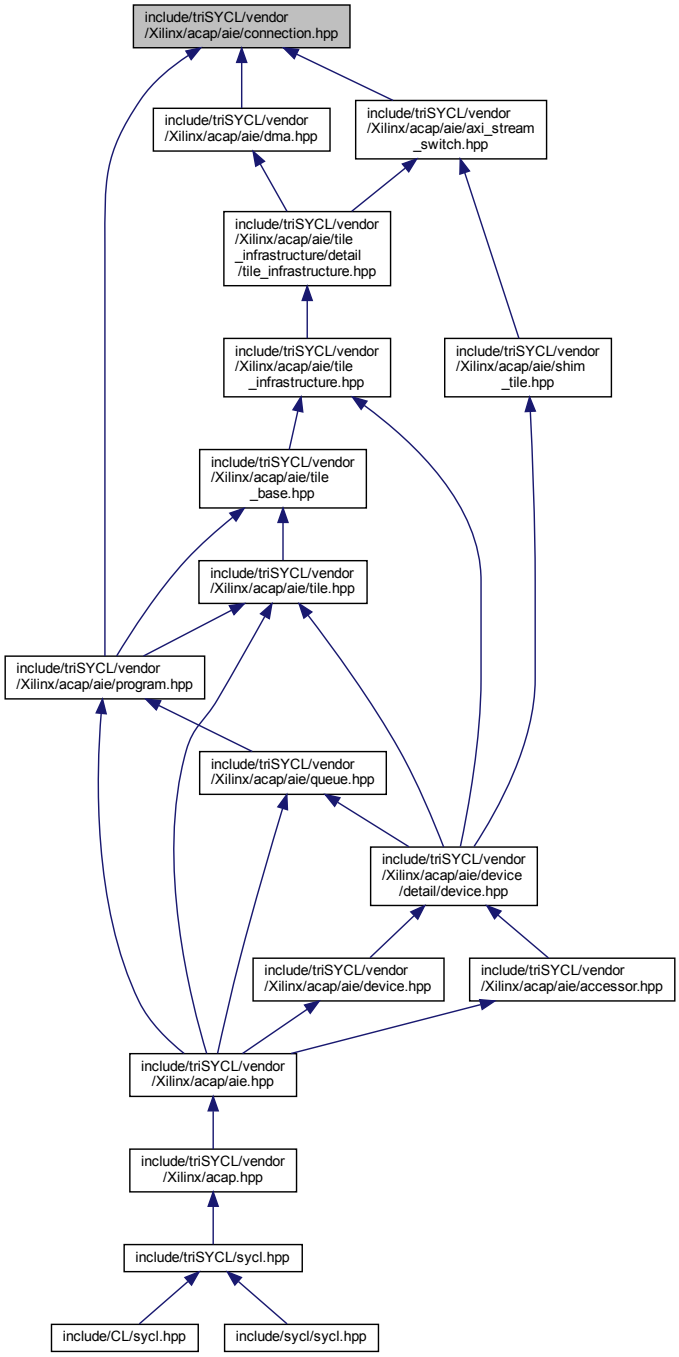
13.263 include/triSYCL/vendor/Xilinx/acap/aie/connection.hpp File Reference

```
#include <string_view>
#include <boost/fiber/all.hpp>
#include <boost/type_index.hpp>
#include "triSYCL/access.hpp"
#include "triSYCL/detail/debug.hpp"
```

Include dependency graph for connection.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::axi_packet
- struct trisycl::vendor::xilinx::acap::aie::axi_packet::shutdown_t

  Marker type used to signal network shutdown.
- class trisycl::vendor::xilinx::acap::aie::communicator_port

  Abstract interface for a communication port.

- class trisycl::vendor::xilinx::acap::aie::fifo_channel

  A FIFO channel connection.
- class trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >

  A router input port directing to an AIE core input or a BLI input.
- struct trisycl::vendor::xilinx::acap::aie::port

  Abstraction of a communication port.
- struct trisycl::vendor::xilinx::acap::aie::port::shim

  Abstraction of a communication port inside a shim tile.
- struct trisycl::vendor::xilinx::acap::aie::port::tile

  Abstraction of a communication port inside a tile.
- struct trisycl::vendor::xilinx::acap::aie::connection

  A connection between some AXI Stream Switch.
- struct trisycl::vendor::xilinx::acap::aie::connection::base

  Base class of connection port.
- struct trisycl::vendor::xilinx::acap::aie::connection::input

  Abstract an input port.
- struct trisycl::vendor::xilinx::acap::aie::connection::output

  Abstract an output port.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.264 connection.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_CONNECTION_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_CONNECTION_HPP
00003
00004 /** \file Simple connections between AIE AXI stream switch and inside
00005 AXI stream switch
00006
00007 \todo Deprecate \c connection since now we have a more precise routing
00008 involving the AXI stream switches so we should instead use the
00009 same data types.
00010
00011 Ronan dot Keryell at Xilinx dot com
00012
00013 This file is distributed under the University of Illinois Open Source
00014 License.  See LICENSE.TXT for details.
00015 */
00016
00017 #include <string_view>
00018
00019 #include <boost/fiber/all.hpp>
00020 #include <boost/type_index.hpp>
00021
00022 #include "triSYCL/access.hpp"
00023 #include "triSYCL/detail/debug.hpp"
00024
00025 namespace trisycl::vendor::xilinx::acap::aie {
00026
00027 /// \ingroup aie
00028 /// @{
00029
00030 /// Packet moving in AIE network-on-chip (NoC)
```

```
00031 class axi_packet {
00032 /// Marker type used to signal network shutdown
00033   struct shutdown_t {};
00034
00035 public:
00036
00037 /// Payload data type
00038   using value_type = std::uint32_t;
00039
00040 /// Marker value used to construct a packet asking for shutdown
00041   static const shutdown_t shutdown;
00042
00043 /// Network payload
00044   value_type data;
00045
00046 /// Signal end of packet stream
00047   bool tlast = false;
00048
00049 /// Signal a router shutdown
00050   bool shutdown_request = false;
00051
00052 /// Implicit constructor from a data value
00053   axi_packet(const value_type & data) :  data { data } {}
00054
00055 /// Construct a shutdown request packet from axi_packet::shutdown
00056   axi_packet(shutdown_t) :  shutdown_request { true } {}
00057
00058   axi_packet() = default;
00059 };
00060
00061
00062 /** Abstract interface for a communication port
00063
00064 For example a router output is actually implemented as an input
00065 to some other consumer (router, core...).
00066 */
00067 class communicator_port :  ::trisycl::detail::debug<communicator_port> {
00068
00069 public:
00070
00071 /// Enqueue a packet on the communicator input
00072   void virtual write(const axi_packet&) = 0;
00073
00074
00075 /// Alias to enqueue a packet on the communicator input
00076   auto& operator«(const axi_packet::value_type& v) {
00077     write(v);
00078     return *this;
00079   }
00080
00081
00082 /** Try to enqueue a packet to input
00083
00084 \return true if the packet is correctly enqueued
00085 */
00086   bool virtual try_write(const axi_packet&) = 0;
00087
00088
00089 /// Waiting read to a core input port
00090   axi_packet::value_type virtual read() = 0;
00091
00092
00093 /** Non-blocking read from a communicator output port
00094
00095 \return true if the value was correctly read
00096 */
00097   bool virtual try_read(axi_packet::value_type&) = 0;
00098
00099
00100 /// Alias to the waiting read to a communicator output port
00101   template <typename T>
00102   auto& operator»(T& v) {
00103     v = static_cast<T>(read());
00104     return *this;
00105   }
00106
00107
00108 /// Nothing specific to do but need a virtual destructor to avoid slicing
00109   virtual ~communicator_port() = default;
00110 };
00111
00112
00113 /// A FIFO channel connection
00114 /// \todo factorize out router minion
00115 class fifo_channel :  public communicator_port {
00116 /// FIFO capacity queue
00117 /// \todo check with hardware team for the value
```

```
00118   auto static constexpr capacity = 16;
00119
00120 /// Payload data type
00121   using value_type = axi_packet::value_type;
00122
00123 /// The underlying FIFO
00124   boost::fibers::buffered_channel<axi_packet> c { capacity };
00125
00126 public:
00127
00128 /// Enqueue a packet to the FIFO channel
00129   void write(const axi_packet &v)override {
00130     c.push(v);
00131   }
00132
00133
00134 /** Try to enqueue a packet to the FIFO channel
00135
00136 \return true if the packet is correctly enqueued
00137 */
00138   bool try_write(const axi_packet &v)override {
00139     return c.try_push(v) == boost::fibers::channel_op_status::success;
00140   }
00141
00142
00143 /// Waiting read from the FIFO channel
00144   value_type read()override {
00145     return c.value_pop().data;
00146   }
00147
00148
00149 /** Non-blocking read from the FIFO channel
00150
00151 \return true if the value was correctly read
00152 */
00153   bool try_read(value_type &v)override {
00154     axi_packet p;
00155     if (c.try_pop(p) == boost::fibers::channel_op_status::success) {
00156       v = p.data;
00157       return true;
00158     }
00159     return false;
00160   }
00161
00162 };
00163
00164
00165 /// A router input port directing to an AIE core input or a BLI input
00166 /// \todo factorize with fifo_port
00167 template <typename AXIStreamSwich>
00168 class port_receiver :  public communicator_port {
00169 /// Router ingress capacity queue
00170 /// \todo check with hardware team for the value
00171   auto static constexpr capacity = 2;
00172
00173 /// Payload data type
00174   using value_type = axi_packet::value_type;
00175
00176   /* boost::fibers::unbuffered_channel has no try_push() function, so
00177 use a buffered version for now
00178
00179 \todo open a GitHub issue on Boost.Fiber
00180 */
00181   boost::fibers::buffered_channel<axi_packet> c { capacity };
00182
00183 /// Keep track of the AXI stream switch owning this port for debugging
00184   AXIStreamSwich& axi_ss;
00185
00186 /// The component model during debug
00187   std::string_view component_name;
00188
00189 public:
00190
00191   port_receiver(AXIStreamSwich& axi_ss, std::string_view component_name)
00192     :  axi_ss { axi_ss }, component_name { component_name } {}
00193
00194
00195 /// Enqueue a packet (coming from the switch) to the core input
00196   void write(const axi_packet &v)override {
00197     TRISYCL_DUMP_T(component_name « ' ' « this « " on tile("
00198                    « axi_ss.x_coordinate
00199                    « ',' « axi_ss.y_coordinate
00200                    « ") on fiber " « boost::this_fiber::get_id()
00201                    « " write data value " « v.data
00202                    « " to buffered_channel " « &c);
00203     c.push(v);
00204   }
```

```
00205
00206
00207 /** Try to enqueue a packet to the core input
00208
00209 \return true if the packet is correctly enqueued
00210 */
00211   bool try_write(const axi_packet &v)override {
00212     return c.try_push(v) == boost::fibers::channel_op_status::success;
00213   }
00214
00215
00216 /// Waiting read by a tile program on a core input port from the switch
00217   value_type read()override {
00218     TRISYCL_DUMP_T(component_name « ' ' « this « " on tile("
00219                    « axi_ss.x_coordinate
00220                    « ',' « axi_ss.y_coordinate
00221                    « ") on fiber " « boost::this_fiber::get_id()
00222                    « " reading from buffered_channel " « &c « "...");
00223     return c.value_pop().data;
00224   }
00225
00226
00227 /** Non-blocking read to a core input port
00228
00229 \return true if the value was correctly read
00230 */
00231   bool try_read(value_type &v)override {
00232     axi_packet p;
00233
00234     if (c.try_pop(p) == boost::fibers::channel_op_status::success) {
00235       v = p.data;
00236       return true;
00237     }
00238     return false;
00239   }
00240
00241 };
00242
00243
00244 /// Abstraction of a communication port
00245 struct port {
00246 /// Abstraction of a communication port inside a shim tile
00247   struct shim {
00248     const int x;
00249     const int port;
00250   };
00251 /// Abstraction of a communication port inside a tile
00252   struct tile {
00253     const int x;
00254     const int y;
00255     const int port;
00256   };
00257 };
00258
00259 /// A connection between some AXI Stream Switch
00260 struct connection {
00261 /// The concrete communication media with some type erasure
00262   std::shared_ptr<std::any> p;
00263
00264 /** Create a connection
00265
00266 \param[in] pipe is the communication media to be used
00267 */
00268   template <typename Pipe>
00269   connection(Pipe && pipe)
00270     :  p { new std::any { std::forward<Pipe>(pipe) } } {}
00271
00272
00273 /// Get the input port of this connection
00274   auto in() {
00275     return input { *this };
00276   }
00277
00278
00279 /// Get the output port of this connection
00280   auto out() {
00281     return output { *this };
00282   }
00283
00284
00285 /// Base class of connection port
00286   struct base {
00287 /// Keep track of the communication media
00288     std::shared_ptr<std::any> p;
00289
00290
00291 /// Construct a port from a connection
```

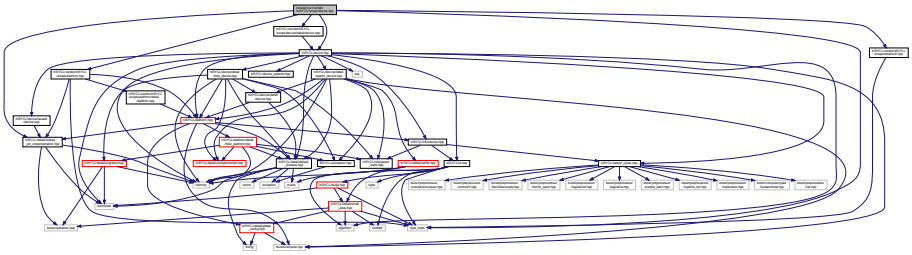
```
00292     base(const connection &c)
00293       :  p { c.p } {}
00294
00295     base() = default;
00296
00297
00298 /** Access the type erased pipe as type T
00299
00300 \param[in] T is the type of the data transmitted by the
00301 connection
00302
00303 \throws trisycl::runtime_error if the connection has not the
00304 correct type
00305 */
00306   template <typename T>
00307     auto pipe_of() {
00308       try {
00309         return std::any_cast<::trisycl::sycl_2_2::static_pipe<T, 4»(*p);
00310       } catch (std::bad_any_cast &) {
00311         throw ::trisycl::runtime_error {
00312           "The current connection is not of type "s
00313           + boost::typeindex::type_id<T>().pretty_name() };
00314       }
00315     }
00316
00317   };
00318
00319
00320 /// Abstract an input port
00321   struct input :  base {
00322 /// Recycle the base constructors
00323     using base::base;
00324
00325 /** Get the input port
00326
00327 \param[in] InputT is the data type to be used in the transfers
00328
00329 \param[in] Target specifies if the connection is blocking or
00330 not
00331
00332 \throws trisycl::runtime_error if the input is not connected
00333 */
00334     template <typename InputT, access::target Target>
00335     auto in() {
00336       if (!p)
00337         throw ::trisycl::runtime_error { "This input is not connected" };
00338       return
00339         pipe_of<InputT>().template get_access<access::mode::read, Target>();
00340     }
00341   };
00342
00343
00344 /// Abstract an output port
00345   struct output :  base {
00346 /// Recycle the base constructors
00347     using base::base;
00348
00349 /** Get the output port
00350
00351 \param[in] InputT is the data type to be used in the transfers
00352
00353 \param[in] Target specifies if the connection is blocking or
00354 not
00355
00356 \throws trisycl::runtime_error if the output is not connected
00357 */
00358     template <typename OutputT, access::target Target>
00359     auto out() {
00360       if (!p)
00361         throw ::trisycl::runtime_error { "This output is not connected" };
00362       return
00363         pipe_of<OutputT>().template get_access<access::mode::write, Target>();
00364     }
00365   };
00366 };
00367
00368 /// @} End the aie Doxygen group
00369
00370 }
00371
00372 /*
00373 # Some Emacs stuff:
00374 ### Local Variables:
00375 ### ispell-local-dictionary:  "american"
00376 ### eval:  (flyspell-prog-mode)
00377 ### End:
00378 */
```

```
00379
00380 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_CONNECTION_HPP
```

13.265 include/sycl/vendor/triSYCL/scope/device.hpp File Reference

```
#include <type_traits>
#include "triSYCL/detail/shared_ptr_implementation.hpp"
#include "triSYCL/device.hpp"
#include "triSYCL/vendor/triSYCL/scope/detail/util.hpp"
#include "triSYCL/vendor/triSYCL/scope/device/detail/device.hpp"
#include "triSYCL/vendor/triSYCL/scope/platform.hpp"
```

Include dependency graph for device.hpp:



Classes

- struct trisycl::vendor::trisycl::scope::empty_device_scope

  An empty device scope to make compilation errors clearer. More...
- class trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >

  A conceptual device implementing some device-scoped storage and based on a platform with platform-scoped storage. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope

Macros

- #define SYCL_VENDOR_TRISYCL_DEVICE_SCOPE 1

  This is an extension providing scope storage for devices.

13.265.1 Macro Definition Documentation

13.265.1.1 SYCL_VENDOR_TRISYCL_DEVICE_SCOPE

```
#define SYCL_VENDOR_TRISYCL_DEVICE_SCOPE 1
```

This is an extension providing scope storage for devices.

Definition at line 29 of file device.hpp.

13.266 device.hpp

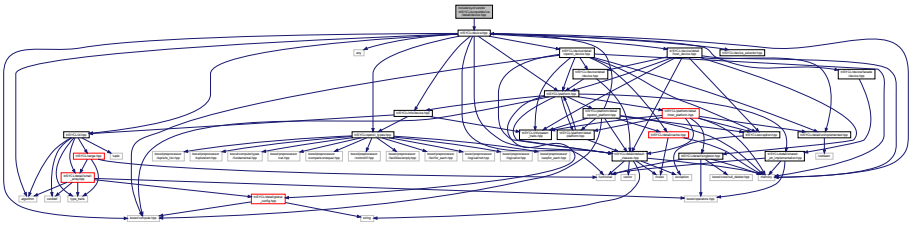
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DEVICE_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DEVICE_HPP
00003
00004 /** \file The SYCL conceptual device with a device scope storage
00005
00006
00007 \todo Add some caching or prevent multiple creation of storage on
00008 the device ?  Or is this a feature to have different scopes on
00009 the same device?
00010
00011 \todo The device and platform are very similar.  Use a common
00012 mix-in to implement both of them?
00013
00014 Ronan at Keryell point FR
00015
00016 This file is distributed under the University of Illinois Open Source
00017 License.  See LICENSE.TXT for details.
00018 */
00019
00020 #include <type_traits>
00021
00022 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00023 #include "triSYCL/device.hpp"
00024 #include "triSYCL/vendor/triSYCL/scope/detail/util.hpp"
00025 #include "triSYCL/vendor/triSYCL/scope/device/detail/device.hpp"
00026 #include "triSYCL/vendor/triSYCL/scope/platform.hpp"
00027
00028 /// This is an extension providing scope storage for devices
00029 #define SYCL_VENDOR_TRISYCL_DEVICE_SCOPE 1
00030
00031 namespace trisycl::vendor::trisycl::scope {
00032
00033 /** \addtogroup vendor_trisycl_scope triSYCL extension for storage scopes
00034 @{
00035 */
00036
00037 /// An empty device scope to make compilation errors clearer
00038 struct empty_device_scope {
00039 };
00040
00041
00042 /** A conceptual device implementing some device-scoped storage and
00043 based on a platform with platform-scoped storage
00044 */
00045 template <typename DeviceStorage = empty_device_scope
00046           , typename ScopedPlatform =
00047           ::trisycl::vendor::trisycl::scope::empty_platform_scope
00048           >
00049 class device
00050   /* Use the underlying device implementation that can be shared in the
00051 SYCL model */
00052   :  public
00053     ::trisycl::detail::shared_ptr_implementation<device<DeviceStorage,
00054                                                         ScopedPlatform>,
00055                                                  detail::device
00056                                                  <DeviceStorage,
00057                                                   ScopedPlatform» {
00058   using spi =
00059     ::trisycl::detail::shared_ptr_implementation<device<DeviceStorage,
00060                                                         ScopedPlatform>,
00061                                                  detail::device
00062                                                  <DeviceStorage,
00063                                                   ScopedPlatform»;
00064
00065   // Allows the comparison operation to access the implementation
00066   friend spi;
```

```
00067
00068 public:
00069
00070   // Make the implementation member directly accessible in this class
00071   using spi::implementation;
00072
00073 /// The type of the scoped storage in the device
00074   using storage_type = DeviceStorage;
00075
00076 /// The device scope storage type
00077   using device_scope_type = storage_type;
00078
00079 /// The platform type targeted by this queue
00080   using platform_type = ScopedPlatform;
00081
00082 /// The platform scope storage type
00083   using platform_scope_type = detail::storage_type_trait_t<platform_type>;
00084
00085 /// True if there is some storage in the device
00086   static constexpr bool has_some_storage_p =
00087     not std::is_same_v<DeviceStorage, empty_device_scope>;
00088
00089
00090 /** Construct the device with some device-scoped storage on top
00091 of a SYCL device
00092
00093 \param[in] d is the real device to use
00094 */
00095   device(const ::trisycl::device &d,
00096          const ScopedPlatform &p = {}) :
00097     spi {
00098       new detail::device<DeviceStorage, ScopedPlatform> { d, p }
00099     } {}
00100
00101
00102 /// By default use the default (host) device with no platform storage
00103   device() :  device(::trisycl::device {}) {}
00104
00105
00106 /// It is still copyable
00107   device(const device&) = default;
00108
00109
00110 /// It is still movable
00111   device(device&&) = default;
00112
00113
00114 /// Access to the device-scoped storage
00115   auto& get_storage()const {
00116     return implementation->get_storage();
00117   }
00118
00119
00120 /// Access to the underlying scoped platform
00121   auto& get_platform()const {
00122     return implementation->get_platform();
00123   }
00124
00125
00126 /** Add a conversion to \c trisycl::device& so it can be used as
00127 a normal device */
00128     operator ::trisycl::device&() const {
00129       return implementation->get_underlying_device();
00130     }
00131
00132 };
00133
00134 /// @} to end the vendor_trisycl_scope Doxygen group
00135
00136 }
00137
00138
00139 /*
00140 # Some Emacs stuff:
00141 ### Local Variables:
00142 ### ispell-local-dictionary:  "american"
00143 ### eval:  (flyspell-prog-mode)
00144 ### End:
00145 */
00146
00147 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DEVICE_HPP
```

13.267 include/sycl/vendor/triSYCL/scope/device/detail/device.hpp File Reference

`#include "triSYCL/device.hpp"`
Include dependency graph for device.hpp:



Classes

- class trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >

  A conceptual device implementing some device-scoped storage and based on a platform with platform-scoped storage. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope
- namespace trisycl::vendor::trisycl::scope::detail

13.268 device.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DEVICE_DETAIL_DEVICE_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DEVICE_DETAIL_DEVICE_HPP
00003
00004 /** \file Implementation details of the SYCL conceptual device with a
00005 device scope storage
00006
00007 \todo The device and platform are very similar.  Use a common
00008 mix-in to implement both of them?
00009
00010 Ronan at Keryell point FR
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include "triSYCL/device.hpp"
00017
00018 namespace trisycl::vendor::trisycl::scope::detail {
00019
00020 /** \addtogroup vendor_trisycl_scope triSYCL extension for storage scopes
00021 @{
00022 */
00023
00024 /** A conceptual device implementing some device-scoped storage and
00025 based on a platform with platform-scoped storage
00026 */
```

```
00027 template <typename DeviceStorage
00028           , typename ScopedPlatform>
00029 class device {
00030 /// The storage-less device behind the scene
00031   ::trisycl::device d;
00032
00033 /** The device-scoped storage default-initialized
00034
00035 \todo For now it is allocated on the host in this CPU emulation
00036 but a device compiler and runtime can create this on a real
00037 device
00038 */
00039   DeviceStorage scope_storage;
00040
00041 /// The scoped platform the scoped device is built into
00042   ScopedPlatform platform_with_scope;
00043
00044
00045 public:
00046
00047 /** Construct the device with some device-scoped storage on top
00048 of a SYCL device
00049
00050 \param[in] d is the real device to use
00051 */
00052   device(const ::trisycl::device &d,
00053          const ScopedPlatform &p)
00054     :  d { d }, platform_with_scope { p } {}
00055
00056
00057 /** Construct the device with some device-scoped storage on top
00058 of a SYCL device
00059
00060 \param[in] d is the real device to use
00061 */
00062   device(const ::trisycl::device &d) :  d { d } {}
00063
00064 /// Get the device behind the curtain
00065   auto& get_underlying_device() {
00066     return d;
00067   }
00068
00069
00070 /// Access to the device-scoped storage
00071   auto& get_storage() {
00072     return scope_storage;
00073   }
00074
00075
00076 /// Access to the underlying scoped platform
00077   auto& get_platform() {
00078     return platform_with_scope;
00079   }
00080
00081 };
00082
00083 /// @} to end the vendor_trisycl_scope Doxygen group
00084
00085 }
00086
00087
00088 /*
00089 # Some Emacs stuff:
00090 ### Local Variables:
00091 ### ispell-local-dictionary:  "american"
00092 ### eval:  (flyspell-prog-mode)
00093 ### End:
00094 */
00095
00096 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DEVICE_DETAIL_DEVICE_HPP
```

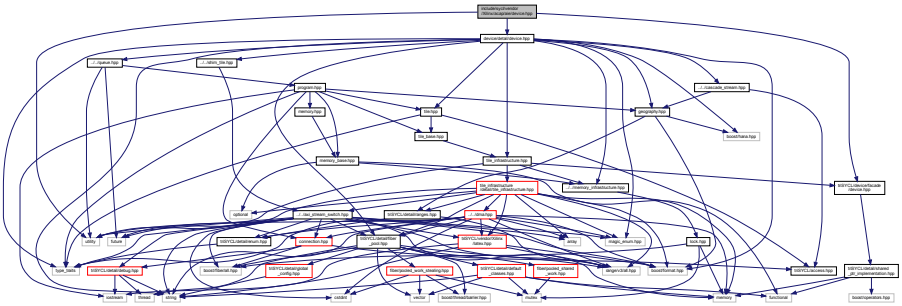
13.269 include/sycl/vendor/Xilinx/acap/aie/device.hpp File Reference

SYCL-friendly device view of AIE.

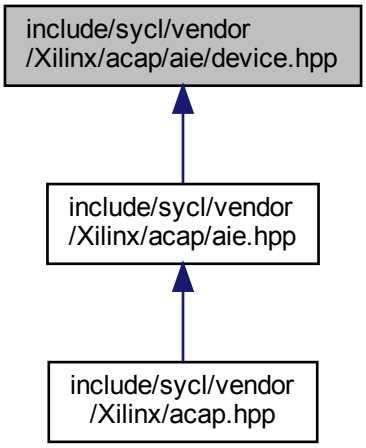
```
#include <utility>
#include "device/detail/device.hpp"
```

```
#include "triSYCL/device/facade/device.hpp"
```

Include dependency graph for device.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::device< Layout >

  Create a SYCL-like device view of an AI Engine CGRA with some layout.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.269.1 Detailed Description

SYCL-friendly device view of AIE.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file device.hpp.

13.270 device.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DEVICE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DEVICE_HPP
00003
00004 /** \file
00005
00006 SYCL-friendly device view of AIE
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include <utility>
00015
00016 #include "device/detail/device.hpp"
00017 #include "triSYCL/device/facade/device.hpp"
00018
00019 /// \ingroup aie
00020 /// @{
00021
00022 namespace trisycl::vendor::xilinx::acap::aie {
00023
00024 /** Create a SYCL-like device view of an AI Engine CGRA with some layout
00025
00026 \param Layout is the layout description of the machine to
00027 instantiate with the physical size
00028 */
00029 template <typename Layout>
00030 class device :  public facade::device<device<Layout>, detail::device<Layout>> {
00031 /// The type encapsulating the implementation
00032   using dd = detail::device<Layout>;
00033
00034 /// The façade used to implement part of the use facing type
00035   using facade_t = facade::device<device<Layout>, dd>;
00036
00037 /// Make the implementation member directly accessible in this class
00038   using facade_t::implementation;
00039
00040  public:
00041 /// Expose some useful internal implementation.
00042 /// Look at them for their documentation
00043   using geo = typename dd::geo;
00044   using cass = typename dd::cass;
00045   using csp = typename dd::csp;
00046   using cmp = typename dd::cmp;
00047   using sass = typename dd::sass;
00048   using ssp = typename dd::ssp;
00049   using smp = typename dd::smp;
00050
00051   // Some shortcuts to useful constants
00052
00053 /// Minimum horizontal coordinate
00054   static auto constexpr x_min = geo::x_min;
00055
00056 /// Maximum horizontal coordinate
00057   static auto constexpr x_max = geo::x_max;
00058
00059 /// Number of tile horizontally in the CGRA
00060   static auto constexpr x_size = geo::x_size;
00061
00062 /// Minimum vertical coordinate
00063   static auto constexpr y_min = geo::y_min;
```

```
00064
00065 /// Maximum vertical coordinate
00066   static auto constexpr y_max = geo::y_max;
00067
00068 /// Number of tile vertically in the CGRA
00069   static auto constexpr y_size = geo::y_size;
00070
00071 /// Total number of tiles in the CGRA
00072   static auto constexpr size = geo::size;
00073
00074 /// The default constructor makes a new device
00075   device()
00076       :  facade_t { std::make_shared<dd>() } {}
00077
00078 /** Apply a function for each tile index of the device
00079
00080 \param f is a callable that is to be called like \c f(x,y) for
00081 each tile
00082 */
00083   template <typename F> void for_each_tile_index(F&& f) {
00084     implementation->for_each_tile_index(std::forward<F>(f));
00085   };
00086
00087 /** Apply a function for each tile infrastructure of the device
00088
00089 \param f is a callable that is to be called like \c f(x,y) for
00090 each tile
00091 */
00092   template <typename F> void for_each_tile(F&& f) {
00093     implementation->for_each_tile(std::forward<F>(f));
00094   };
00095
00096 /** Apply a function for each x tile index of the device
00097
00098 \param f is a callable that is to be called like \c f(x) for
00099 each horizontal index value
00100 */
00101   template <typename F> void for_each_tile_x_index(F&& f) {
00102     implementation->for_each_tile_x_index(std::forward<F>(f));
00103   };
00104
00105 /** Apply a function for each y tile index of the device
00106
00107 \param f is a callable that is to be called like \c f(x) for
00108 each vertical index value
00109 */
00110   template <typename F> void for_each_tile_y_index(F&& f) {
00111     implementation->for_each_tile_y_index(std::forward<F>(f));
00112   };
00113
00114 /** Access to the common infrastructure part of a tile
00115
00116 \param[in] x is the horizontal tile coordinate
00117
00118 \param[in] y is the vertical tile coordinate
00119
00120 \throws trisycl::runtime_error if the coordinate is invalid
00121 */
00122   auto& tile(int x, int y) { return implementation->tile(x, y); }
00123
00124 /** Access to the common infrastructure part of a tile memory
00125
00126 \param[in] x is the horizontal tile coordinate
00127
00128 \param[in] y is the vertical tile coordinate
00129
00130 \throws trisycl::runtime_error if the coordinate is invalid
00131 */
00132   auto& mem(int x, int y) { return implementation->mem(x, y); }
00133
00134 /** Access to the shim tile
00135
00136 \param[in] x is the horizontal coordinate of the shim tile
00137
00138 \throws trisycl::runtime_error if the coordinate is invalid
00139 */
00140   auto& shim(int x) { return implementation->shim(x); }
00141
00142 /// Access the cascade connections
00143 /// \todo To remove?
00144   auto& cascade() { return implementation->cascade(); }
00145
00146 /// Create a queue on this device
00147   auto queue() { return vendor::xilinx::acap::aie::queue { *this }; }
00148
00149 /** Create a program execution for this device
00150
```

```
00151 \param Tile is the description of the program tiles to
00152 instantiate.  By default each tile will run an empty program.
00153
00154 \param Memory is the description of the machine memory modules.  By
00155 default the machine has empty memory modules.
00156 */
00157   template <template <typename Device, int X, int Y>
00158             typename Tile = acap::aie::tile,
00159             template <typename Device, int X, int Y>
00160             typename Memory = acap::aie::memory>
00161   auto program() {
00162     return aie::program<device, Tile, Memory> { *this };
00163   }
00164
00165 /** Shortcut to run synchronously a program on this queue
00166
00167 \param Tile is the description of the program tiles to
00168 instantiate.  By default each tile will run an empty program.
00169
00170 \param Memory is the description of the machine memory modules.  By
00171 default the machine has empty memory modules.
00172 */
00173   template <template <typename Device, int X, int Y>
00174             typename Tile = acap::aie::tile,
00175             template <typename Device, int X, int Y>
00176             typename Memory = acap::aie::memory>
00177   void run() {
00178     queue().template run<Tile, Memory>();
00179   }
00180
00181 /** Shortcut to run synchronously an heterogeneous invocable on this
00182 queue
00183
00184 \param f is an invocable taking an heterogeneous tile handler
00185 */
00186   template <typename Invocable> void run(Invocable&& f) {
00187     queue().run(std::forward<Invocable>(f));
00188   }
00189
00190 /** Shortcut to run synchronously a uniform invocable on
00191 this queue
00192
00193 \param f is an invocable taking a uniform tile handler
00194 */
00195   template <typename Invocable> void uniform_run(Invocable&& f) {
00196     queue().uniform_run(std::forward<Invocable>(f));
00197     queue().wait();
00198   }
00199
00200 /** Connect the ports of 2 tiles or shims together with an
00201 hyperspace switched circuit, jumping over the underlying routing
00202 infrastructure.
00203
00204 This is used to speed up communication in emulation and to skip
00205 the need of a routing planner.
00206
00207 \todo To deprecate and implement in another way?
00208
00209 \param[in] T is the type of the data to be transferred
00210
00211 \param[in] SrcPort is the type of the source port, such as
00212 port::tile or port::shim
00213
00214 \param[in] DstPort is the type of the destination port, such as
00215 port::tile or port::shim
00216
00217 \throws trisycl::runtime_error if some coordinates or port
00218 numbers are invalid
00219
00220 \todo Refactor, make the difference between user & physical ports
00221 */
00222   template <typename SrcPort, typename DstPort>
00223   void connect(SrcPort&& src, DstPort&& dst) {
00224     implementation->connect(std::forward<SrcPort>(src),
00225                             std::forward<DstPort>(dst));
00226   }
00227
00228 /** Apply a invocable on all the AXI stream of the neighborhood of
00229 each tile */
00230   template <typename F> void for_each_tile_neighborhood(F&& f) {
00231     implementation->for_each_tile_neighborhood(std::forward<F>(f));
00232   }
00233
00234 /** Display the device layout
00235
00236 \param[in] file_name is the name of the file to write the LaTeX
00237 drawing into or use std::cerr by default
```

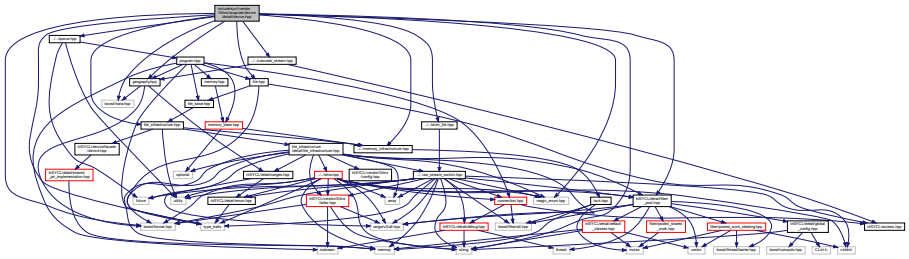
```
00238 */
00239   auto display(const std::string& file_name = {}) {
00240     return implementation->display(file_name);
00241   }
00242 };
00243
00244 /// @} End the aie Doxygen group
00245
00246 } // namespace trisycl::vendor::xilinx::acap::aie
00247
00248 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DEVICE_HPP
```

13.271 include/sycl/vendor/Xilinx/acap/aie/device/detail/device.hpp File Reference

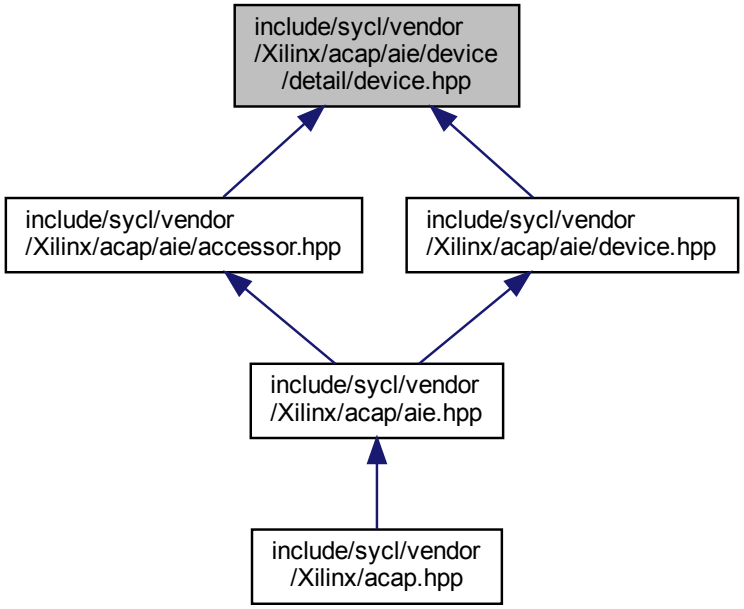
SYCL-friendly device view of AIE.

```
#include <string>
#include <type_traits>
#include "magic_enum.hpp"
#include <boost/format.hpp>
#include <boost/hana.hpp>
#include "../../cascade_stream.hpp"
#include "../../geography.hpp"
#include "../../memory_infrastructure.hpp"
#include "../../queue.hpp"
#include "../../shim_tile.hpp"
#include "../../tile.hpp"
#include "../../tile_infrastructure.hpp"
#include "triSYCL/detail/fiber_pool.hpp"
```

Include dependency graph for device.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::detail::device< Layout >

  Create a SYCL-like device view of an AI Engine CGRA with some layout.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::detail

13.271.1 Detailed Description

SYCL-friendly device view of AIE.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file device.hpp.

13.272 device.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DEVICE_DETAIL_DEVICE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DEVICE_DETAIL_DEVICE_HPP
00003
00004 /** \file
00005
00006 SYCL-friendly device view of AIE
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include <string>
00015 #include <type_traits>
00016
00017 #include "magic_enum.hpp"
00018 #include <boost/format.hpp>
00019 #include <boost/hana.hpp>
00020
00021 #include "../../cascade_stream.hpp"
00022 #include "../../geography.hpp"
00023 #include "../../memory_infrastructure.hpp"
00024 #include "../../queue.hpp"
00025 #include "../../shim_tile.hpp"
00026 #include "../../tile.hpp"
00027 #include "../../tile_infrastructure.hpp"
00028
00029 #include "triSYCL/detail/fiber_pool.hpp"
00030
00031 /// \ingroup aie
00032 /// @{
00033
00034 namespace trisycl::vendor::xilinx::acap::aie::detail {
00035
00036 /** Create a SYCL-like device view of an AI Engine CGRA with some layout
00037
00038 \param Layout is the layout description of the machine to
00039 instantiate with the physical size
00040 */
00041 template <typename Layout> struct device {
00042
00043 /// The geography of the CGRA
00044   using geo = geography<Layout>;
00045
00046 /// Naming shortcut for the core AXI stream switch
00047   using cass = typename geo::core_axi_stream_switch;
00048
00049 /// Naming shortcut for the slave ports of the core AXI stream switch
00050   using csp = typename cass::slave_port_layout;
00051
00052 /// Naming shortcut for the master ports of the core AXI stream switch
00053   using cmp = typename cass::master_port_layout;
00054
00055 /// Naming shortcut for the shim AXI stream switch
00056   using sass = typename geo::shim_axi_stream_switch;
00057
00058 /// Naming shortcut for the slave ports of the shim AXI stream switch
00059   using ssp = typename sass::slave_port_layout;
00060
00061 /// Naming shortcut for the master ports of the shim AXI stream switch
00062   using smp = typename sass::master_port_layout;
00063
00064 /// The cascade stream infrastructure of the CGRA
00065 /// \todo remove and distribute it instead
00066   cascade_stream<geo> cs;
00067
00068 /** A fiber pool executor to run the infrastructure powered by
00069 TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS std::thread */
00070   ::trisycl::detail::fiber_pool fiber_executor {
00071     1, ::trisycl::detail::fiber_pool::sched::round_robin, false
00072   };
00073   /* { TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS,
00074 detail::fiber_pool::sched::work_stealing, false };
00075 */
00076
00077 /// Keep track of all the (non detail) infrastructure tiles of this device
00078   aie::tile_infrastructure<geo> ti[geo::y_size][geo::x_size];
00079
00080 /** Access to the common infrastructure part of a tile
00081
00082 \param[in] x is the horizontal tile coordinate
```

```
00083
00084 \param[in] y is the vertical tile coordinate
00085
00086 \throws trisycl::runtime_error if the coordinate is invalid
00087 */
00088   auto& tile(int x, int y) {
00089     geo::validate_x_y(x, y);
00090     return ti[y][x];
00091   }
00092
00093 /** Apply a function for each tile index of the device
00094
00095 \param f is a callable that is to be called like \c f(x,y) for
00096 each tile
00097 */
00098   template <typename F> void for_each_tile_index(F&& f) {
00099     for (auto [x, y] :
00100          ranges::views::cartesian_product(ranges::views::iota(0, geo::x_size),
00101                                           ranges::views::iota(0, geo::y_size)))
00102       f(x, y);
00103   };
00104
00105 /** Apply a function for each tile infrastructure of the device
00106
00107 \param f is a callable that is to be called like \c f(x,y) for
00108 each tile
00109 */
00110   template <typename F> void for_each_tile(F&& f) {
00111     for_each_tile_index([&](auto x, auto y) { f(tile(x, y)); });
00112   };
00113
00114 /** Apply a function for each x tile index of the device
00115
00116 \param f is a callable that is to be called like \c f(x) for
00117 each horizontal index value
00118 */
00119   template <typename F> void for_each_tile_x_index(F&& f) {
00120     for (auto x :  ranges::views::iota(0, geo::x_size))
00121       f(x);
00122   };
00123
00124 /** Apply a function for each y tile index of the device
00125
00126 \param f is a callable that is to be called like \c f(x) for
00127 each vertical index value
00128 */
00129   template <typename F> void for_each_tile_y_index(F&& f) {
00130     for (auto y :  ranges::views::iota(0, geo::y_size))
00131       f(y);
00132   };
00133
00134 /** Shortcut to access to the common infrastructure part of a tile memory
00135
00136 \param[in] x is the horizontal tile coordinate
00137
00138 \param[in] y is the vertical tile coordinate
00139
00140 \throws trisycl::runtime_error if the coordinate is invalid
00141 */
00142   auto& mem(int x, int y) {
00143     return tile(x, y).mem();
00144   }
00145
00146 /** The shim tiles on the lower row of the tile array
00147
00148 For now we consider only homogeneous shim tiles.
00149 */
00150   shim_tile<device> shims[geo::x_size];
00151
00152 /** Access to the shim tile
00153
00154 \param[in] x is the horizontal coordinate of the shim tile
00155
00156 \throws trisycl::runtime_error if the coordinate is invalid
00157 */
00158   auto& shim(int x) {
00159     geo::validate_x(x);
00160     return shims[x];
00161   }
00162
00163 /// Access the cascade connections
00164 /// \todo To remove?
00165   auto& cascade() { return cs; }
00166
00167 /** Connect the ports of 2 tiles or shims together with an
00168 hyperspace switched circuit, jumping over the underlying routing
00169 infrastructure.
```

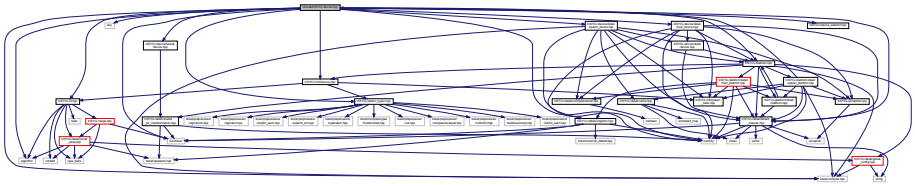
```
00170
00171 This is used to speed up communication in emulation and to skip
00172 the need of a routing planner.
00173
00174 \todo To deprecate and implement in another way?
00175
00176 \param[in] T is the type of the data to be transferred
00177
00178 \param[in] SrcPort is the type of the source port, such as
00179 port::tile or port::shim
00180
00181 \param[in] DstPort is the type of the destination port, such as
00182 port::tile or port::shim
00183
00184 \throws trisycl::runtime_error if some coordinates or port
00185 numbers are invalid
00186
00187 \todo Refactor, make the difference between user & physical ports
00188 */
00189   template <typename SrcPort, typename DstPort>
00190   void connect(SrcPort&& src, DstPort&& dst) {
00191     static_assert(std::is_same_v<SrcPort, port::tile> ||
00192                   std::is_same_v<SrcPort, port::shim>,
00193                   "SrcPort type should be port::tile or port::shim");
00194     static_assert(std::is_same_v<DstPort, port::tile> ||
00195                   std::is_same_v<DstPort, port::shim>,
00196                   "DstPort type should be port::tile or port::shim");
00197     // Get the destination in the final switch
00198     auto channel = [&] {
00199       if constexpr (std::is_same_v<DstPort, port::tile>)
00200         return tile(dst.x, dst.y).in_connection(dst.port);
00201       else if constexpr (std::is_same_v<DstPort, port::shim>)
00202         return shim(dst.x).bli_in_connection(dst.port);
00203     }();
00204     // And connect the source  port to it
00205     if constexpr (std::is_same_v<SrcPort, port::tile>)
00206       tile(src.x, src.y).out_connection(src.port) = channel;
00207     else if constexpr (std::is_same_v<SrcPort, port::shim>)
00208       shim(src.x).bli_out_connection(src.port) = channel;
00209   }
00210
00211 /// Apply a function on all the AXI stream of the neighborhood of each tile
00212   template <typename F> void for_each_tile_neighborhood(F&& f) {
00213     for_each_tile_index([&](auto x, auto y) {
00214       // No CTAD yet with Boost::Hana and Clang++-10 (but works with g++-9)
00215       boost::hana::for_each(
00216           geo::core_axi_stream_switch::interconnect, [&](auto connections) {
00217             auto [dx, dy, master_range, slave_range] = connections;
00218             if (geo::is_x_y_valid(x + dx, y + dy))
00219               for (auto [o, i] :  ranges::views::zip(master_range, slave_range))
00220                 f(x, y, x + dx, y + dy, o, i);
00221           });
00222     });
00223   }
00224
00225 /// Build some device level infrastructure
00226   device() {
00227     // Initialize all the tiles with their network connections first
00228     for_each_tile_index([&](auto x, auto y) {
00229       // Create & start the tile infrastructure
00230       tile(x, y) = { x, y, *this, fiber_executor };
00231     });
00232     for_each_tile_x_index([&](auto x) {
00233       // Start the shim infrastructure
00234       shim(x).start(x, fiber_executor);
00235     });
00236
00237     // Only then we can connect the inter-core tile AXI stream NoC
00238     for_each_tile_neighborhood(
00239         [&](auto x, auto y, auto nx, auto ny, auto m, auto s) {
00240           tile(x, y).output(m) = tile(nx, ny).input(s);
00241         });
00242
00243 /// Make the connections around the shim tiles
00244     for_each_tile_x_index([&](auto x) {
00245       // The connection from shim tile to core tile
00246       for (auto [o, i] :
00247            ranges::views::zip(sass::m_north_range, cass::s_south_range))
00248         shim(x).output(o) = tile(x, 0).input(i);
00249       // The connection from core tile to shim tile
00250       for (auto [o, i] :
00251            ranges::views::zip(cass::m_south_range, sass::s_north_range))
00252         tile(x, 0).output(o) = shim(x).input(i);
00253       if (!geo::is_east_column(x))
00254         for (auto [o, i] :
00255              ranges::views::zip(sass::m_east_range, sass::s_west_range))
00256           shim(x).output(o) = shim(x + 1).input(i);
```

```
00257       if (!geo::is_west_column(x))
00258         for (auto [o, i] :
00259              ranges::views::zip(sass::m_west_range, sass::s_east_range))
00260           shim(x).output(o) = shim(x - 1).input(i);
00261     });
00262   }
00263
00264 /** Display the device layout
00265
00266 \param[in] file_name is the name of the file to write the LaTeX
00267 drawing into or use std::cerr by default
00268 */
00269   auto display(const std::string& file_name = {}) {
00270     // Compute the drawing size starting with the individual tile size
00271     auto tile_size = tile(0, 0).display_size();
00272     // And expanding according to the device size.  Add 1 in each
00273     // dimension to fit in the page because of the borders
00274     auto x_size = tile_size.x() * geo::x_size + 1;
00275     auto y_size = tile_size.y() * geo::y_size + 1;
00276 /// The LaTeX generation is handled by a LaTeX context
00277     latex::context c { { x_size, y_size }, file_name };
00278
00279     for_each_tile_index([&](auto x, auto y) { tile(x, y).display(c); });
00280
00281     // Connect each tile to its neighbors
00282     for_each_tile_neighborhood(
00283         [&](auto x, auto y, auto nx, auto ny, auto m, auto s) {
00284           c.add((boost::format { R"(
00285 \draw (node cs:name=TileX%1%Y%2%M%3%)
00286 -- (node cs:name=TileX%4%Y%5%S%6%);)" } %
00287 x % y % c.clean_node(magic_enum::enum_name(m)) % nx % ny %
00288 c.clean_node(magic_enum::enum_name(s)))
00289 .str());
00290 });
00291
00292 c.display();
00293 }
00294 };
00295
00296 /// @} End the aie Doxygen group
00297
00298 } // namespace trisycl::vendor::xilinx::acap::aie::detail
00299
00300 /*
00301 # Some Emacs stuff:
00302 ### Local Variables:
00303 ### ispell-local-dictionary:  "american"
00304 ### eval:  (flyspell-prog-mode)
00305 ### End:
00306 */
00307
00308 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DEVICE_DETAIL_DEVICE_HPP
```

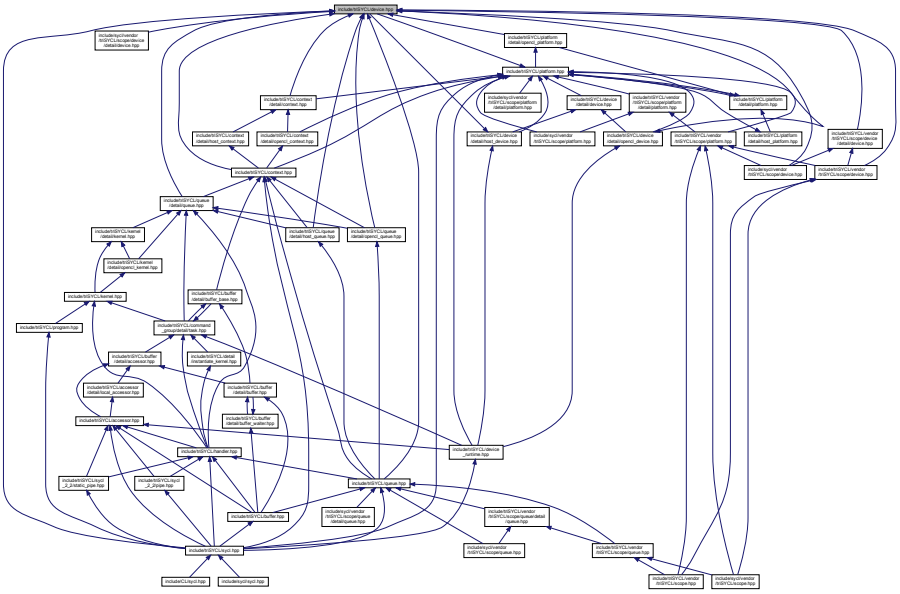
13.273 include/triSYCL/device.hpp File Reference

```
#include <algorithm>
#include <functional>
#include <memory>
#include <any>
#include <boost/compute.hpp>
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/device/facade/device.hpp"
#include "triSYCL/device/detail/host_device.hpp"
#include "triSYCL/device/detail/opencl_device.hpp"
#include "triSYCL/id.hpp"
#include "triSYCL/info/device.hpp"
#include "triSYCL/device_selector.hpp"
#include "triSYCL/opencl_types.hpp"
#include "triSYCL/platform.hpp"
```

Include dependency graph for device.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::device

  SYCL device. More...
- struct std::hash< trisycl::device >

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace std

13.274 device.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DEVICE_HPP
00002 #define TRISYCL_SYCL_DEVICE_HPP
00003
00004 /** \file The OpenCL SYCL device
```

```
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <algorithm>
00013 #include <functional>
00014 #include <memory>
00015 #include <any>
00016
00017 #ifdef TRISYCL_OPENCL
00018 #include <boost/compute.hpp>
00019 #endif
00020
00021 #include "triSYCL/detail/default_classes.hpp"
00022
00023 #include "triSYCL/device/facade/device.hpp"
00024 #include "triSYCL/device/detail/host_device.hpp"
00025 #ifdef TRISYCL_OPENCL
00026 #include "triSYCL/device/detail/opencl_device.hpp"
00027 #endif
00028 #include "triSYCL/id.hpp"
00029 #include "triSYCL/info/device.hpp"
00030 #include "triSYCL/device_selector.hpp"
00031 #include "triSYCL/opencl_types.hpp"
00032 #include "triSYCL/platform.hpp"
00033
00034 namespace trisycl {
00035
00036 class device_selector;
00037 class platform;
00038
00039 /** \addtogroup execution Platforms, contexts, devices and queues
00040 @{
00041 */
00042
00043 /// SYCL device
00044 class device
00045   /* Use the underlying device implementation that can be shared in the
00046 SYCL model */
00047   :  public facade::device<device, detail::device> {
00048
00049   // The type encapsulating the implementation
00050   using implementation_t = facade::device<device, detail::device>;
00051
00052 public:
00053
00054   // Make the implementation member directly accessible in this class
00055   using implementation_t::implementation;
00056
00057 /// The default constructor uses the SYCL host device
00058   device() :  implementation_t { detail::host_device::instance() } {}
00059
00060
00061 #ifdef TRISYCL_OPENCL
00062 /** Construct a device class instance using cl_device_id of the
00063 OpenCL device
00064
00065 Return synchronous errors via the SYCL exception class.
00066
00067 Retain a reference to the OpenCL device and if this device was
00068 an OpenCL subdevice the device should be released by the caller
00069 when it is no longer needed.
00070 */
00071   device(cl_device_id device_id)
00072     :  device { boost::compute::device { device_id } } {}
00073
00074
00075 /** Construct a device class instance using a boost::compute::device
00076
00077 This is a triSYCL extension for boost::compute interoperation.
00078
00079 Return synchronous errors via the SYCL exception class.
00080 */
00081   device(const boost::compute::device &d)
00082     :  implementation_t { detail::opencl_device::instance(d) } {}
00083 #endif
00084
00085
00086 /** Construct a device class instance using the device selector
00087 provided
00088
00089 Return errors via C++ exception class.
00090
00091 \todo Make it non-explicit in the specification?
```

```
00092 */
00093   explicit device(const device_selector &ds) {
00094     auto devices = device::get_devices();
00095     if (devices.empty())
00096       // \todo Put a SYCL exception
00097       throw std::domain_error("No device at all!  Internal error...");
00098
00099     /* Find the device with the best score according to the given
00100 device_selector */
00101     auto max = std::max_element(devices.cbegin(), devices.cend(),
00102                                 [&] (const device &d1, const device &d2) {
00103                                   return ds(d1) < ds(d2);
00104                                 });
00105     if (ds(*max) < 0)
00106       // \todo Put a SYCL exception
00107       throw std::domain_error("No device selected because no positive "
00108                               "device_selector score found");
00109
00110     // Create the current device as a shared copy of the selected one
00111     implementation = max->implementation;
00112   }
00113
00114
00115 #ifdef TRISYCL_OPENCL
00116 /** Return the cl_device_id of the underlying OpenCL platform
00117
00118 Return synchronous errors via the SYCL exception class.
00119
00120 Retain a reference to the returned cl_device_id object.  Caller
00121 should release it when finished.
00122
00123 In the case where this is the SYCL host device it will throw an
00124 exception.
00125 */
00126   cl_device_id get()const {
00127     return implementation->get();
00128   }
00129
00130
00131 /** Return the underlying Boost.Compute device if it is an
00132 OpenCL device
00133
00134 This is a triSYCL extension
00135 */
00136   boost::compute::device get_boost_compute()const {
00137     return implementation->get_boost_compute();
00138   }
00139 #endif
00140
00141
00142 /// Return true if the device is the SYCL host device
00143   bool is_host()const {
00144     return implementation->is_host();
00145   }
00146
00147
00148 /// Return true if the device is an OpenCL CPU device
00149   bool is_cpu()const {
00150     return implementation->is_cpu();
00151   }
00152
00153
00154 /// Return true if the device is an OpenCL GPU device
00155   bool is_gpu()const {
00156     return implementation->is_gpu();
00157   }
00158
00159
00160 /// Return true if the device is an OpenCL accelerator device
00161   bool is_accelerator()const {
00162     return implementation->is_accelerator();
00163   }
00164
00165
00166
00167 /** Return the device_type of a device
00168
00169 \todo Present in Boost.Compute, to be added to the specification
00170 */
00171   info::device_type type()const {
00172     if (is_host())
00173       return info::device_type::host;
00174     else if (is_cpu())
00175       return info::device_type::cpu;
00176     else if (is_gpu())
00177       return info::device_type::gpu;
00178     else if (is_accelerator())
```

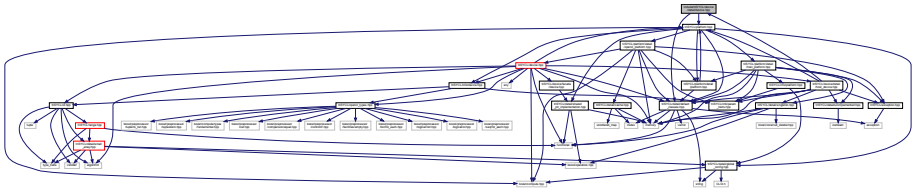
```
00179       return info::device_type::accelerator;
00180     else
00181       // \todo Put a SYCL exception
00182       throw std::domain_error("Unknown trisycl::info::device_type");
00183   }
00184
00185
00186 /** Return the platform of device
00187
00188 Return synchronous errors via the SYCL exception class.
00189 */
00190   platform get_platform()const {
00191     return implementation->get_platform();
00192   }
00193
00194
00195 /** Return a list of all available devices
00196
00197 Return synchronous errors via SYCL exception classes.
00198 */
00199 #ifdef _MSC_VER
00200   inline
00201 #endif
00202   static vector_class<device>
00203     get_devices(info::device_type device_type = info::device_type::all)
00204     TRISYCL_WEAK_ATTRIB_SUFFIX;
00205
00206 /** Query the device for OpenCL info::device info
00207
00208 Return synchronous errors via the SYCL exception class.
00209
00210 \todo
00211 */
00212 /** Query the device for OpenCL info::device info
00213
00214 Return synchronous errors via the SYCL exception class.
00215
00216 \todo
00217 */
00218   template <info::device param>
00219   typename info::param_traits<info::device, param>::return_type
00220   get_info()const {
00221     return std::any_cast<
00222       typename info::param_traits<info::device, param>::return_type
00223       >(implementation->get_info(param));
00224   }
00225
00226 /// Test if a specific extension is supported on the device
00227   bool has_extension(const string_class &extension)const {
00228     return implementation->has_extension(extension);
00229   }
00230
00231   // Available only when prop == info::partition_property::partition_equally
00232   template <info::partition_property prop>
00233   vector_class<device> create_sub_devices(size_t nbSubDev)const
00234 {
00235     throw trisycl::feature_not_supported("unsupported feature\n");
00236   }
00237
00238   // Available only when prop == info::partition_property::partition_by_counts
00239   template <info::partition_property prop>
00240   vector_class<device> create_sub_devices(const vector_class<size_t> &counts)const
00241 {
00242     throw trisycl::feature_not_supported("unsupported feature\n");
00243   }
00244
00245   // Available only when prop == info::partition_property::partition_by_affinity_domain
00246   template <info::partition_property prop>
00247   vector_class<device> create_sub_devices(info::partition_affinity_domain affinityDomain)const
00248 {
00249     throw trisycl::feature_not_supported("unsupported feature\n");
00250   }
00251
00252 };
00253
00254 /// @} to end the Doxygen group
00255
00256 }
00257
00258
00259 /* Inject a custom specialization of std::hash to have the buffer
00260 usable into an unordered associative container
00261
00262 \todo Add this to the spec
00263 */
00264 namespace std {
00265
```

```
00266 template <> struct hash<trisycl::device> {
00267
00268   auto operator()(const trisycl::device &d)const {
00269     // Forward the hashing to the implementation
00270     return d.hash();
00271   }
00272
00273 };
00274
00275 }
00276
00277 /*
00278 # Some Emacs stuff:
00279 ### Local Variables:
00280 ### ispell-local-dictionary:  "american"
00281 ### eval:  (flyspell-prog-mode)
00282 ### End:
00283 */
00284
00285 #endif // TRISYCL_SYCL_DEVICE_HPP
```

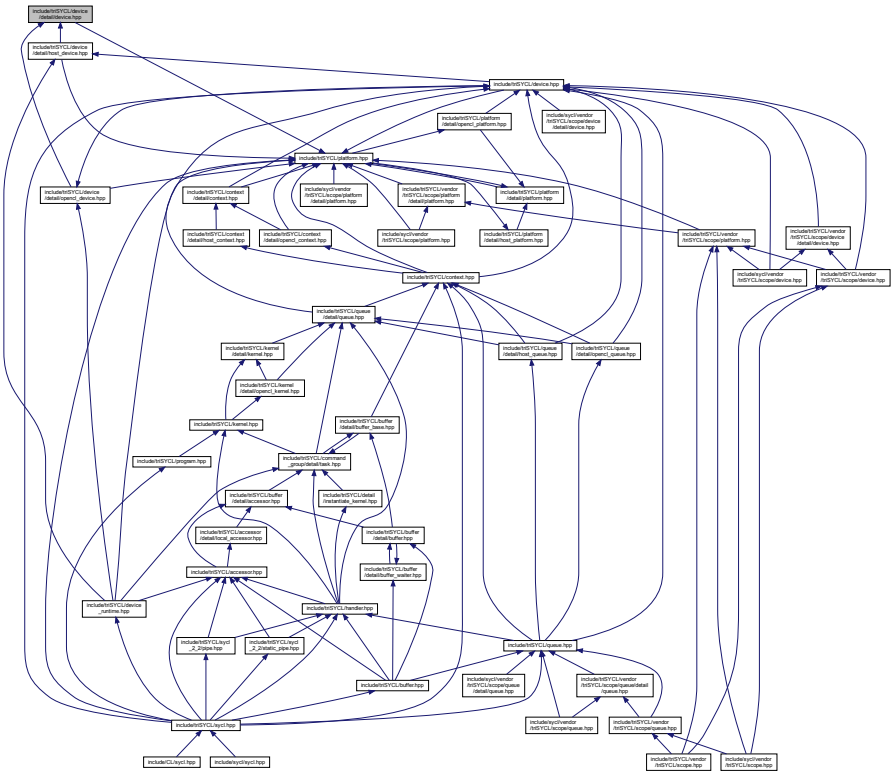
13.275 include/triSYCL/device/detail/device.hpp File Reference

```
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/platform.hpp"
```

Include dependency graph for device.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::device

  An abstract class representing various models of SYCL devices. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.276 device.hpp

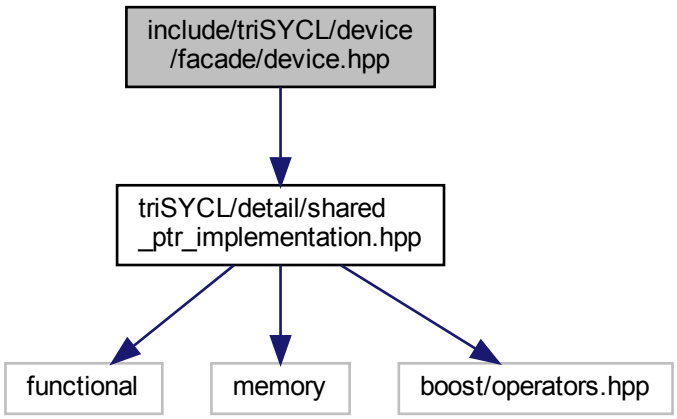
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DEVICE_DETAIL_DEVICE_HPP
00002 #define TRISYCL_SYCL_DEVICE_DETAIL_DEVICE_HPP
00003
00004 /** \file The OpenCL SYCL abstract device
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include "triSYCL/detail/default_classes.hpp"
```

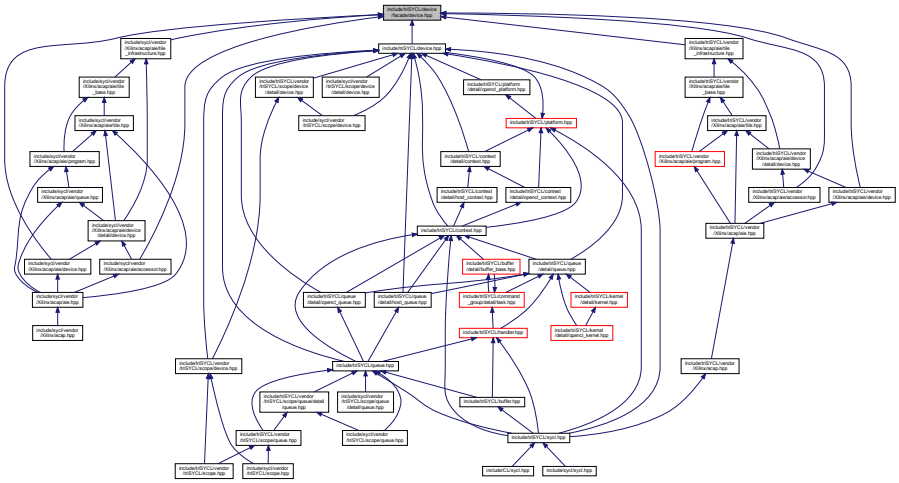
```
00013
00014 #include "triSYCL/platform.hpp"
00015
00016 namespace trisycl::detail {
00017
00018 /** \addtogroup execution Platforms, contexts, devices and queues
00019 @{
00020 */
00021
00022 /// An abstract class representing various models of SYCL devices
00023 class device {
00024
00025 public:
00026
00027 #ifdef TRISYCL_OPENCL
00028 /// Return the cl_device_id of the underlying OpenCL platform
00029   virtual cl_device_id get() const = 0;
00030
00031 /// Return the underlying Boost.Compute device, if any
00032   virtual boost::compute::device &get_boost_compute() = 0;
00033 #endif
00034
00035
00036 /// Return true if the device is a SYCL host device
00037   virtual bool is_host() const = 0;
00038
00039
00040 /// Return true if the device is an OpenCL CPU device
00041   virtual bool is_cpu() const = 0;
00042
00043
00044 /// Return true if the device is an OpenCL GPU device
00045   virtual bool is_gpu() const = 0;
00046
00047
00048 /// Return true if the device is an OpenCL accelerator device
00049   virtual bool is_accelerator() const = 0;
00050
00051
00052 /// Return the platform of device
00053   virtual trisycl::platform get_platform() const = 0;
00054
00055
00056 /// Query the device for OpenCL info::device info
00057   virtual std::any get_info(info::device param) const = 0;
00058
00059
00060 /// Specify whether a specific extension is supported on the device.
00061   virtual bool has_extension(const string_class &extension) const = 0;
00062
00063
00064   // Virtual to call the real destructor
00065   virtual ~device() {}
00066
00067 };
00068
00069 /// @} to end the execution Doxygen group
00070
00071 }
00072
00073 /*
00074 # Some Emacs stuff:
00075 ### Local Variables:
00076 ### ispell-local-dictionary:  "american"
00077 ### eval:  (flyspell-prog-mode)
00078 ### End:
00079 */
00080
00081 #endif // TRISYCL_SYCL_DEVICE_DETAIL_DEVICE_HPP
```

13.277 include/triSYCL/device/facade/device.hpp File Reference

`#include "triSYCL/detail/shared_ptr_implementation.hpp"`
Include dependency graph for device.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::facade::device< UserFacing, ImplementationDetail >

  SYCL device. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::facade

13.278 device.hpp

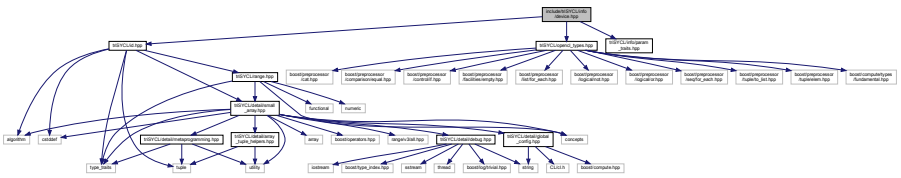
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DEVICE_FACADE_DEVICE_HPP
00002 #define TRISYCL_SYCL_DEVICE_FACADE_DEVICE_HPP
00003
00004 /** \file A SYCL device façade to implement more easily a SYCL
00005 device concept
00006
00007 This file is distributed under the University of Illinois Open Source
00008 License.  See LICENSE.TXT for details.
00009 */
00010
00011 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00012
00013 namespace trisycl::facade {
00014
00015 /** \addtogroup execution Platforms, contexts, devices and queues
00016 @{
00017 */
00018
00019 /// SYCL device
00020 template <typename UserFacing, typename ImplementationDetail>
00021 class device
00022   /* Use the underlying device implementation that can be shared in the
00023 SYCL model */
00024   :  public detail::shared_ptr_implementation<UserFacing, ImplementationDetail> {
00025
00026 public:
00027
00028 /// The type encapsulating the implementation
00029   using implementation_t =
00030     detail::shared_ptr_implementation<UserFacing, ImplementationDetail>;
00031
00032 /// The real detail type doing the work behind the scene
00033   using detail_t = ImplementationDetail;
00034
00035 /// Import the constructors
00036   using implementation_t::implementation_t;
00037
00038 };
00039
00040 /// @} to end the Doxygen group
00041
00042 }
00043
00044 #endif // TRISYCL_SYCL_DEVICE_FACADE_DEVICE_HPP
```

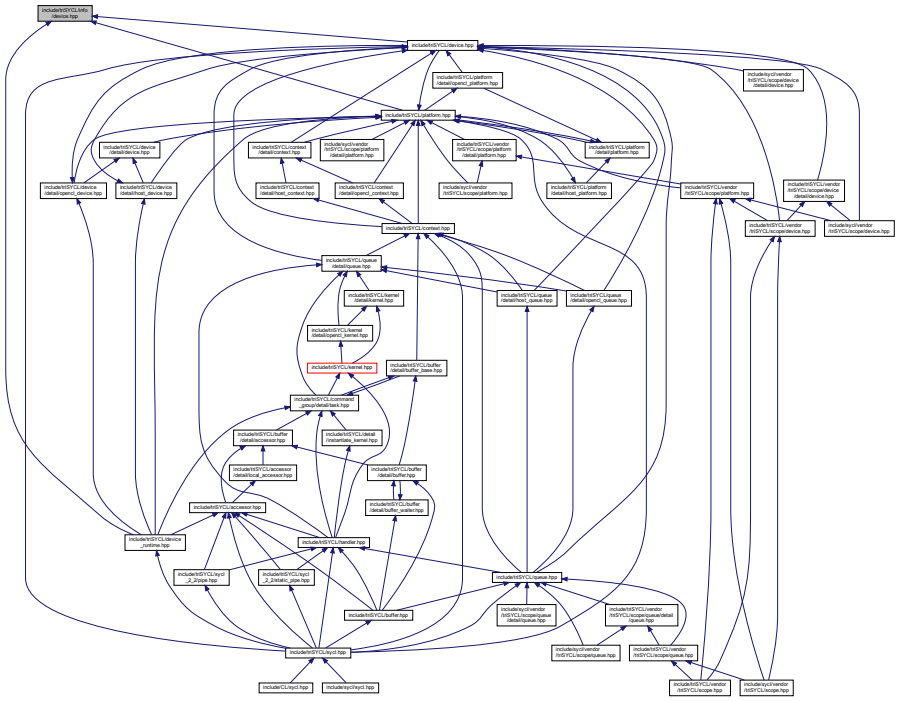
13.279 include/triSYCL/info/device.hpp File Reference

```
#include "triSYCL/id.hpp"
#include "triSYCL/opencl_types.hpp"
#include "triSYCL/info/param_traits.hpp"
```

Include dependency graph for device.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::info

Typedefs

- using trisycl::info::device_fp_config = unsigned int
- using trisycl::info::device_exec_capabilities = unsigned int
- using trisycl::info::device_queue_properties = unsigned int

Enumerations

- enum class trisycl::info::device_type : unsigned int {
  trisycl::info::device_type::cpu , trisycl::info::device_type::gpu , trisycl::info::device_type::accelerator , trisycl::info::device_type::custom ,
  trisycl::info::device_type::automatic , trisycl::info::device_type::host , trisycl::info::device_type::opencl , trisycl::info::device_type::all }

  Type of devices.
- enum class trisycl::info::device : int {
  trisycl::info::device::device_type , trisycl::info::device::vendor_id , trisycl::info::device::max_compute_units ,
  trisycl::info::device::max_work_item_dimensions ,
  trisycl::info::device::max_work_item_sizes , trisycl::info::device::max_work_group_size , trisycl::info::device::preferred_vector_w
  , trisycl::info::device::preferred_vector_width_short ,
  trisycl::info::device::preferred_vector_width_int , trisycl::info::device::preferred_vector_width_long , trisycl::info::device::preferrec

, trisycl::info::device::preferred_vector_width_double , trisycl::info::device::preferred_vector_width_half , trisycl::info::device::native_vector_width_char , trisycl::info::device::native_vec , trisycl::info::device::native_vector_width_int , trisycl::info::device::native_vector_width_long , trisycl::info::device::native_vector_width_float , trisycl::info::device::native_vector , trisycl::info::device::native_vector_width_half , trisycl::info::device::max_clock_frequency , trisycl::info::device::address_bits , trisycl::info::device::max_mem_alloc_size , trisycl::info::device::image_support , trisycl::info::device::max_read_image_args , trisycl::info::device::max_write_image_args , trisycl::info::device::image2d_max_wi , trisycl::info::device::image2d_max_height , trisycl::info::device::image3d_max_width , trisycl::info::device::image3d_max_height , trisycl::info::device::image3d_max_depth , trisycl::info::device::image_max_buffer_size , trisycl::info::device::image_max_array_size , trisycl::info::device::max_samplers , trisycl::info::device::max_parameter_size , trisycl::info::device::mem_base_addr_align , trisycl::info::device::half_fp_config , trisycl::info::device::single_fp_config , trisycl::info::device::double_fp_config , trisycl::info::device::global_mem_cache_type , trisycl::info::device::global_mem_cache_line_size , trisycl::info::device::global_mem_cache_size , trisycl::info::device::global_m , trisycl::info::device::max_constant_buffer_size , trisycl::info::device::max_constant_args , trisycl::info::device::local_mem_type , trisycl::info::device::local_mem_size , trisycl::info::device::error_correction_support , trisycl::info::device::host_unified_memory , trisycl::info::device::profiling_timer_resolution , trisycl::info::device::is_endian_little , trisycl::info::device::is_available , trisycl::info::device::is_compiler_available , trisycl::info::device::is_linker_available , trisycl::info::device::execution_capabilities , trisycl::info::device::queue_profiling , trisycl::info::device::built_in_kernels , trisycl::info::device::platform , trisycl::info::device::name , trisycl::info::device::vendor , trisycl::info::device::driver_version , trisycl::info::device::profile , trisycl::info::device::version , trisycl::info::device::opencl_c_versi , trisycl::info::device::extensions , trisycl::info::device::printf_buffer_size , trisycl::info::device::preferred_interop_user_sync , trisycl::info::device::parent_device , trisycl::info::device::partition_max_sub_devices , trisycl::info::device::partition_properties , trisycl::info::device::partition_affinity_ , trisycl::info::device::partition_type_property , trisycl::info::device::partition_type_affinity_domain , trisycl::info::device::reference_count }

Device information descriptors.

- enum class trisycl::info::partition_property : int { trisycl::info::partition_property::no_partition , trisycl::info::partition_property::par , trisycl::info::partition_property::partition_by_counts , trisycl::info::partition_property::partition_by_affinity_domain }
- enum class trisycl::info::partition_affinity_domain : int { trisycl::info::partition_affinity_domain::not_applicable , trisycl::info::partition_affinity_domain::numa , trisycl::info::partition_affinity_domain::L4_cache , trisycl::info::partition_affinity_domain::L3_cache , trisycl::info::partition_affinity_domain::L2_cache , trisycl::info::partition_affinity_domain::L1_cache , trisycl::info::partition_affinity }
- enum class trisycl::info::local_mem_type : int { trisycl::info::local_mem_type::none , trisycl::info::local_mem_type::local , trisycl::info::local_mem_type::global }
- enum class trisycl::info::fp_config : int { trisycl::info::fp_config::denorm , trisycl::info::fp_config::inf_nan , trisycl::info::fp_config::round_to_nearest , trisycl::info::fp_config::round_to_zero , trisycl::info::fp_config::round_to_inf , trisycl::info::fp_config::fma , trisycl::info::fp_config::correctly_rounded_divide_sqrt , trisycl::info::fp_config::soft_float }
- enum class trisycl::info::global_mem_cache_type : int { trisycl::info::global_mem_cache_type::none , trisycl::info::global_mem_cache_type::read_only , trisycl::info::global_mem_cache_type::write_only }
- enum class trisycl::info::execution_capability : unsigned int { trisycl::info::execution_capability::exec_kernel , trisycl::info::execution_capability::exec_native_kernel }

Functions

- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image2d_max_width, size_t)

Query the return type for get_info() on context stuff.

- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image2d_max_height, size_t)
- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image3d_max_width, size_t)
- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image3d_max_height, size_t)
- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image3d_max_depth, size_t)
- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image_max_buffer_size, size_t)
- trisycl::info::TRISYCL_INFO_PARAM_TRAITS (info::device::image_max_array_size, size_t)

13.280 device.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_INFO_DEVICE_HPP
00002 #define TRISYCL_SYCL_INFO_DEVICE_HPP
00003
00004 /** \file The OpenCL SYCL device information parameters
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include "triSYCL/id.hpp"
00013 #include "triSYCL/opencl_types.hpp"
00014 #include "triSYCL/info/param_traits.hpp"
00015
00016 namespace trisycl {
00017
00018 class device;
00019 class platform;
00020
00021 /** \addtogroup execution Platforms, contexts, devices and queues
00022 @{
00023 */
00024
00025 namespace info {
00026
00027 /** Type of devices
00028
00029 To be used either to define a device type or to select more
00030 broadly a kind of device
00031
00032 \todo To be moved in the specification from platform to device
00033
00034 \todo Add opencl to the specification
00035
00036 \todo there is no accelerator_selector and custom_accelerator
00037 */
00038 enum class device_type :  unsigned int {
00039   cpu,
00040   gpu,
00041   accelerator,
00042   custom,
00043   automatic,
00044   host,
00045   opencl,
00046   all
00047 };
00048
00049
00050 /** Device information descriptors
00051
00052 From specs/latex/headers/deviceInfo.h in the specification
00053
00054 \todo Should be unsigned int?
00055 */
00056 enum class device :  int {
00057   device_type,
00058   vendor_id,
00059   max_compute_units,
00060   max_work_item_dimensions,
00061   max_work_item_sizes,
00062   max_work_group_size,
00063   preferred_vector_width_char,
00064   preferred_vector_width_short,
00065   preferred_vector_width_int,
00066   preferred_vector_width_long,
```

```
00067   preferred_vector_width_float,
00068   preferred_vector_width_double,
00069   preferred_vector_width_half,
00070   native_vector_width_char,
00071   native_vector_width_short,
00072   native_vector_width_int,
00073   native_vector_width_long,
00074   native_vector_width_float,
00075   native_vector_width_double,
00076   native_vector_width_half,
00077   max_clock_frequency,
00078   address_bits,
00079   max_mem_alloc_size,
00080   image_support,
00081   max_read_image_args,
00082   max_write_image_args,
00083   image2d_max_width,
00084   image2d_max_height,
00085   image3d_max_width,
00086   image3d_max_height,
00087   image3d_max_depth,
00088   image_max_buffer_size,
00089   image_max_array_size,
00090   max_samplers,
00091   max_parameter_size,
00092   mem_base_addr_align,
00093   half_fp_config,
00094   single_fp_config,
00095   double_fp_config,
00096   global_mem_cache_type,
00097   global_mem_cache_line_size,
00098   global_mem_cache_size,
00099   global_mem_size,
00100   max_constant_buffer_size,
00101   max_constant_args,
00102   local_mem_type,
00103   local_mem_size,
00104   error_correction_support,
00105   host_unified_memory,
00106   profiling_timer_resolution,
00107   is_endian_little,
00108   is_available,
00109   is_compiler_available,
00110   is_linker_available,
00111   execution_capabilities,
00112   queue_profiling,
00113   built_in_kernels,
00114   platform,
00115   name,
00116   vendor,
00117   driver_version,
00118   profile,
00119   version,
00120   opencl_c_version,
00121   extensions,
00122   printf_buffer_size,
00123   preferred_interop_user_sync,
00124   parent_device,
00125   partition_max_sub_devices,
00126   partition_properties,
00127   partition_affinity_domains,
00128   partition_type_property,
00129   partition_type_affinity_domain,
00130   reference_count
00131 };
00132
00133 enum class partition_property :  int {
00134   no_partition,
00135   partition_equally,
00136   partition_by_counts,
00137   partition_by_affinity_domain,
00138 };
00139
00140 enum class partition_affinity_domain :  int {
00141   not_applicable,
00142   numa,
00143   L4_cache,
00144   L3_cache,
00145   L2_cache,
00146   L1_cache,
00147   next_partitionable
00148 };
00149
00150 enum class local_mem_type :  int {
00151   none,
00152   local,
00153   global
```

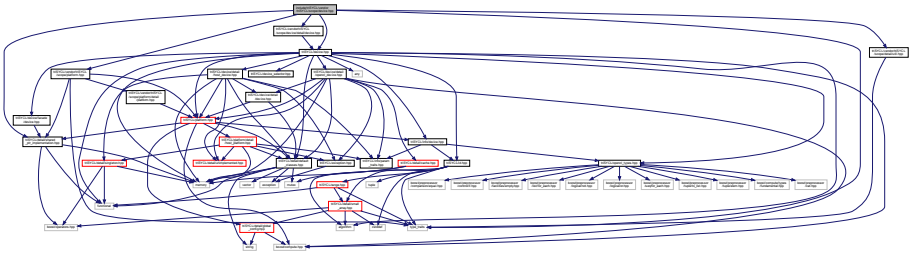
```
00154 };
00155
00156 enum class fp_config :  int {
00157   denorm,
00158   inf_nan,
00159   round_to_nearest,
00160   round_to_zero,
00161   round_to_inf,
00162   fma,
00163   correctly_rounded_divide_sqrt,
00164   soft_float
00165 };
00166
00167 enum class global_mem_cache_type :  int {
00168   none,
00169   read_only,
00170   write_only
00171 };
00172
00173 enum class execution_capability :  unsigned int {
00174   exec_kernel,
00175   exec_native_kernel
00176 };
00177
00178
00179 using device_fp_config = unsigned int;
00180 using device_exec_capabilities = unsigned int;
00181 using device_queue_properties = unsigned int;
00182
00183
00184 /** Query the return type for get_info() on context stuff
00185
00186 \todo To be implemented, return always void.
00187 */
00188 TRISYCL_INFO_PARAM_TRAITS_ANY_T(info::device, void)
00189 TRISYCL_INFO_PARAM_TRAITS(info::device::device_type, info::device_type)
00190 TRISYCL_INFO_PARAM_TRAITS(info::device::vendor_id, trisycl::cl_uint)
00191 TRISYCL_INFO_PARAM_TRAITS(info::device::max_compute_units, trisycl::cl_uint)
00192 TRISYCL_INFO_PARAM_TRAITS(info::device::max_work_item_dimensions, trisycl::cl_uint)
00193 TRISYCL_INFO_PARAM_TRAITS(info::device::max_work_item_sizes, trisycl::id<3>)
00194 TRISYCL_INFO_PARAM_TRAITS(info::device::max_work_group_size, std::size_t)
00195
00196 TRISYCL_INFO_PARAM_TRAITS(info::device::preferred_vector_width_char, trisycl::cl_uint)
00197 TRISYCL_INFO_PARAM_TRAITS(info::device::preferred_vector_width_short, trisycl::cl_uint)
00198 TRISYCL_INFO_PARAM_TRAITS(info::device::preferred_vector_width_int, trisycl::cl_uint)
00199 TRISYCL_INFO_PARAM_TRAITS(info::device::preferred_vector_width_long, trisycl::cl_uint)
00200 TRISYCL_INFO_PARAM_TRAITS(info::device::preferred_vector_width_float, trisycl::cl_uint)
00201 TRISYCL_INFO_PARAM_TRAITS(info::device::preferred_vector_width_double, trisycl::cl_uint)
00202 TRISYCL_INFO_PARAM_TRAITS(info::device::preferred_vector_width_half, trisycl::cl_uint)
00203
00204 TRISYCL_INFO_PARAM_TRAITS(info::device::native_vector_width_char, trisycl::cl_uint)
00205 TRISYCL_INFO_PARAM_TRAITS(info::device::native_vector_width_short, trisycl::cl_uint)
00206 TRISYCL_INFO_PARAM_TRAITS(info::device::native_vector_width_int, trisycl::cl_uint)
00207 TRISYCL_INFO_PARAM_TRAITS(info::device::native_vector_width_long, trisycl::cl_uint)
00208 TRISYCL_INFO_PARAM_TRAITS(info::device::native_vector_width_float, trisycl::cl_uint)
00209 TRISYCL_INFO_PARAM_TRAITS(info::device::native_vector_width_double, trisycl::cl_uint)
00210 TRISYCL_INFO_PARAM_TRAITS(info::device::native_vector_width_half, trisycl::cl_uint)
00211
00212 TRISYCL_INFO_PARAM_TRAITS(info::device::max_clock_frequency, trisycl::cl_uint)
00213 TRISYCL_INFO_PARAM_TRAITS(info::device::address_bits, trisycl::cl_uint)
00214 TRISYCL_INFO_PARAM_TRAITS(info::device::max_mem_alloc_size, trisycl::cl_ulong)
00215 TRISYCL_INFO_PARAM_TRAITS(info::device::image_support, bool)
00216
00217 TRISYCL_INFO_PARAM_TRAITS(info::device::max_read_image_args, trisycl::cl_uint)
00218 TRISYCL_INFO_PARAM_TRAITS(info::device::max_write_image_args, trisycl::cl_uint)
00219 TRISYCL_INFO_PARAM_TRAITS(info::device::image2d_max_width, size_t);
00220 TRISYCL_INFO_PARAM_TRAITS(info::device::image2d_max_height, size_t);
00221 TRISYCL_INFO_PARAM_TRAITS(info::device::image3d_max_width, size_t);
00222 TRISYCL_INFO_PARAM_TRAITS(info::device::image3d_max_height, size_t);
00223 TRISYCL_INFO_PARAM_TRAITS(info::device::image3d_max_depth, size_t);
00224 TRISYCL_INFO_PARAM_TRAITS(info::device::image_max_buffer_size, size_t);
00225 TRISYCL_INFO_PARAM_TRAITS(info::device::image_max_array_size, size_t);
00226 TRISYCL_INFO_PARAM_TRAITS(info::device::max_samplers, trisycl::cl_uint)
00227 TRISYCL_INFO_PARAM_TRAITS(info::device::max_parameter_size, size_t)
00228 TRISYCL_INFO_PARAM_TRAITS(info::device::mem_base_addr_align, trisycl::cl_uint)
00229 TRISYCL_INFO_PARAM_TRAITS(info::device::half_fp_config, vector_class<fp_config>)
00230 TRISYCL_INFO_PARAM_TRAITS(info::device::single_fp_config, vector_class<fp_config>)
00231 TRISYCL_INFO_PARAM_TRAITS(info::device::double_fp_config, vector_class<fp_config>)
00232 TRISYCL_INFO_PARAM_TRAITS(info::device::global_mem_cache_type, global_mem_cache_type)
00233 TRISYCL_INFO_PARAM_TRAITS(info::device::global_mem_cache_line_size, trisycl::cl_uint)
00234 TRISYCL_INFO_PARAM_TRAITS(info::device::global_mem_cache_size, trisycl::cl_ulong)
00235 TRISYCL_INFO_PARAM_TRAITS(info::device::global_mem_size, trisycl::cl_ulong)
00236 TRISYCL_INFO_PARAM_TRAITS(info::device::max_constant_buffer_size, trisycl::cl_ulong)
00237 TRISYCL_INFO_PARAM_TRAITS(info::device::max_constant_args, trisycl::cl_uint)
00238 TRISYCL_INFO_PARAM_TRAITS(info::device::local_mem_type, local_mem_type)
00239 TRISYCL_INFO_PARAM_TRAITS(info::device::local_mem_size, trisycl::cl_ulong)
00240 TRISYCL_INFO_PARAM_TRAITS(info::device::error_correction_support, bool)
```

```
00241 TRISYCL_INFO_PARAM_TRAITS(info::device::host_unified_memory, bool)
00242 TRISYCL_INFO_PARAM_TRAITS(info::device::profiling_timer_resolution, size_t)
00243 TRISYCL_INFO_PARAM_TRAITS(info::device::is_endian_little, bool)
00244 TRISYCL_INFO_PARAM_TRAITS(info::device::is_available, bool)
00245 TRISYCL_INFO_PARAM_TRAITS(info::device::is_compiler_available, bool)
00246 TRISYCL_INFO_PARAM_TRAITS(info::device::is_linker_available, bool)
00247 TRISYCL_INFO_PARAM_TRAITS(info::device::execution_capabilities, vector_class<execution_capability>)
00248 TRISYCL_INFO_PARAM_TRAITS(info::device::queue_profiling, bool)
00249 TRISYCL_INFO_PARAM_TRAITS(info::device::built_in_kernels, vector_class<string_class>)
00250 TRISYCL_INFO_PARAM_TRAITS(info::device::platform, trisycl::platform)
00251 TRISYCL_INFO_PARAM_TRAITS(info::device::name, string_class)
00252 TRISYCL_INFO_PARAM_TRAITS(info::device::vendor, string_class)
00253 TRISYCL_INFO_PARAM_TRAITS(info::device::driver_version, string_class)
00254 TRISYCL_INFO_PARAM_TRAITS(info::device::profile, string_class)
00255 TRISYCL_INFO_PARAM_TRAITS(info::device::version, string_class)
00256 TRISYCL_INFO_PARAM_TRAITS(info::device::opencl_c_version, string_class)
00257 TRISYCL_INFO_PARAM_TRAITS(info::device::extensions, vector_class<string_class>)
00258 TRISYCL_INFO_PARAM_TRAITS(info::device::printf_buffer_size, size_t)
00259 TRISYCL_INFO_PARAM_TRAITS(info::device::preferred_interop_user_sync, bool)
00260 TRISYCL_INFO_PARAM_TRAITS(info::device::parent_device, trisycl::device)
00261 TRISYCL_INFO_PARAM_TRAITS(info::device::partition_max_sub_devices, trisycl::cl_uint)
00262 TRISYCL_INFO_PARAM_TRAITS(info::device::partition_properties, vector_class<partition_property>)
00263 TRISYCL_INFO_PARAM_TRAITS(info::device::partition_affinity_domains,
      vector_class<partition_affinity_domain>)
00264 TRISYCL_INFO_PARAM_TRAITS(info::device::partition_type_property, partition_property)
00265 TRISYCL_INFO_PARAM_TRAITS(info::device::partition_type_affinity_domain, partition_affinity_domain)
00266 TRISYCL_INFO_PARAM_TRAITS(info::device::reference_count, trisycl::cl_uint)
00267 }
00268 }
00269
00270 /*
00271 # Some Emacs stuff:
00272 ### Local Variables:
00273 ### ispell-local-dictionary:  "american"
00274 ### eval:  (flyspell-prog-mode)
00275 ### End:
00276 */
00277
00278 #endif // TRISYCL_SYCL_INFO_DEVICE_HPP
```

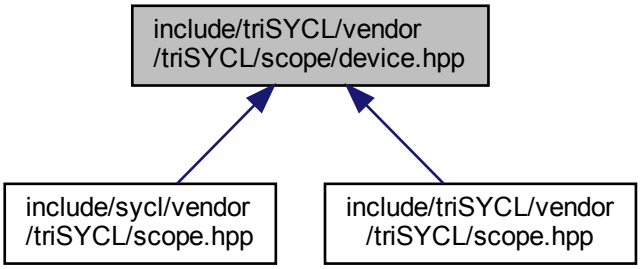
13.281 include/triSYCL/vendor/triSYCL/scope/device.hpp File Reference

```
#include <type_traits>
#include "triSYCL/detail/shared_ptr_implementation.hpp"
#include "triSYCL/device.hpp"
#include "triSYCL/vendor/triSYCL/scope/detail/util.hpp"
#include "triSYCL/vendor/triSYCL/scope/device/detail/device.hpp"
#include "triSYCL/vendor/triSYCL/scope/platform.hpp"
```

Include dependency graph for device.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::trisycl::scope::empty_device_scope

  An empty device scope to make compilation errors clearer. More...
- class trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >

  A conceptual device implementing some device-scoped storage and based on a platform with platform-scoped storage. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope

Macros

- #define SYCL_VENDOR_TRISYCL_DEVICE_SCOPE 1

  This is an extension providing scope storage for devices.

13.281.1 Macro Definition Documentation

13.281.1.1 SYCL_VENDOR_TRISYCL_DEVICE_SCOPE

```
#define SYCL_VENDOR_TRISYCL_DEVICE_SCOPE 1
```

This is an extension providing scope storage for devices.

Definition at line 29 of file device.hpp.

13.282 device.hpp

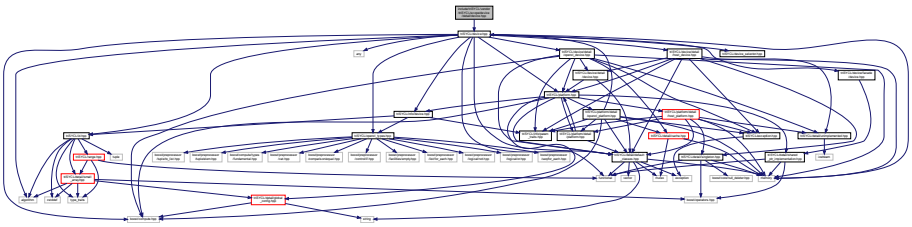
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DEVICE_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DEVICE_HPP
00003
00004 /** \file The SYCL conceptual device with a device scope storage
00005
00006
00007 \todo Add some caching or prevent multiple creation of storage on
00008 the device ?  Or is this a feature to have different scopes on
00009 the same device?
00010
00011 \todo The device and platform are very similar.  Use a common
00012 mix-in to implement both of them?
00013
00014 Ronan at Keryell point FR
00015
00016 This file is distributed under the University of Illinois Open Source
00017 License.  See LICENSE.TXT for details.
00018 */
00019
00020 #include <type_traits>
00021
00022 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00023 #include "triSYCL/device.hpp"
00024 #include "triSYCL/vendor/triSYCL/scope/detail/util.hpp"
00025 #include "triSYCL/vendor/triSYCL/scope/device/detail/device.hpp"
00026 #include "triSYCL/vendor/triSYCL/scope/platform.hpp"
00027
00028 /// This is an extension providing scope storage for devices
00029 #define SYCL_VENDOR_TRISYCL_DEVICE_SCOPE 1
00030
00031 namespace trisycl::vendor::trisycl::scope {
00032
00033 /** \addtogroup vendor_trisycl_scope triSYCL extension for storage scopes
00034 @{
00035 */
00036
00037 /// An empty device scope to make compilation errors clearer
00038 struct empty_device_scope {
00039 };
00040
00041
00042 /** A conceptual device implementing some device-scoped storage and
00043 based on a platform with platform-scoped storage
00044 */
00045 template <typename DeviceStorage = empty_device_scope
00046           , typename ScopedPlatform =
00047           ::trisycl::vendor::trisycl::scope::empty_platform_scope
00048           >
00049 class device
00050   /* Use the underlying device implementation that can be shared in the
00051 SYCL model */
00052   :  public
00053      ::trisycl::detail::shared_ptr_implementation<device<DeviceStorage,
00054                                                          ScopedPlatform>,
00055                                                   detail::device
00056                                                   <DeviceStorage,
00057                                                    ScopedPlatform>> {
00058   using spi =
00059     ::trisycl::detail::shared_ptr_implementation<device<DeviceStorage,
00060                                                         ScopedPlatform>,
00061                                                  detail::device
00062                                                  <DeviceStorage,
00063                                                   ScopedPlatform>>;
00064
00065   // Allows the comparison operation to access the implementation
00066   friend spi;
00067
00068 public:
00069
00070   // Make the implementation member directly accessible in this class
00071   using spi::implementation;
00072
00073 /// The type of the scoped storage in the device
00074   using storage_type = DeviceStorage;
00075
00076 /// The device scope storage type
00077   using device_scope_type = storage_type;
00078
00079 /// The platform type targeted by this queue
00080   using platform_type = ScopedPlatform;
00081
00082 /// The platform scope storage type
```

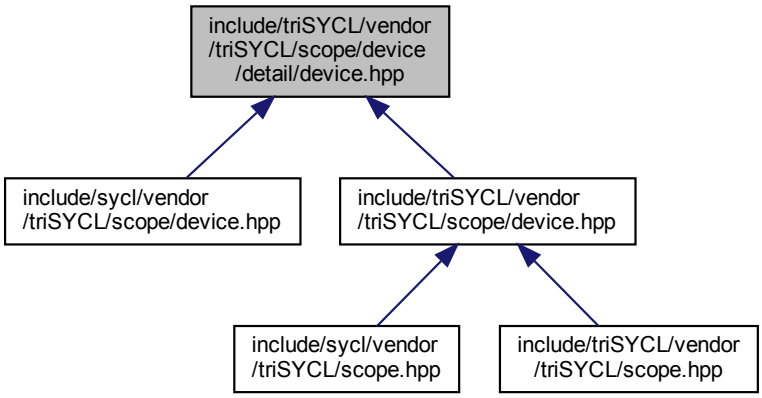
```
00083   using platform_scope_type = detail::storage_type_trait_t<platform_type>;
00084
00085 /// True if there is some storage in the device
00086   static constexpr bool has_some_storage_p =
00087     not std::is_same_v<DeviceStorage, empty_device_scope>;
00088
00089
00090 /** Construct the device with some device-scoped storage on top
00091 of a SYCL device
00092
00093 \param[in] d is the real device to use
00094 */
00095   device(const ::trisycl::device &d,
00096          const ScopedPlatform &p = {}) :
00097     spi {
00098       new detail::device<DeviceStorage, ScopedPlatform> { d, p }
00099     } {}
00100
00101
00102 /// By default use the default (host) device with no platform storage
00103   device() :  device(::trisycl::device {}) {}
00104
00105
00106 /// It is still copyable
00107   device(const device&) = default;
00108
00109
00110 /// It is still movable
00111   device(device&&) = default;
00112
00113
00114 /// Access to the device-scoped storage
00115   auto& get_storage()const {
00116     return implementation->get_storage();
00117   }
00118
00119
00120 /// Access to the underlying scoped platform
00121   auto& get_platform()const {
00122     return implementation->get_platform();
00123   }
00124
00125
00126 /** Add a conversion to \c trisycl::device& so it can be used as
00127 a normal device */
00128     operator ::trisycl::device&() const {
00129       return implementation->get_underlying_device();
00130     }
00131
00132 };
00133
00134 /// @} to end the vendor_trisycl_scope Doxygen group
00135
00136 }
00137
00138
00139 /*
00140 # Some Emacs stuff:
00141 ### Local Variables:
00142 ### ispell-local-dictionary:  "american"
00143 ### eval:  (flyspell-prog-mode)
00144 ### End:
00145 */
00146
00147 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DEVICE_HPP
```

13.283 include/triSYCL/vendor/triSYCL/scope/device/detail/device.hpp File Reference

`#include "triSYCL/device.hpp"`
Include dependency graph for device.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >

  A conceptual device implementing some device-scoped storage and based on a platform with platform-scoped storage. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope
- namespace trisycl::vendor::trisycl::scope::detail

13.284 device.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DEVICE_DETAIL_DEVICE_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DEVICE_DETAIL_DEVICE_HPP
00003
00004 /** \file Implementation details of the SYCL conceptual device with a
00005 device scope storage
00006
00007 \todo The device and platform are very similar.  Use a common
00008 mix-in to implement both of them?
00009
00010 Ronan at Keryell point FR
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include "triSYCL/device.hpp"
00017
00018 namespace trisycl::vendor::trisycl::scope::detail {
00019
00020 /** \addtogroup vendor_trisycl_scope triSYCL extension for storage scopes
00021 @{
00022 */
00023
00024 /** A conceptual device implementing some device-scoped storage and
00025 based on a platform with platform-scoped storage
00026 */
00027 template <typename DeviceStorage
00028           , typename ScopedPlatform>
00029 class device {
00030 /// The storage-less device behind the scene
00031   ::trisycl::device d;
00032
00033 /** The device-scoped storage default-initialized
00034
00035 \todo For now it is allocated on the host in this CPU emulation
00036 but a device compiler and runtime can create this on a real
00037 device
00038 */
00039   DeviceStorage scope_storage;
00040
00041 /// The scoped platform the scoped device is built into
00042   ScopedPlatform platform_with_scope;
00043
00044
00045 public:
00046
00047 /** Construct the device with some device-scoped storage on top
00048 of a SYCL device
00049
00050 \param[in] d is the real device to use
00051 */
00052   device(const ::trisycl::device &d,
00053          const ScopedPlatform &p)
00054     :  d { d }, platform_with_scope { p } {}
00055
00056
00057 /** Construct the device with some device-scoped storage on top
00058 of a SYCL device
00059
00060 \param[in] d is the real device to use
00061 */
00062   device(const ::trisycl::device &d) :  d { d } {}
00063
00064 /// Get the device behind the curtain
00065   auto& get_underlying_device() {
00066     return d;
00067   }
00068
00069
00070 /// Access to the device-scoped storage
00071   auto& get_storage() {
00072     return scope_storage;
00073   }
00074
00075
00076 /// Access to the underlying scoped platform
00077   auto& get_platform() {
00078     return platform_with_scope;
00079   }
00080
00081 };
00082
```

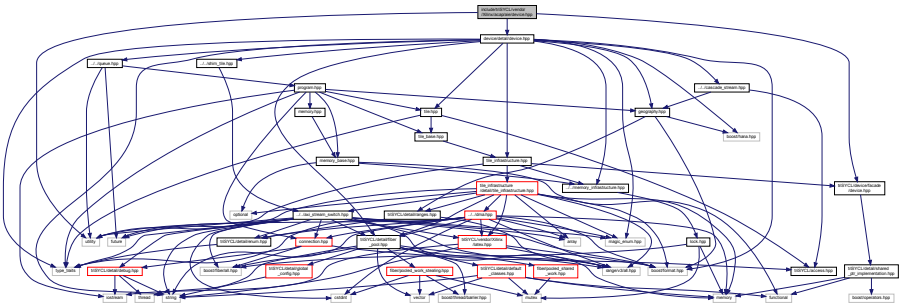
```
00083 /// @} to end the vendor_trisycl_scope Doxygen group
00084
00085 }
00086
00087
00088 /*
00089 # Some Emacs stuff:
00090 ### Local Variables:
00091 ### ispell-local-dictionary:  "american"
00092 ### eval:  (flyspell-prog-mode)
00093 ### End:
00094 */
00095
00096 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_DEVICE_DETAIL_DEVICE_HPP
```

13.285 include/triSYCL/vendor/Xilinx/acap/aie/device.hpp File Reference

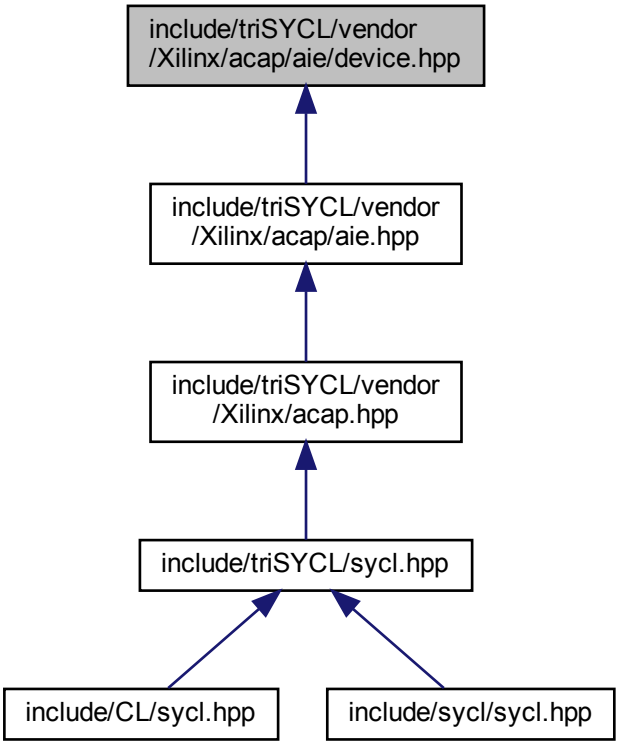
SYCL-friendly device view of AIE.

```
#include <utility>
#include "device/detail/device.hpp"
#include "triSYCL/device/facade/device.hpp"
```

Include dependency graph for device.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::device< Layout >

  Create a SYCL-like device view of an AI Engine CGRA with some layout.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.285.1 Detailed Description

SYCL-friendly device view of AIE.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file device.hpp.

13.286 device.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DEVICE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DEVICE_HPP
00003
00004 /** \file
00005
00006 SYCL-friendly device view of AIE
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include <utility>
00015
00016 #include "device/detail/device.hpp"
00017 #include "triSYCL/device/facade/device.hpp"
00018
00019 /// \ingroup aie
00020 /// @{
00021
00022 namespace trisycl::vendor::xilinx::acap::aie {
00023
00024 /** Create a SYCL-like device view of an AI Engine CGRA with some layout
00025
00026 \param Layout is the layout description of the machine to
00027 instantiate with the physical size
00028 */
00029 template <typename Layout>
00030 class device :  public facade::device<device<Layout>, detail::device<Layout>> {
00031 /// The type encapsulating the implementation
00032   using dd = detail::device<Layout>;
00033
00034 /// The façade used to implement part of the use facing type
00035   using facade_t = facade::device<device<Layout>, dd>;
00036
00037 /// Make the implementation member directly accessible in this class
00038   using facade_t::implementation;
00039
00040  public:
00041 /// Expose some useful internal implementation.
00042 /// Look at them for their documentation
00043   using geo = typename dd::geo;
00044   using cass = typename dd::cass;
00045   using csp = typename dd::csp;
00046   using cmp = typename dd::cmp;
00047   using sass = typename dd::sass;
00048   using ssp = typename dd::ssp;
00049   using smp = typename dd::smp;
00050
00051   // Some shortcuts to useful constants
00052
00053 /// Minimum horizontal coordinate
00054   static auto constexpr x_min = geo::x_min;
00055
00056 /// Maximum horizontal coordinate
00057   static auto constexpr x_max = geo::x_max;
00058
00059 /// Number of tile horizontally in the CGRA
00060   static auto constexpr x_size = geo::x_size;
00061
00062 /// Minimum vertical coordinate
00063   static auto constexpr y_min = geo::y_min;
00064
00065 /// Maximum vertical coordinate
00066   static auto constexpr y_max = geo::y_max;
00067
00068 /// Number of tile vertically in the CGRA
00069   static auto constexpr y_size = geo::y_size;
00070
00071 /// Total number of tiles in the CGRA
00072   static auto constexpr size = geo::size;
00073
00074 /// The default constructor makes a new device
00075   device()
00076       :  facade_t { std::make_shared<dd>() } {}
00077
00078 /** Apply a function for each tile index of the device
00079
00080 \param f is a callable that is to be called like \c f(x,y) for
00081 each tile
00082 */
```

```
00083   template <typename F> void for_each_tile_index(F&& f) {
00084     implementation->for_each_tile_index(std::forward<F>(f));
00085   };
00086
00087 /** Apply a function for each tile infrastructure of the device
00088
00089 \param f is a callable that is to be called like \c f(x,y) for
00090 each tile
00091 */
00092   template <typename F> void for_each_tile(F&& f) {
00093     implementation->for_each_tile(std::forward<F>(f));
00094   };
00095
00096 /** Apply a function for each x tile index of the device
00097
00098 \param f is a callable that is to be called like \c f(x) for
00099 each horizontal index value
00100 */
00101   template <typename F> void for_each_tile_x_index(F&& f) {
00102     implementation->for_each_tile_x_index(std::forward<F>(f));
00103   };
00104
00105 /** Apply a function for each y tile index of the device
00106
00107 \param f is a callable that is to be called like \c f(x) for
00108 each vertical index value
00109 */
00110   template <typename F> void for_each_tile_y_index(F&& f) {
00111     implementation->for_each_tile_y_index(std::forward<F>(f));
00112   };
00113
00114 /** Access to the common infrastructure part of a tile
00115
00116 \param[in] x is the horizontal tile coordinate
00117
00118 \param[in] y is the vertical tile coordinate
00119
00120 \throws trisycl::runtime_error if the coordinate is invalid
00121 */
00122   auto& tile(int x, int y) { return implementation->tile(x, y); }
00123
00124 /** Access to the common infrastructure part of a tile memory
00125
00126 \param[in] x is the horizontal tile coordinate
00127
00128 \param[in] y is the vertical tile coordinate
00129
00130 \throws trisycl::runtime_error if the coordinate is invalid
00131 */
00132   auto& mem(int x, int y) { return implementation->mem(x, y); }
00133
00134 /** Access to the shim tile
00135
00136 \param[in] x is the horizontal coordinate of the shim tile
00137
00138 \throws trisycl::runtime_error if the coordinate is invalid
00139 */
00140   auto& shim(int x) { return implementation->shim(x); }
00141
00142 /// Access the cascade connections
00143 /// \todo To remove?
00144   auto& cascade() { return implementation->cascade(); }
00145
00146 /// Create a queue on this device
00147   auto queue() { return vendor::xilinx::acap::aie::queue { *this }; }
00148
00149 /** Create a program execution for this device
00150
00151 \param Tile is the description of the program tiles to
00152 instantiate.  By default each tile will run an empty program.
00153
00154 \param Memory is the description of the machine memory modules.  By
00155 default the machine has empty memory modules.
00156 */
00157   template <template <typename Device, int X, int Y>
00158             typename Tile = acap::aie::tile,
00159             template <typename Device, int X, int Y>
00160             typename Memory = acap::aie::memory>
00161   auto program() {
00162     return aie::program<device, Tile, Memory> { *this };
00163   }
00164
00165 /** Shortcut to run synchronously a program on this queue
00166
00167 \param Tile is the description of the program tiles to
00168 instantiate.  By default each tile will run an empty program.
00169
```

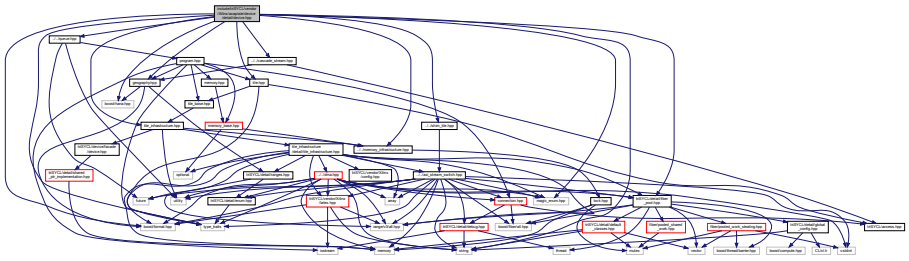
```
00170 \param Memory is the description of the machine memory modules.  By
00171 default the machine has empty memory modules.
00172 */
00173   template <template <typename Device, int X, int Y>
00174             typename Tile = acap::aie::tile,
00175             template <typename Device, int X, int Y>
00176             typename Memory = acap::aie::memory>
00177   void run() {
00178     queue().template run<Tile, Memory>();
00179   }
00180
00181 /** Shortcut to run synchronously an heterogeneous invocable on this
00182 queue
00183
00184 \param f is an invocable taking an heterogeneous tile handler
00185 */
00186   template <typename Invocable> void run(Invocable&& f) {
00187     queue().run(std::forward<Invocable>(f));
00188   }
00189
00190 /** Shortcut to run synchronously a uniform invocable on
00191 this queue
00192
00193 \param f is an invocable taking a uniform tile handler
00194 */
00195   template <typename Invocable> void uniform_run(Invocable&& f) {
00196     queue().uniform_run(std::forward<Invocable>(f));
00197     queue().wait();
00198   }
00199
00200 /** Connect the ports of 2 tiles or shims together with an
00201 hyperspace switched circuit, jumping over the underlying routing
00202 infrastructure.
00203
00204 This is used to speed up communication in emulation and to skip
00205 the need of a routing planner.
00206
00207 \todo To deprecate and implement in another way?
00208
00209 \param[in] T is the type of the data to be transferred
00210
00211 \param[in] SrcPort is the type of the source port, such as
00212 port::tile or port::shim
00213
00214 \param[in] DstPort is the type of the destination port, such as
00215 port::tile or port::shim
00216
00217 \throws trisycl::runtime_error if some coordinates or port
00218 numbers are invalid
00219
00220 \todo Refactor, make the difference between user & physical ports
00221 */
00222   template <typename SrcPort, typename DstPort>
00223   void connect(SrcPort&& src, DstPort&& dst) {
00224     implementation->connect(std::forward<SrcPort>(src),
00225                             std::forward<DstPort>(dst));
00226   }
00227
00228 /** Apply a invocable on all the AXI stream of the neighborhood of
00229 each tile */
00230   template <typename F> void for_each_tile_neighborhood(F&& f) {
00231     implementation->for_each_tile_neighborhood(std::forward<F>(f));
00232   }
00233
00234 /** Display the device layout
00235
00236 \param[in] file_name is the name of the file to write the LaTeX
00237 drawing into or use std::cerr by default
00238 */
00239   auto display(const std::string& file_name = {}) {
00240     return implementation->display(file_name);
00241   }
00242 };
00243
00244 /// @} End the aie Doxygen group
00245
00246 } // namespace trisycl::vendor::xilinx::acap::aie
00247
00248 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DEVICE_HPP
```

13.287 include/triSYCL/vendor/Xilinx/acap/aie/device/detail/device.hpp File Reference

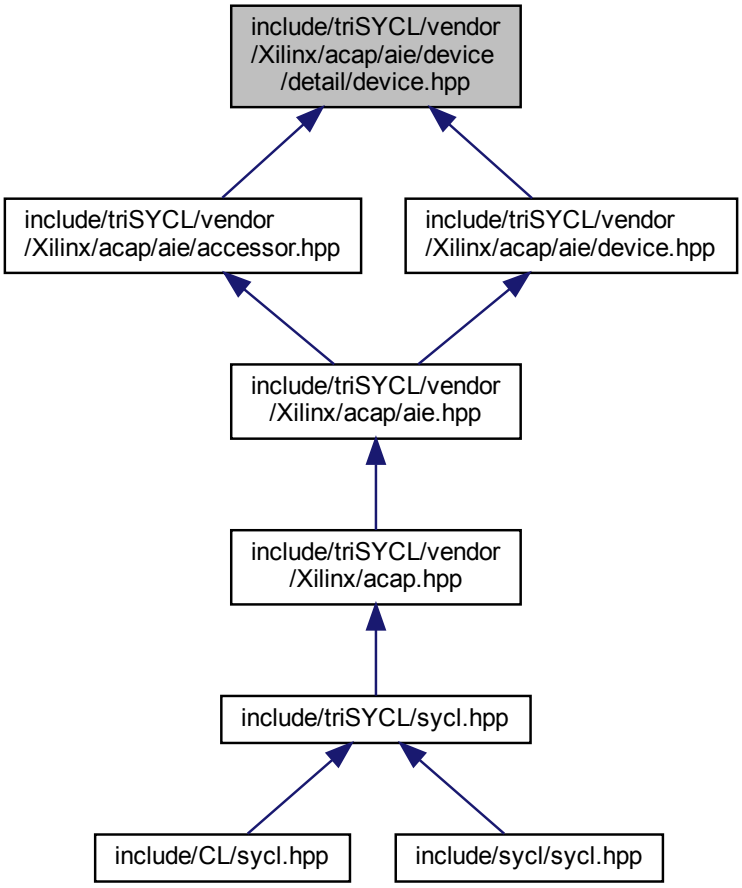
SYCL-friendly device view of AIE.

```
#include <string>
#include <type_traits>
#include "magic_enum.hpp"
#include <boost/format.hpp>
#include <boost/hana.hpp>
#include "../../cascade_stream.hpp"
#include "../../geography.hpp"
#include "../../memory_infrastructure.hpp"
#include "../../queue.hpp"
#include "../../shim_tile.hpp"
#include "../../tile.hpp"
#include "../../tile_infrastructure.hpp"
#include "triSYCL/detail/fiber_pool.hpp"
```

Include dependency graph for device.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::detail::device< Layout >

  Create a SYCL-like device view of an AI Engine CGRA with some layout.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::detail

13.287.1 Detailed Description

SYCL-friendly device view of AIE.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file device.hpp.

13.288 device.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DEVICE_DETAIL_DEVICE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DEVICE_DETAIL_DEVICE_HPP
00003
00004 /** \file
00005
00006 SYCL-friendly device view of AIE
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include <string>
00015 #include <type_traits>
00016
00017 #include "magic_enum.hpp"
00018 #include <boost/format.hpp>
00019 #include <boost/hana.hpp>
00020
00021 #include "../../cascade_stream.hpp"
00022 #include "../../geography.hpp"
00023 #include "../../memory_infrastructure.hpp"
00024 #include "../../queue.hpp"
00025 #include "../../shim_tile.hpp"
00026 #include "../../tile.hpp"
00027 #include "../../tile_infrastructure.hpp"
00028
00029 #include "triSYCL/detail/fiber_pool.hpp"
00030
00031 /// \ingroup aie
00032 /// @{
00033
00034 namespace trisycl::vendor::xilinx::acap::aie::detail {
00035
00036 /** Create a SYCL-like device view of an AI Engine CGRA with some layout
00037
00038 \param Layout is the layout description of the machine to
00039 instantiate with the physical size
00040 */
00041 template <typename Layout> struct device {
00042
00043 /// The geography of the CGRA
00044   using geo = geography<Layout>;
00045
00046 /// Naming shortcut for the core AXI stream switch
00047   using cass = typename geo::core_axi_stream_switch;
00048
00049 /// Naming shortcut for the slave ports of the core AXI stream switch
00050   using csp = typename cass::slave_port_layout;
00051
00052 /// Naming shortcut for the master ports of the core AXI stream switch
00053   using cmp = typename cass::master_port_layout;
00054
00055 /// Naming shortcut for the shim AXI stream switch
00056   using sass = typename geo::shim_axi_stream_switch;
00057
00058 /// Naming shortcut for the slave ports of the shim AXI stream switch
00059   using ssp = typename sass::slave_port_layout;
00060
00061 /// Naming shortcut for the master ports of the shim AXI stream switch
00062   using smp = typename sass::master_port_layout;
00063
```

```
00064 /// The cascade stream infrastructure of the CGRA
00065 /// \todo remove and distribute it instead
00066   cascade_stream<geo> cs;
00067
00068 /** A fiber pool executor to run the infrastructure powered by
00069 TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS std::thread */
00070   ::trisycl::detail::fiber_pool fiber_executor {
00071     1, ::trisycl::detail::fiber_pool::sched::round_robin, false
00072   };
00073   /* { TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS,
00074 detail::fiber_pool::sched::work_stealing, false };
00075 */
00076
00077 /// Keep track of all the (non detail) infrastructure tiles of this device
00078   aie::tile_infrastructure<geo> ti[geo::y_size][geo::x_size];
00079
00080 /** Access to the common infrastructure part of a tile
00081
00082 \param[in] x is the horizontal tile coordinate
00083
00084 \param[in] y is the vertical tile coordinate
00085
00086 \throws trisycl::runtime_error if the coordinate is invalid
00087 */
00088   auto& tile(int x, int y) {
00089     geo::validate_x_y(x, y);
00090     return ti[y][x];
00091   }
00092
00093 /** Apply a function for each tile index of the device
00094
00095 \param f is a callable that is to be called like \c f(x,y) for
00096 each tile
00097 */
00098   template <typename F> void for_each_tile_index(F&& f) {
00099     for (auto [x, y] :
00100          ranges::views::cartesian_product(ranges::views::iota(0, geo::x_size),
00101                                           ranges::views::iota(0, geo::y_size)))
00102       f(x, y);
00103   };
00104
00105 /** Apply a function for each tile infrastructure of the device
00106
00107 \param f is a callable that is to be called like \c f(x,y) for
00108 each tile
00109 */
00110   template <typename F> void for_each_tile(F&& f) {
00111     for_each_tile_index([&](auto x, auto y) { f(tile(x, y)); });
00112   };
00113
00114 /** Apply a function for each x tile index of the device
00115
00116 \param f is a callable that is to be called like \c f(x) for
00117 each horizontal index value
00118 */
00119   template <typename F> void for_each_tile_x_index(F&& f) {
00120     for (auto x :  ranges::views::iota(0, geo::x_size))
00121       f(x);
00122   };
00123
00124 /** Apply a function for each y tile index of the device
00125
00126 \param f is a callable that is to be called like \c f(x) for
00127 each vertical index value
00128 */
00129   template <typename F> void for_each_tile_y_index(F&& f) {
00130     for (auto y :  ranges::views::iota(0, geo::y_size))
00131       f(y);
00132   };
00133
00134 /** Shortcut to access to the common infrastructure part of a tile memory
00135
00136 \param[in] x is the horizontal tile coordinate
00137
00138 \param[in] y is the vertical tile coordinate
00139
00140 \throws trisycl::runtime_error if the coordinate is invalid
00141 */
00142   auto& mem(int x, int y) {
00143     return tile(x, y).mem();
00144   }
00145
00146 /** The shim tiles on the lower row of the tile array
00147
00148 For now we consider only homogeneous shim tiles.
00149 */
00150   shim_tile<device> shims[geo::x_size];
```

```
00151
00152 /** Access to the shim tile
00153
00154 \param[in] x is the horizontal coordinate of the shim tile
00155
00156 \throws trisycl::runtime_error if the coordinate is invalid
00157 */
00158   auto& shim(int x) {
00159     geo::validate_x(x);
00160     return shims[x];
00161   }
00162
00163 /// Access the cascade connections
00164 /// \todo To remove?
00165   auto& cascade() { return cs; }
00166
00167 /** Connect the ports of 2 tiles or shims together with an
00168 hyperspace switched circuit, jumping over the underlying routing
00169 infrastructure.
00170
00171 This is used to speed up communication in emulation and to skip
00172 the need of a routing planner.
00173
00174 \todo To deprecate and implement in another way?
00175
00176 \param[in] T is the type of the data to be transferred
00177
00178 \param[in] SrcPort is the type of the source port, such as
00179 port::tile or port::shim
00180
00181 \param[in] DstPort is the type of the destination port, such as
00182 port::tile or port::shim
00183
00184 \throws trisycl::runtime_error if some coordinates or port
00185 numbers are invalid
00186
00187 \todo Refactor, make the difference between user & physical ports
00188 */
00189   template <typename SrcPort, typename DstPort>
00190   void connect(SrcPort&& src, DstPort&& dst) {
00191     static_assert(std::is_same_v<SrcPort, port::tile> ||
00192                   std::is_same_v<SrcPort, port::shim>,
00193                   "SrcPort type should be port::tile or port::shim");
00194     static_assert(std::is_same_v<DstPort, port::tile> ||
00195                   std::is_same_v<DstPort, port::shim>,
00196                   "DstPort type should be port::tile or port::shim");
00197     // Get the destination in the final switch
00198     auto channel = [&] {
00199       if constexpr (std::is_same_v<DstPort, port::tile>)
00200         return tile(dst.x, dst.y).in_connection(dst.port);
00201       else if constexpr (std::is_same_v<DstPort, port::shim>)
00202         return shim(dst.x).bli_in_connection(dst.port);
00203     }();
00204     // And connect the source  port to it
00205     if constexpr (std::is_same_v<SrcPort, port::tile>)
00206       tile(src.x, src.y).out_connection(src.port) = channel;
00207     else if constexpr (std::is_same_v<SrcPort, port::shim>)
00208       shim(src.x).bli_out_connection(src.port) = channel;
00209   }
00210
00211 /// Apply a function on all the AXI stream of the neighborhood of each tile
00212   template <typename F> void for_each_tile_neighborhood(F&& f) {
00213     for_each_tile_index([&](auto x, auto y) {
00214       // No CTAD yet with Boost::Hana and Clang++-10 (but works with g++-9)
00215       boost::hana::for_each(
00216           geo::core_axi_stream_switch::interconnect, [&](auto connections) {
00217             auto [dx, dy, master_range, slave_range] = connections;
00218             if (geo::is_x_y_valid(x + dx, y + dy))
00219               for (auto [o, i] :  ranges::views::zip(master_range, slave_range))
00220                 f(x, y, x + dx, y + dy, o, i);
00221           });
00222     });
00223   }
00224
00225 /// Build some device level infrastructure
00226   device() {
00227     // Initialize all the tiles with their network connections first
00228     for_each_tile_index([&](auto x, auto y) {
00229       // Create & start the tile infrastructure
00230       tile(x, y) = { x, y, *this, fiber_executor };
00231     });
00232     for_each_tile_x_index([&](auto x) {
00233       // Start the shim infrastructure
00234       shim(x).start(x, fiber_executor);
00235     });
00236
00237     // Only then we can connect the inter-core tile AXI stream NoC
```

```
00238     for_each_tile_neighborhood(
00239         [&](auto x, auto y, auto nx, auto ny, auto m, auto s) {
00240           tile(x, y).output(m) = tile(nx, ny).input(s);
00241         });
00242
00243 /// Make the connections around the shim tiles
00244     for_each_tile_x_index([&](auto x) {
00245       // The connection from shim tile to core tile
00246       for (auto [o, i] :
00247            ranges::views::zip(sass::m_north_range, cass::s_south_range))
00248         shim(x).output(o) = tile(x, 0).input(i);
00249       // The connection from core tile to shim tile
00250       for (auto [o, i] :
00251            ranges::views::zip(cass::m_south_range, sass::s_north_range))
00252         tile(x, 0).output(o) = shim(x).input(i);
00253       if (!geo::is_east_column(x))
00254         for (auto [o, i] :
00255              ranges::views::zip(sass::m_east_range, sass::s_west_range))
00256           shim(x).output(o) = shim(x + 1).input(i);
00257       if (!geo::is_west_column(x))
00258         for (auto [o, i] :
00259              ranges::views::zip(sass::m_west_range, sass::s_east_range))
00260           shim(x).output(o) = shim(x - 1).input(i);
00261     });
00262   }
00263
00264 /** Display the device layout
00265
00266 \param[in] file_name is the name of the file to write the LaTeX
00267 drawing into or use std::cerr by default
00268 */
00269   auto display(const std::string& file_name = {}) {
00270     // Compute the drawing size starting with the individual tile size
00271     auto tile_size = tile(0, 0).display_size();
00272     // And expanding according to the device size.  Add 1 in each
00273     // dimension to fit in the page because of the borders
00274     auto x_size = tile_size.x() * geo::x_size + 1;
00275     auto y_size = tile_size.y() * geo::y_size + 1;
00276 /// The LaTeX generation is handled by a LaTeX context
00277     latex::context c { { x_size, y_size }, file_name };
00278
00279     for_each_tile_index([&](auto x, auto y) { tile(x, y).display(c); });
00280
00281     // Connect each tile to its neighbors
00282     for_each_tile_neighborhood(
00283         [&](auto x, auto y, auto nx, auto ny, auto m, auto s) {
00284           c.add((boost::format { R"(
00285 \draw (node cs:name=TileX%1%Y%2%M%3%)
00286 -- (node cs:name=TileX%4%Y%5%S%6%);)" } %
00287 x % y % c.clean_node(magic_enum::enum_name(m)) % nx % ny %
00288 c.clean_node(magic_enum::enum_name(s)))
00289 .str());
00290 });
00291
00292 c.display();
00293 }
00294 };
00295
00296 /// @} End the aie Doxygen group
00297
00298 } // namespace trisycl::vendor::xilinx::acap::aie::detail
00299
00300 /*
00301 # Some Emacs stuff:
00302 ### Local Variables:
00303 ### ispell-local-dictionary:  "american"
00304 ### eval:  (flyspell-prog-mode)
00305 ### End:
00306 */
00307
00308 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DEVICE_DETAIL_DEVICE_HPP
```

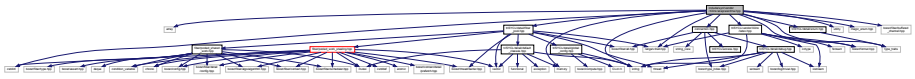
13.289 include/sycl/vendor/Xilinx/acap/aie/dma.hpp File Reference

Model of a tile DMA.

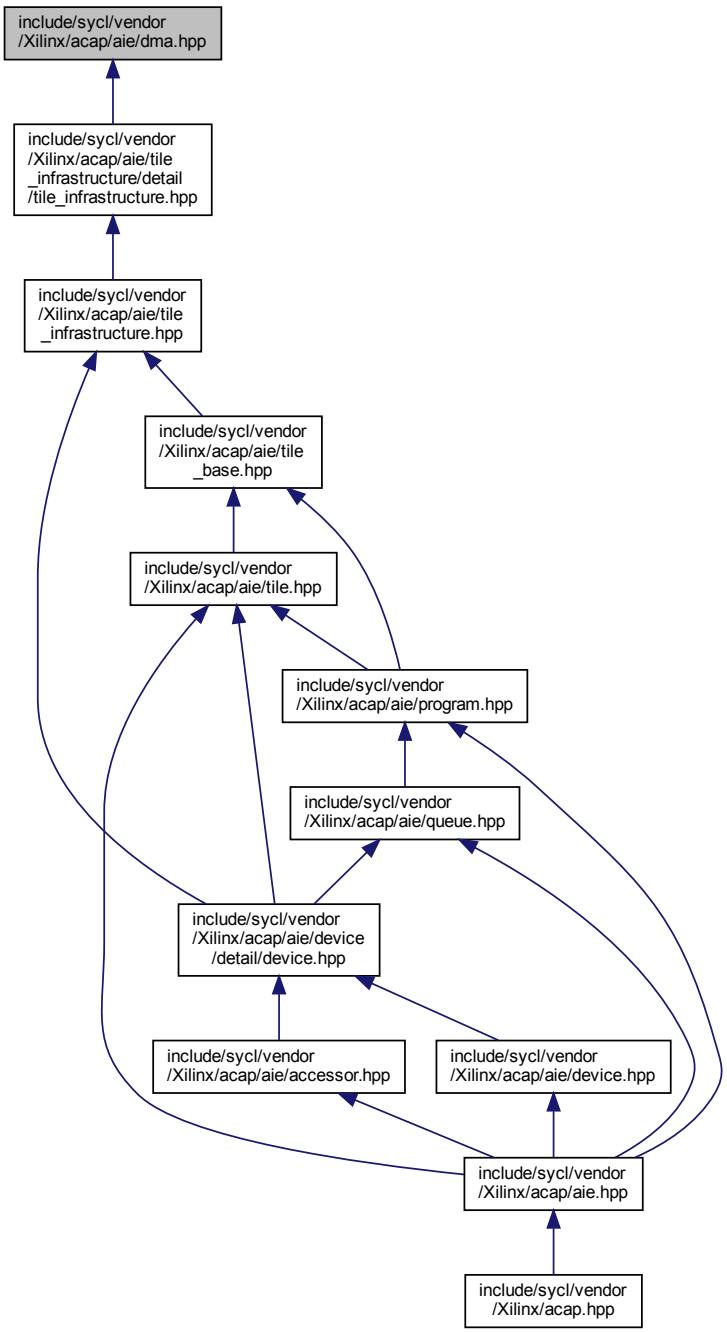
```
#include <array>
#include <cstdint>
#include <memory>
```

```
#include <string>
#include <type_traits>
#include <utility>
#include "magic_enum.hpp"
#include <boost/fiber/buffered_channel.hpp>
#include <boost/format.hpp>
#include <range/v3/all.hpp>
#include "connection.hpp"
#include "triSYCL/detail/enum.hpp"
#include "triSYCL/detail/fiber_pool.hpp"
#include "triSYCL/vendor/Xilinx/latex.hpp"
```

Include dependency graph for dma.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >

  Implement a small DSL to chain DMA & locking operations like allowed by the hardware. More...
- class trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >

  Generic DMA.
- class trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >

Receiving DMA.

- class trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >

  Sending DMA.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.289.1 Detailed Description

Model of a tile DMA.

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file dma.hpp.

13.290 dma.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DMA_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DMA_HPP
00003
00004 /** \file
00005
00006 Model of a tile DMA
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <array>
00013 #include <cstdint>
00014 #include <memory>
00015 #include <string>
00016 #include <type_traits>
00017 #include <utility>
00018
00019 #include "magic_enum.hpp"
00020 #include <boost/fiber/buffered_channel.hpp>
00021 #include <boost/format.hpp>
00022
00023 #include <range/v3/all.hpp>
00024
00025 #include "connection.hpp"
00026 #include "triSYCL/detail/enum.hpp"
00027 #include "triSYCL/detail/fiber_pool.hpp"
00028 #include "triSYCL/vendor/Xilinx/latex.hpp"
00029
00030 namespace trisycl::vendor::xilinx::acap::aie {
00031
00032 /// \ingroup aie
00033 /// @{
00034
00035 /** Implement a small DSL to chain DMA & locking operations like
00036 allowed by the hardware
00037
00038 \param[in] Tile is the type of tile to keep a reference on it
00039
00040 \param[in] DMA is the type of the DMA involved in the operation
00041
00042 \param[in] LockRef is the type of the reference to the lock once a
```

```
00043 lock() operation is involved.  Otherwise just use a \c bool as a
00044 dummy type internally.
00045 */
00046 template <typename Tile, typename DMA, typename LockRef = bool> class dma_dsl {
00047 /// Keep track of the tile involved in the operation chain
00048   Tile& tile;
00049
00050 /// Keep track of the DMA involved in the operation chain
00051   DMA& dma;
00052
00053 /** Keep track of the current lock involved in the operation chain
00054 if any */
00055   LockRef lck;
00056
00057  public:
00058 /// Keep track of operation with only DMA involved
00059   dma_dsl(Tile& t, DMA& d)
00060       :  tile { t }
00061       , dma { d } {}
00062
00063 /// Keep track of operation with DMA and lock involved
00064   dma_dsl(Tile& t, DMA& d, LockRef l)
00065       :  tile { t }
00066       , dma { d }
00067       , lck { l } {}
00068
00069 /// Select a lock for the following operations
00070   auto lock(int l) {
00071     auto& lock_ref = tile.mem().lock(l);
00072     return dma_dsl<Tile, DMA, decltype(lock_ref)> { tile, dma, lock_ref };
00073   }
00074
00075 /// Enqueue a DMA receive command
00076   template <typename Cmd> auto& receive(Cmd&& cmd) {
00077     dma.receive(std::forward<Cmd>(cmd));
00078     // To be able to chain another DMA operation on it
00079     return *this;
00080   }
00081
00082 /// Enqueue a DMA transfer command
00083   template <typename Cmd> auto& send(Cmd&& cmd) {
00084     dma.send(std::forward<Cmd>(cmd));
00085     // To be able to chain another DMA operation on it
00086     return *this;
00087   }
00088
00089 /** Wait for the transfers to complete
00090
00091 The API assumes that the program is written in such a way that
00092 at some point the queue will be empty so there is no continuous
00093 DMA command enqueuing for example.  */
00094   auto& wait() {
00095     dma.wait();
00096     // To be able to chain another DMA operation on it
00097     return *this;
00098   }
00099 };
00100
00101 /// Generic DMA
00102 template <typename AXIStreamSwitch, typename SpecializedDMA>
00103 class dma :  ::trisycl::detail::debug<SpecializedDMA> {
00104
00105 /// Use std::span for 1D contiguous transfers only for now
00106   // using dma_command = std::span<axi_packet::value_type>;
00107
00108  private:
00109 /// The type of commands executable by the DMA. Very generic for now
00110   using dma_command = std::function<void(void)>;
00111
00112 /// Queue for generic DMA commands
00113   boost::fibers::buffered_channel<dma_command> c { 8 };
00114
00115 /// To shepherd the fiber playing the data mover behind the DMA
00116   ::trisycl::detail::fiber_pool::future<void> data_mover;
00117
00118 /// Count the in-flight commands
00119   std::atomic<int> nb_command { 0 };
00120
00121 /// To wait for the DMA command queue to drain
00122   boost::fibers::condition_variable waiting_room;
00123
00124 /// To protect the waiting mechanism
00125   boost::fibers::mutex waiting_mutex;
00126
00127 /// Wait for the fiber to complete
00128   void join() {
00129     // Get the value of the future, to get an exception if any
```

```
00130     data_mover.get();
00131   }
00132
00133 /// Launch the data mover on the execution pool
00134   template <typename Invocable>
00135   void launch(::trisycl::detail::fiber_pool& fiber_executor, Invocable&& f) {}
00136
00137 /// Dequeue a DMA command
00138   auto pop_command(auto& command) { return c.pop(command); }
00139
00140 /// Notify a command has been processed
00141   void commit_command() {
00142     if (--nb_command == 0)
00143       // Warn anyone waiting for the DMA queue to drain it might be empty now
00144       waiting_room.notify_all();
00145   }
00146
00147  protected:
00148 /// Enqueue a DMA command
00149   template <typename Cmd> void push_command(Cmd&& command) {
00150     ++nb_command;
00151     c.push(std::forward<Cmd>(command));
00152   }
00153
00154  public:
00155 /// Start the DMA engine using an executor
00156   dma(::trisycl::detail::fiber_pool& fe) {
00157     data_mover = fe.submit([&] {
00158       dma_command sp;
00159       for (;;) {
00160         // Read a DMA command
00161         if (this->pop_command(sp) != boost::fibers::channel_op_status::success)
00162           // The channel is closed, stop working
00163           break;
00164         // Execute the command
00165         sp();
00166         commit_command();
00167       }
00168     });
00169   }
00170
00171 /// Destructor handling the correct infrastructure shutdown
00172   ~dma() {
00173     // Close the command queue so the DMA does not accept any work
00174     c.close();
00175     /* Wait for the commands to drain to avoid calling
00176 std::terminate on destruction...  */
00177     join();
00178   }
00179
00180 /** Wait for the transfers to complete
00181
00182 The API assumes that the program is written in such a way that
00183 at some point the queue will be empty so there is no continuous
00184 DMA command enqueuing for example.  */
00185   SpecializedDMA& wait() {
00186     if (nb_command != 0) {
00187       // We need to wait because there are still operations executing
00188       std::unique_lock lk { waiting_mutex };
00189       waiting_room.wait(lk, [&] { return nb_command == 0; });
00190     }
00191     /* To be able to chain another DMA operation on it, return the DMA
00192 object itself */
00193     return static_cast<SpecializedDMA&>(*this);
00194   }
00195 };
00196
00197 /** Receiving DMA
00198
00199 \todo Factorize out the FIFO port
00200 */
00201 template <typename AXIStreamSwitch>
00202 class receiving_dma
00203     :  public dma<AXIStreamSwitch, receiving_dma<AXIStreamSwitch>>
00204     , public communicator_port {
00205   using dma_base = dma<AXIStreamSwitch, receiving_dma<AXIStreamSwitch>>;
00206
00207   /* boost::fibers::unbuffered_channel has no try_push() function, so
00208 use a buffered version for now
00209
00210 \todo open a GitHub issue on Boost.Fiber
00211 */
00212   boost::fibers::buffered_channel<axi_packet> fifo { 8 };
00213
00214 /// Keep track of the AXI stream switch owning this port for debugging
00215   AXIStreamSwitch& axi_ss;
00216
```

```
00217  public:
00218 /// Start the receiving DMA engine using an executor
00219   receiving_dma(AXIStreamSwitch& axi_ss, ::trisycl::detail::fiber_pool& fe)
00220       :  dma_base { fe }
00221       , axi_ss { axi_ss } {}
00222
00223 /** Enqueue a DMA transfer to receive a span
00224
00225 Use std::span for 1D contiguous transfers only for now
00226 */
00227   void receive(std::span<axi_packet::value_type> sp) {
00228     this->push_command([=, this] {
00229       /* Write each element received from input router port into the
00230 memory described by DMA operation */
00231       for (auto& e :  sp) {
00232         e = read();
00233       }
00234     });
00235   }
00236
00237 /// The network sends some data to the DMA receiver input
00238   void write(const axi_packet& v)override { fifo.push(v); }
00239
00240 /** The network try to send some data to the DMA receiver input
00241
00242 \return true if the packet is correctly enqueued or if the
00243 stream is closed, in order to to move on since we are in the
00244 tear-down phase of the infrastructure
00245
00246 \todo think about it again...
00247 */
00248   bool try_write(const axi_packet& v)override {
00249     return this->fifo.try_push(v) != boost::fibers::channel_op_status::full;
00250   }
00251
00252 /// Waiting read by a tile program on a core input port from the switch
00253   axi_packet::value_type read()override { return fifo.value_pop().data; }
00254
00255 /** Non-blocking read to a core input port
00256
00257 \return true if the value was correctly read
00258 */
00259   bool try_read(axi_packet::value_type& v)override {
00260     axi_packet p;
00261
00262     if (fifo.try_pop(p) == boost::fibers::channel_op_status::success) {
00263       v = p.data;
00264       return true;
00265     }
00266     return false;
00267   }
00268 };
00269
00270 /** Sending DMA
00271
00272 \todo Factorize out the FIFO port
00273 */
00274 template <typename AXIStreamSwitch>
00275 class sending_dma :  public dma<AXIStreamSwitch, sending_dma<AXIStreamSwitch>> {
00276   using dma_base = dma<AXIStreamSwitch, sending_dma<AXIStreamSwitch>>;
00277
00278   std::shared_ptr<communicator_port> output_port;
00279
00280  public:
00281 /// Start the DMA engine using an executor to push things on a port
00282   sending_dma(::trisycl::detail::fiber_pool& fe,
00283               std::shared_ptr<communicator_port> output)
00284       :  dma_base { fe }
00285       , output_port { output } {}
00286
00287 /// Enqueue a DMA transfer to send a span
00288   void send(std::span<axi_packet::value_type> sp) {
00289     this->push_command([=, this] {
00290       // Send each element of this span to the output port
00291       for (const auto& e :  sp) {
00292         output_port->write(e);
00293       }
00294     });
00295   }
00296 };
00297
00298 /// @} End the aie Doxygen group
00299
00300 } // namespace trisycl::vendor::xilinx::acap::aie
00301
00302 /*
00303 # Some Emacs stuff:
```

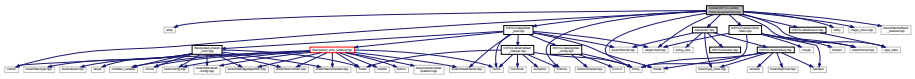
```
00304 ### Local Variables:
00305 ### ispell-local-dictionary:  "american"
00306 ### eval:  (flyspell-prog-mode)
00307 ### End:
00308 */
00309
00310 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DMA_HPP
```

13.291 include/triSYCL/vendor/Xilinx/acap/aie/dma.hpp File Reference

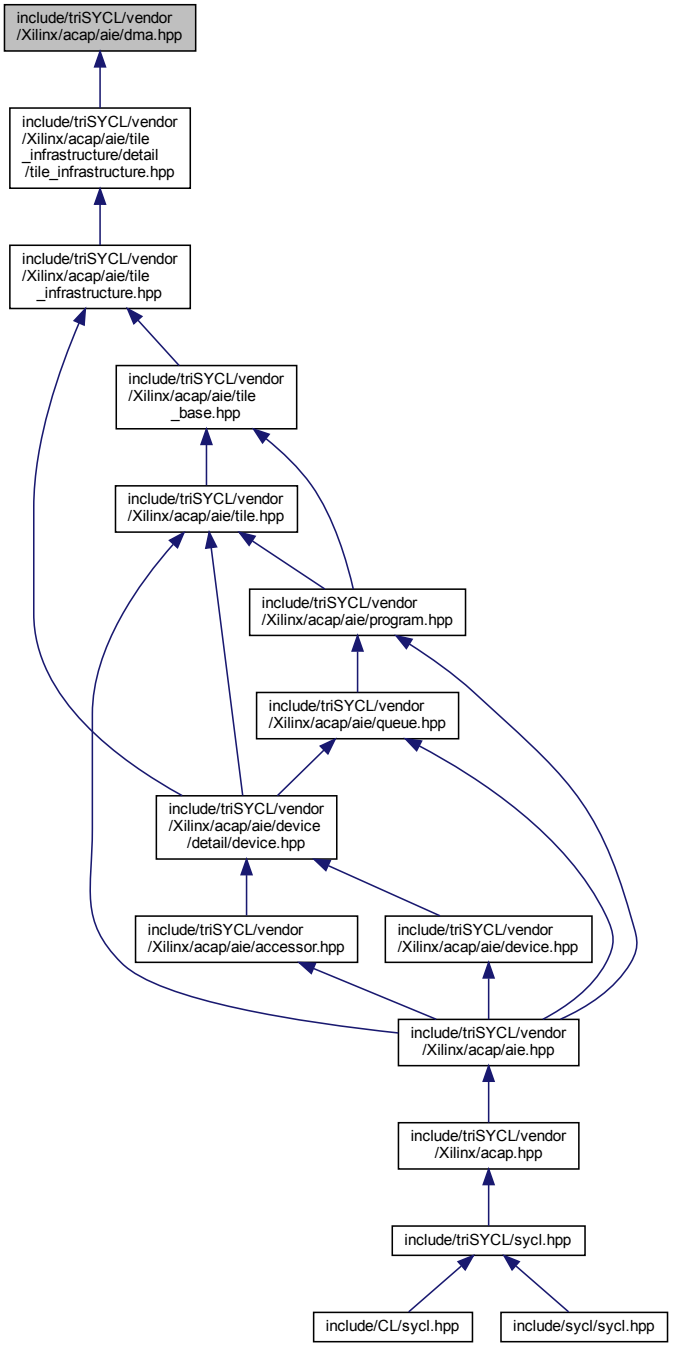
Model of a tile DMA.

```
#include <array>
#include <cstdint>
#include <memory>
#include <string>
#include <type_traits>
#include <utility>
#include "magic_enum.hpp"
#include <boost/fiber/buffered_channel.hpp>
#include <boost/format.hpp>
#include <range/v3/all.hpp>
#include "connection.hpp"
#include "triSYCL/detail/enum.hpp"
#include "triSYCL/detail/fiber_pool.hpp"
#include "triSYCL/vendor/Xilinx/latex.hpp"
```

Include dependency graph for dma.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >

  Implement a small DSL to chain DMA & locking operations like allowed by the hardware. More...
- class trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >

  Generic DMA.
- class trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >

Receiving DMA.

- class trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >

  Sending DMA.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.291.1 Detailed Description

Model of a tile DMA.

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file dma.hpp.

13.292 dma.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DMA_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DMA_HPP
00003
00004 /** \file
00005
00006 Model of a tile DMA
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <array>
00013 #include <cstdint>
00014 #include <memory>
00015 #include <string>
00016 #include <type_traits>
00017 #include <utility>
00018
00019 #include "magic_enum.hpp"
00020 #include <boost/fiber/buffered_channel.hpp>
00021 #include <boost/format.hpp>
00022
00023 #include <range/v3/all.hpp>
00024
00025 #include "connection.hpp"
00026 #include "triSYCL/detail/enum.hpp"
00027 #include "triSYCL/detail/fiber_pool.hpp"
00028 #include "triSYCL/vendor/Xilinx/latex.hpp"
00029
00030 namespace trisycl::vendor::xilinx::acap::aie {
00031
00032 /// \ingroup aie
00033 /// @{
00034
00035 /** Implement a small DSL to chain DMA & locking operations like
00036 allowed by the hardware
00037
00038 \param[in] Tile is the type of tile to keep a reference on it
00039
00040 \param[in] DMA is the type of the DMA involved in the operation
00041
00042 \param[in] LockRef is the type of the reference to the lock once a
```

```
00043 lock() operation is involved.  Otherwise just use a \c bool as a
00044 dummy type internally.
00045 */
00046 template <typename Tile, typename DMA, typename LockRef = bool> class dma_dsl {
00047 /// Keep track of the tile involved in the operation chain
00048   Tile& tile;
00049
00050 /// Keep track of the DMA involved in the operation chain
00051   DMA& dma;
00052
00053 /** Keep track of the current lock involved in the operation chain
00054 if any */
00055   LockRef lck;
00056
00057  public:
00058 /// Keep track of operation with only DMA involved
00059   dma_dsl(Tile& t, DMA& d)
00060       :  tile { t }
00061       , dma { d } {}
00062
00063 /// Keep track of operation with DMA and lock involved
00064   dma_dsl(Tile& t, DMA& d, LockRef l)
00065       :  tile { t }
00066       , dma { d }
00067       , lck { l } {}
00068
00069 /// Select a lock for the following operations
00070   auto lock(int l) {
00071     auto& lock_ref = tile.mem().lock(l);
00072     return dma_dsl<Tile, DMA, decltype(lock_ref)> { tile, dma, lock_ref };
00073   }
00074
00075 /// Enqueue a DMA receive command
00076   template <typename Cmd> auto& receive(Cmd&& cmd) {
00077     dma.receive(std::forward<Cmd>(cmd));
00078     // To be able to chain another DMA operation on it
00079     return *this;
00080   }
00081
00082 /// Enqueue a DMA transfer command
00083   template <typename Cmd> auto& send(Cmd&& cmd) {
00084     dma.send(std::forward<Cmd>(cmd));
00085     // To be able to chain another DMA operation on it
00086     return *this;
00087   }
00088
00089 /** Wait for the transfers to complete
00090
00091 The API assumes that the program is written in such a way that
00092 at some point the queue will be empty so there is no continuous
00093 DMA command enqueuing for example.  */
00094   auto& wait() {
00095     dma.wait();
00096     // To be able to chain another DMA operation on it
00097     return *this;
00098   }
00099 };
00100
00101 /// Generic DMA
00102 template <typename AXIStreamSwitch, typename SpecializedDMA>
00103 class dma :  ::trisycl::detail::debug<SpecializedDMA> {
00104
00105 /// Use std::span for 1D contiguous transfers only for now
00106   // using dma_command = std::span<axi_packet::value_type>;
00107
00108  private:
00109 /// The type of commands executable by the DMA. Very generic for now
00110   using dma_command = std::function<void(void)>;
00111
00112 /// Queue for generic DMA commands
00113   boost::fibers::buffered_channel<dma_command> c { 8 };
00114
00115 /// To shepherd the fiber playing the data mover behind the DMA
00116   ::trisycl::detail::fiber_pool::future<void> data_mover;
00117
00118 /// Count the in-flight commands
00119   std::atomic<int> nb_command { 0 };
00120
00121 /// To wait for the DMA command queue to drain
00122   boost::fibers::condition_variable waiting_room;
00123
00124 /// To protect the waiting mechanism
00125   boost::fibers::mutex waiting_mutex;
00126
00127 /// Wait for the fiber to complete
00128   void join() {
00129     // Get the value of the future, to get an exception if any
```

```
00130     data_mover.get();
00131   }
00132
00133 /// Launch the data mover on the execution pool
00134   template <typename Invocable>
00135   void launch(::trisycl::detail::fiber_pool& fiber_executor, Invocable&& f) {}
00136
00137 /// Dequeue a DMA command
00138   auto pop_command(auto& command) { return c.pop(command); }
00139
00140 /// Notify a command has been processed
00141   void commit_command() {
00142     if (--nb_command == 0)
00143       // Warn anyone waiting for the DMA queue to drain it might be empty now
00144       waiting_room.notify_all();
00145   }
00146
00147  protected:
00148 /// Enqueue a DMA command
00149   template <typename Cmd> void push_command(Cmd&& command) {
00150     ++nb_command;
00151     c.push(std::forward<Cmd>(command));
00152   }
00153
00154  public:
00155 /// Start the DMA engine using an executor
00156   dma(::trisycl::detail::fiber_pool& fe) {
00157     data_mover = fe.submit([&] {
00158       dma_command sp;
00159       for (;;) {
00160         // Read a DMA command
00161         if (this->pop_command(sp) != boost::fibers::channel_op_status::success)
00162           // The channel is closed, stop working
00163           break;
00164         // Execute the command
00165         sp();
00166         commit_command();
00167       }
00168     });
00169   }
00170
00171 /// Destructor handling the correct infrastructure shutdown
00172   ~dma() {
00173     // Close the command queue so the DMA does not accept any work
00174     c.close();
00175     /* Wait for the commands to drain to avoid calling
00176 std::terminate on destruction...  */
00177     join();
00178   }
00179
00180 /** Wait for the transfers to complete
00181
00182 The API assumes that the program is written in such a way that
00183 at some point the queue will be empty so there is no continuous
00184 DMA command enqueuing for example.  */
00185   SpecializedDMA& wait() {
00186     if (nb_command != 0) {
00187       // We need to wait because there are still operations executing
00188       std::unique_lock lk { waiting_mutex };
00189       waiting_room.wait(lk, [&] { return nb_command == 0; });
00190     }
00191     /* To be able to chain another DMA operation on it, return the DMA
00192 object itself */
00193     return static_cast<SpecializedDMA&>(*this);
00194   }
00195 };
00196
00197 /** Receiving DMA
00198
00199 \todo Factorize out the FIFO port
00200 */
00201 template <typename AXIStreamSwitch>
00202 class receiving_dma
00203     :  public dma<AXIStreamSwitch, receiving_dma<AXIStreamSwitch>>
00204     , public communicator_port {
00205   using dma_base = dma<AXIStreamSwitch, receiving_dma<AXIStreamSwitch>>;
00206
00207   /* boost::fibers::unbuffered_channel has no try_push() function, so
00208 use a buffered version for now
00209
00210 \todo open a GitHub issue on Boost.Fiber
00211 */
00212   boost::fibers::buffered_channel<axi_packet> fifo { 8 };
00213
00214 /// Keep track of the AXI stream switch owning this port for debugging
00215   AXIStreamSwitch& axi_ss;
00216
```

```
00217  public:
00218 /// Start the receiving DMA engine using an executor
00219   receiving_dma(AXIStreamSwitch& axi_ss, ::trisycl::detail::fiber_pool& fe)
00220       :  dma_base { fe }
00221       , axi_ss { axi_ss } {}
00222
00223 /** Enqueue a DMA transfer to receive a span
00224
00225 Use std::span for 1D contiguous transfers only for now
00226 */
00227   void receive(std::span<axi_packet::value_type> sp) {
00228     this->push_command([=, this] {
00229       /* Write each element received from input router port into the
00230 memory described by DMA operation */
00231       for (auto& e :  sp) {
00232         e = read();
00233       }
00234     });
00235   }
00236
00237 /// The network sends some data to the DMA receiver input
00238   void write(const axi_packet& v)override { fifo.push(v); }
00239
00240 /** The network try to send some data to the DMA receiver input
00241
00242 \return true if the packet is correctly enqueued or if the
00243 stream is closed, in order to to move on since we are in the
00244 tear-down phase of the infrastructure
00245
00246 \todo think about it again...
00247 */
00248   bool try_write(const axi_packet& v)override {
00249     return this->fifo.try_push(v) != boost::fibers::channel_op_status::full;
00250   }
00251
00252 /// Waiting read by a tile program on a core input port from the switch
00253   axi_packet::value_type read()override { return fifo.value_pop().data; }
00254
00255 /** Non-blocking read to a core input port
00256
00257 \return true if the value was correctly read
00258 */
00259   bool try_read(axi_packet::value_type& v)override {
00260     axi_packet p;
00261
00262     if (fifo.try_pop(p) == boost::fibers::channel_op_status::success) {
00263       v = p.data;
00264       return true;
00265     }
00266     return false;
00267   }
00268 };
00269
00270 /** Sending DMA
00271
00272 \todo Factorize out the FIFO port
00273 */
00274 template <typename AXIStreamSwitch>
00275 class sending_dma :  public dma<AXIStreamSwitch, sending_dma<AXIStreamSwitch>> {
00276   using dma_base = dma<AXIStreamSwitch, sending_dma<AXIStreamSwitch>>;
00277
00278   std::shared_ptr<communicator_port> output_port;
00279
00280  public:
00281 /// Start the DMA engine using an executor to push things on a port
00282   sending_dma(::trisycl::detail::fiber_pool& fe,
00283               std::shared_ptr<communicator_port> output)
00284       :  dma_base { fe }
00285       , output_port { output } {}
00286
00287 /// Enqueue a DMA transfer to send a span
00288   void send(std::span<axi_packet::value_type> sp) {
00289     this->push_command([=, this] {
00290       // Send each element of this span to the output port
00291       for (const auto& e :  sp) {
00292         output_port->write(e);
00293       }
00294     });
00295   }
00296 };
00297
00298 /// @} End the aie Doxygen group
00299
00300 } // namespace trisycl::vendor::xilinx::acap::aie
00301
00302 /*
00303 # Some Emacs stuff:
```

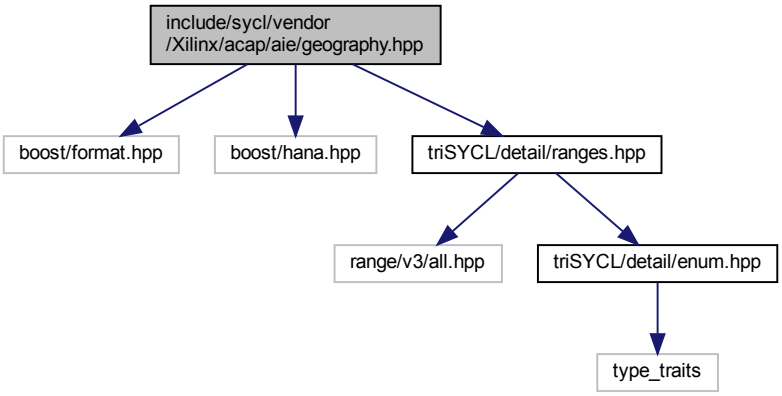
```
00304 ### Local Variables:
00305 ### ispell-local-dictionary:  "american"
00306 ### eval:  (flyspell-prog-mode)
00307 ### End:
00308 */
00309
00310 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_DMA_HPP
```

13.293 include/sycl/vendor/Xilinx/acap/aie/geography.hpp File Reference

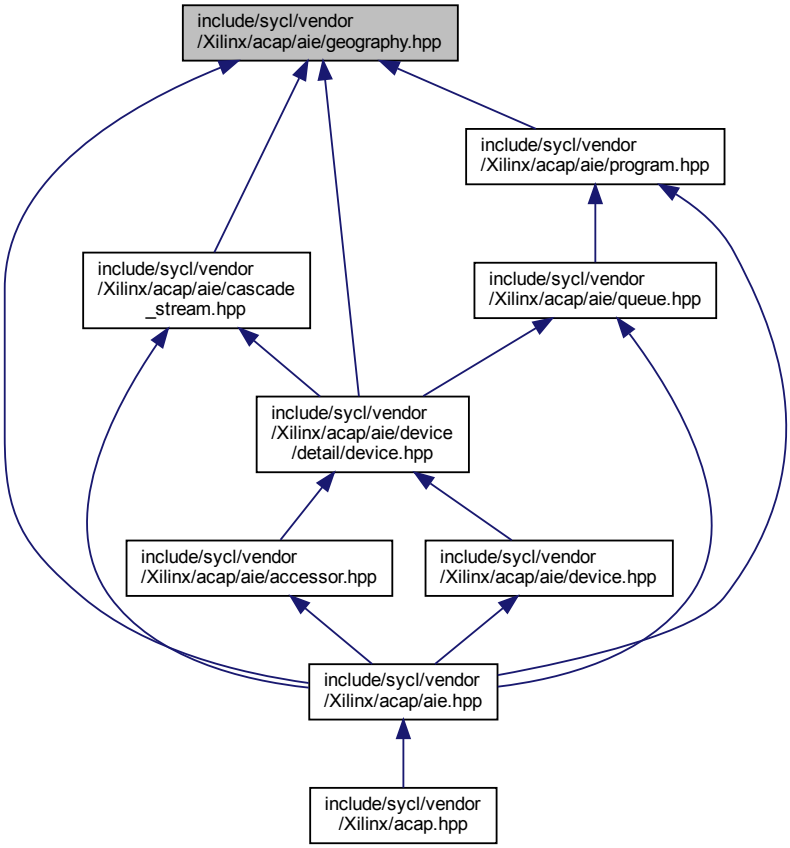
Geographic information about the AI Engine CGRA.

```
#include <boost/format.hpp>
#include <boost/hana.hpp>
#include "triSYCL/detail/ranges.hpp"
```

Include dependency graph for geography.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::geography< Layout >

  Some geographic information about the AI Engine array. More...
- struct trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch
- struct trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.293.1 Detailed Description

Geographic information about the AI Engine CGRA.

Define in a central location most of the ACAP architectural parameters.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file geography.hpp.

13.294 geography.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_GEOGRAPHY_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_GEOGRAPHY_HPP
00003
00004 /** \file
00005
00006 Geographic information about the AI Engine CGRA
00007
00008 Define in a central location most of the ACAP architectural
00009 parameters.
00010
00011 Ronan dot Keryell at Xilinx dot com
00012
00013 This file is distributed under the University of Illinois Open Source
00014 License.  See LICENSE.TXT for details.
00015 */
00016
00017 #include <boost/format.hpp>
00018 #include <boost/hana.hpp>
00019
00020 #include "triSYCL/detail/ranges.hpp"
00021
00022 namespace trisycl::vendor::xilinx::acap::aie {
00023
00024 /// \ingroup aie
00025 /// @{
00026
00027 /** Some geographic information about the AI Engine array
00028
00029 \param Layout is a class describing the basic layout of the CGRA,
00030 setting for example the x & y size
00031 */
00032 template <typename Layout>
00033 struct geography :  Layout {
00034 /// Keep the layout used for this CGRA
00035   using layout = Layout;
00036
00037 /// Minimum horizontal coordinate
00038   static auto constexpr x_min = 0;
00039
00040 /// Minimum vertical coordinate
00041   static auto constexpr y_min = 0;
00042
00043 /// Number of tile horizontally in the CGRA
00044   static auto constexpr x_size = layout::x_max + 1;
00045
00046 /// Number of tile vertically in the CGRA
00047   static auto constexpr y_size = layout::y_max + 1;
00048
00049 /// Total number of tiles in the CGRA
00050   static auto constexpr size = x_size*y_size;
00051
00052
00053 /** A tuple with the coordinate tuples of all the tiles inside the
00054 CGRA geometry */
00055   static auto constexpr tile_coordinates = boost::hana::cartesian_product(
00056     boost::hana::make_tuple(
00057         boost::hana::range_c<int, y_min, y_size>
00058       , boost::hana::range_c<int, x_min, x_size>
00059                             )
00060                                                                     );
00061
```

```
00062
00063 /** Generate a tuple by tiling the Tileable object onto the CGRA geometry
00064
00065 \param Tileable is the type to tile according to coordinate \p X
00066 and \p Y
00067 */
00068   template <template <int X, int Y> typename Tileable>
00069   static auto generate_tiles() {
00070     return boost::hana::transform(
00071         tile_coordinates
00072       , [] (auto coord) {
00073           return Tileable<boost::hana::at_c<1>(coord),
00074                           boost::hana::at_c<0>(coord)> {};
00075         }
00076                                   );
00077   }
00078
00079
00080 /** Compute the linearized coordinate of a tile
00081
00082 \param[in] x is the horizontal tile coordinate
00083
00084 \param[in] y is the vertical tile coordinate
00085 */
00086   static auto constexpr linear_id(int x, int y) {
00087     return x + x_size*y;
00088   }
00089
00090
00091 /** Compute the horizontal coordinate of a tile from linear id
00092
00093 \param[in] linear_id is a linearized coordinate
00094 */
00095   static auto constexpr linear_x(int linear_id) {
00096     return linear_id%x_size;
00097   }
00098
00099
00100 /** Compute the vertical coordinate of a tile from a linear id
00101
00102 \param[in] linear_id is a linearized coordinate
00103 */
00104   static auto constexpr linear_y(int linear_id) {
00105     return linear_id/x_size;
00106   }
00107
00108
00109 /** Test if an horizontal coordinate is valid
00110
00111 \param[in] x is the horizontal tile coordinate
00112 */
00113   static bool constexpr is_x_valid(int x) {
00114     return x_min <= x && x <= layout::x_max;
00115   }
00116
00117
00118 /** Validate an horizontal coordinate
00119
00120 Do nothing but throwing if the coordinate is not a valid tile
00121 coordinate.
00122
00123 \param[in] x is the horizontal tile coordinate
00124
00125 \throws trisycl::runtime_error if the coordinate is invalid
00126 */
00127   static void validate_x(int x) {
00128     if (!is_x_valid(x)) {
00129       throw ::trisycl::runtime_error {
00130         (boost::format { "The x coordinate %1% is not between %2% and %3%" }
00131           % x % x_min % layout::x_max).str() };
00132     }
00133   }
00134
00135
00136 /** Test if a vertical coordinate is valid
00137
00138 \param[in] y is the vertical tile coordinate
00139 */
00140   static bool constexpr is_y_valid(int y) {
00141     return y_min <= y && y <= layout::y_max;
00142   }
00143
00144
00145 /** Validate a vertical coordinate
00146
00147 Do nothing but throwing if the coordinate is not a valid tile
00148 coordinate.
```

```
00149
00150 \param[in] y is the vertical tile coordinate
00151
00152 \throws trisycl::runtime_error if the coordinate is invalid
00153 */
00154   static void validate_y(int y) {
00155     if (!is_y_valid(y)) {
00156       throw ::trisycl::runtime_error {
00157         (boost::format { "The y coordinate %1% is not between %2% and %3%" }
00158           % y % y_min % layout::y_max).str() };
00159     }
00160   }
00161
00162
00163 /** Test if horizontal and vertical coordinates are valid
00164
00165 \param[in] x is the horizontal tile coordinate
00166
00167 \param[in] y is the vertical tile coordinate
00168 */
00169   static bool constexpr is_x_y_valid(int x, int y) {
00170     return is_x_valid(x) && is_y_valid(y);
00171   }
00172
00173 /** Validate the horizontal and vertical coordinates
00174
00175 Do nothing but throwing if the coordinates are not valid tile
00176 coordinates.
00177
00178 \param[in] x is the horizontal tile coordinate
00179
00180 \param[in] y is the vertical tile coordinate
00181
00182 \throws trisycl::runtime_error if the coordinate is invalid
00183 */
00184   static void validate_x_y(int x, int y) {
00185     validate_x(x);
00186     validate_y(y);
00187   }
00188
00189
00190 /** Test if a tile is in the Western column
00191
00192 \param[in] x is the horizontal tile coordinate
00193 */
00194   static bool constexpr is_west_column(int x) {
00195     return x == x_min;
00196   }
00197
00198
00199 /** Test if a tile is in the Eastern column
00200
00201 \param[in] x is the horizontal tile coordinate
00202 */
00203   static bool constexpr is_east_column(int x) {
00204     return x == layout::x_max;
00205   }
00206
00207
00208 /** Test if a tile is in the Southern row
00209
00210 \param[in] y is the vertical tile coordinate
00211 */
00212   static bool constexpr is_south_row(int y) {
00213     return y == y_min;
00214   }
00215
00216
00217 /** Test if a tile is in the Northern row
00218
00219 \param[in] y is the vertical tile coordinate
00220 */
00221   static bool constexpr is_north_row(int y) {
00222     return y == layout::y_max;
00223   }
00224
00225
00226 /** Test if a tile owns the start of the cascade_stream
00227
00228 \param[in] x is the horizontal tile coordinate
00229
00230 \param[in] y is the vertical tile coordinate
00231 */
00232   static bool constexpr is_cascade_start(int x, int y) {
00233     return x == x_min && y == y_min;
00234   }
00235
```

```
00236
00237 /** Test if a tile owns the end of the cascade_stream
00238
00239 \param[in] x is the horizontal tile coordinate
00240
00241 \param[in] y is the vertical tile coordinate
00242 */
00243   static bool constexpr is_cascade_end(int x, int y) {
00244     // The x position depends on the parity of the last line
00245     return x == ((y & 1) ?  x_min :  layout::x_max) && y == layout::y_max;
00246   }
00247
00248
00249 /** Compute the linearized coordinate of a tile along the cascade
00250 stream
00251
00252 \param[in] x is the horizontal tile coordinate
00253
00254 \param[in] y is the vertical tile coordinate
00255 */
00256   static auto constexpr cascade_linear_id(int x, int y) {
00257     return x_size*y + ((y & 1) ?  layout::x_max - x :  x);
00258   }
00259
00260
00261 /** Compute the horizontal coordinate of a tile from a cascade
00262 linear id
00263
00264 \param[in] linear_id is a linear id along the cascade stream
00265 */
00266   static auto constexpr cascade_linear_x(int linear_id) {
00267     auto x = linear_x(linear_id);
00268     // According to the row parity, symmetrize the x coordinate
00269     return (cascade_linear_y(linear_id) & 1) ?  layout::x_max - x :  x;
00270   }
00271
00272
00273 /** Compute the vertical coordinate of a tile from a cascade
00274 linear id
00275
00276 \param[in] linear_id is a linear id along the cascade stream
00277 */
00278   static auto constexpr cascade_linear_y(int linear_id) {
00279     // Same as the linear id, actually
00280     return linear_y(linear_id);
00281   }
00282
00283
00284 /** Test if a memory module coordinate offset is plausible
00285
00286 1 of the displacement is 0 and the other one is -1 or 1
00287
00288 \param[in] dx is the horizontal displacement (-1,0,+1)
00289
00290 \param[in] dy is the vertical displacement (-1,0,+1)
00291 */
00292   static bool constexpr is_valid_memory_module_offset(int dx, int dy) {
00293     if (dx == 0)
00294      // Cannot use std::abs() because it is not constexpr in C++17
00295       return dy == -1 || dy == 1;
00296     else if (dy == 0)
00297       return dx == -1 || dx == 1;
00298     else return false;
00299   }
00300
00301
00302 /** Test if a memory module exists and is connected to a tile
00303
00304 \param[in] x is the horizontal tile coordinate
00305
00306 \param[in] y is the vertical tile coordinate
00307
00308 \param[in] dx is the horizontal displacement (-1,0,+1)
00309
00310 \param[in] dy is the vertical displacement (-1,0,+1)
00311 */
00312   static bool constexpr is_memory_module(int x, int y, int dx, int dy) {
00313     if (!is_valid_memory_module_offset(dx, dy))
00314       return false;
00315     if (y == y_min && dy == -1)
00316       // No memory module on the South of the first (Southern) line
00317       return false;
00318     else if (y == layout::y_max && dy == 1)
00319       // No memory module on the North of the last (Northern) line
00320       return false;
00321     else if (x == x_min && !(y & 1) && (dx == -1))
00322       // No memory module on the West of the first (Western) column on
```

```
00323       // even lines
00324       return false;
00325     else if (x == layout::x_max && (y & 1) && (dx == 1))
00326       // No memory module on the East of the last (Eastern) column on
00327       // odd lines
00328       return false;
00329     else return true;
00330   }
00331
00332
00333 /** Compute the memory module x coordinate delta from its tile
00334
00335 \param[in] y is the vertical tile coordinate
00336
00337 \param[in] dx is the horizontal displacement (-1,0,+1)
00338 */
00339   static auto constexpr memory_module_dx_coordinate(int y, int dx) {
00340     // Mimic Figure 2-20 on p.  65
00341     return (dx - 1)/2 + (y & 1);
00342   }
00343
00344
00345 /** Compute the linear id of a memory module attached to a tile
00346
00347 \param[in] x is the horizontal tile coordinate
00348
00349 \param[in] y is the vertical tile coordinate
00350
00351 \param[in] dx is the horizontal displacement (-1,0,+1)
00352
00353 \param[in] dy is the vertical displacement (-1,0,+1)
00354 */
00355   static auto constexpr memory_module_linear_id(int x, int y, int dx, int dy) {
00356     if (dx == 0)
00357       return linear_id(x, y + dy);
00358     else
00359       // Take into account the horizontal chessboard-like skewing
00360       return linear_id(x + memory_module_dx_coordinate(y, dx), y);
00361   }
00362
00363
00364   // The organization of the AXI stream switch on a core tile
00365   struct core_axi_stream_switch {
00366 /// Default number of registers on default router paths
00367     static auto constexpr latency = 4;
00368
00369 /// Depth of additional FIFO on specific FIFO paths
00370     static auto constexpr fifo_depth = 16;
00371
00372 /** Layout of the AXI stream master ports in the switch
00373
00374 Revision v2.02, 4.2.1 Types of AXI-Streams, Table 4-2, p.  101 */
00375     enum class master_port_layout :  std::int8_t {
00376       me_0,
00377       me_1, me_last = me_1, ///< Used for user port validation
00378       dma_0,
00379       dma_1, dma_last = dma_1, ///< Used for DMA validation
00380       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation
00381       fifo_0,
00382       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00383       south_0,
00384       south_1,
00385       south_2,
00386       south_3, south_last = south_3, ///< Used for south validation
00387       west_0,
00388       west_1,
00389       west_2,
00390       west_3, west_last = west_3, ///< Used for west validation
00391       north_0,
00392       north_1,
00393       north_2,
00394       north_3,
00395       north_4,
00396       north_5, north_last = north_5, ///< Used for north validation
00397       east_0,
00398       east_1,
00399       east_2,
00400       east_3, east_last = east_3, ///< Used for east validation
00401       // To measure the enum
00402       size
00403     };
00404
00405   private:
00406
00407 /// Use a hidden friend to have introspection for master_port_layout
00408     friend bool constexpr is_axi_master(master_port_layout) {
00409       // Yes, master_port_layout is definitely an AXI master
```

```
00410       return true;
00411     }
00412
00413   public:
00414
00415 /// \todo factorize with a mix-in the following between master and slave
00416
00417 /// Number of AXI stream master ports in the switch
00418     static constexpr auto nb_master_port =
00419       detail::underlying_value(master_port_layout::size);
00420
00421 /// A range of the master ports connected to the tile inputs
00422     static auto inline m_me_range =
00423       views::enum_type(master_port_layout::me_0, master_port_layout::me_last);
00424
00425 /// Number of master ports connected to the tile inputs
00426     static constexpr auto m_me_size =
00427       detail::enum_count(master_port_layout::me_0, master_port_layout::me_last);
00428
00429 /// A range of the master ports connected to the DMA inputs
00430     static auto inline m_dma_range =
00431       views::enum_type(master_port_layout::dma_0, master_port_layout::dma_last);
00432
00433 /// Number of master ports connected to the DMA inputs
00434     static constexpr auto m_dma_size =
00435       detail::enum_count(master_port_layout::dma_0,
00436                          master_port_layout::dma_last);
00437
00438 /// A range of the master ports connected to the tile control inputs
00439     static auto inline m_tile_ctrl_range =
00440       views::enum_type(master_port_layout::tile_ctrl,
00441                        master_port_layout::tile_ctrl_last);
00442
00443 /// Number of master ports connected to the tile control inputs
00444     static constexpr auto m_tile_ctrl_size =
00445       detail::enum_count(master_port_layout::tile_ctrl,
00446                          master_port_layout::tile_ctrl_last);
00447
00448 /// A range of the master ports connected to the FIFO inputs
00449     static auto inline m_fifo_range =
00450       views::enum_type(master_port_layout::fifo_0,
00451                        master_port_layout::fifo_last);
00452
00453 /// Number of master ports connected to the FIFO inputs
00454     static constexpr auto m_fifo_size =
00455       detail::enum_count(master_port_layout::fifo_0,
00456                          master_port_layout::fifo_last);
00457
00458 /// A range of the South master ports
00459     static auto inline m_south_range =
00460       views::enum_type(master_port_layout::south_0,
00461                        master_port_layout::south_last);
00462
00463 /// Number of South master ports
00464     static constexpr auto m_south_size =
00465       detail::enum_count(master_port_layout::south_0,
00466                          master_port_layout::south_last);
00467
00468 /// A range of the West master ports
00469     static auto inline m_west_range =
00470       views::enum_type(master_port_layout::west_0,
00471                        master_port_layout::west_last);
00472
00473 /// Number of West master ports
00474     static constexpr auto m_west_size =
00475       detail::enum_count(master_port_layout::west_0,
00476                          master_port_layout::west_last);
00477
00478 /// A range of the North master ports
00479     static auto inline m_north_range =
00480       views::enum_type(master_port_layout::north_0,
00481                        master_port_layout::north_last);
00482
00483 /// Number of North master ports
00484     static constexpr auto m_north_size =
00485       detail::enum_count(master_port_layout::north_0,
00486                          master_port_layout::north_last);
00487
00488 /// A range of the East master ports
00489     static auto inline m_east_range =
00490       views::enum_type(master_port_layout::east_0,
00491                        master_port_layout::east_last);
00492
00493 /// Number of East master ports
00494     static constexpr auto m_east_size =
00495       detail::enum_count(master_port_layout::east_0,
00496                          master_port_layout::east_last);
```

```
00497
00498 /** Layout of the AXI stream slave ports in the switch
00499
00500 Revision v2.02, 4.2.1 Types of AXI-Streams, Table 4-3, p.  102 */
00501     enum class slave_port_layout :  std::int8_t {
00502       me_0,
00503       me_1, me_last = me_1, ///< Used for user port validation
00504       dma_0,
00505       dma_1, dma_last = dma_1, ///< Used for DMA validation
00506       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation,
00507       fifo_0,
00508       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00509       south_0,
00510       south_1,
00511       south_2,
00512       south_3,
00513       south_4,
00514       south_5, south_last = south_5, ///< Used for south validation
00515       west_0,
00516       west_1,
00517       west_2,
00518       west_3, west_last = west_3, ///< Used for west validation
00519       north_0,
00520       north_1,
00521       north_2,
00522       north_3, north_last = north_3, ///< Used for north validation
00523       east_0,
00524       east_1,
00525       east_2,
00526       east_3, east_last = east_3, ///< Used for east validation
00527       core_trace,
00528       mem_trace,
00529       // To measure the enum
00530       size
00531     };
00532
00533   private:
00534
00535 /// Use a hidden friend to have introspection for slave_port_layout
00536     friend bool constexpr is_axi_master(slave_port_layout) {
00537       // No, slave_port_layout is definitely not an AXI master
00538       return false;
00539     }
00540
00541   public:
00542
00543 /// Number of AXI stream slave ports in the switch
00544     static constexpr auto nb_slave_port =
00545       detail::underlying_value(slave_port_layout::size);
00546
00547 /// A range of the slave ports connected to the tile outputs
00548     static auto inline s_me_range =
00549       views::enum_type(slave_port_layout::me_0, slave_port_layout::me_last);
00550
00551 /// Number of slave ports connected to the tile outputs
00552     static constexpr auto s_me_size =
00553       detail::enum_count(slave_port_layout::me_0, slave_port_layout::me_last);
00554
00555 /// A range of the slave ports connected to the DMA outputs
00556     static auto inline s_dma_range =
00557       views::enum_type(slave_port_layout::dma_0, slave_port_layout::dma_last);
00558
00559 /// Number of slave ports connected to the DMA outputs
00560     static constexpr auto s_dma_size =
00561       detail::enum_count(slave_port_layout::dma_0, slave_port_layout::dma_last);
00562
00563 /// A range of the slave ports connected to the tile control outputs
00564     static auto inline s_tile_ctrl_range =
00565       views::enum_type(slave_port_layout::tile_ctrl,
00566                        slave_port_layout::tile_ctrl_last);
00567
00568 /// Number of slave ports connected to the tile control outputs
00569     static constexpr auto s_tile_ctrl_size =
00570       detail::enum_count(slave_port_layout::tile_ctrl,
00571                          slave_port_layout::tile_ctrl_last);
00572
00573 /// A range of the slave ports connected to the FIFO outputs
00574     static auto inline s_fifo_range =
00575       views::enum_type(slave_port_layout::fifo_0, slave_port_layout::fifo_last);
00576
00577 /// Number of slave ports connected to the FIFO outputs
00578     static constexpr auto s_fifo_size =
00579       detail::enum_count(slave_port_layout::fifo_0,
00580                          slave_port_layout::fifo_last);
00581
00582 /// A range of the South slave ports
00583     static auto inline s_south_range =
```

```
00584       views::enum_type(slave_port_layout::south_0,
00585                        slave_port_layout::south_last);
00586
00587 /// Number of South slave ports
00588     static constexpr auto s_south_size =
00589       detail::enum_count(slave_port_layout::south_0,
00590                          slave_port_layout::south_last);
00591
00592 /// A range of the West slave ports
00593     static auto inline s_west_range =
00594       views::enum_type(slave_port_layout::west_0, slave_port_layout::west_last);
00595
00596 /// Number of West slave ports
00597     static constexpr auto s_west_size =
00598       detail::enum_count(slave_port_layout::west_0,
00599                          slave_port_layout::west_last);
00600
00601 /// A range of the North slave ports
00602     static auto inline s_north_range =
00603       views::enum_type(slave_port_layout::north_0,
00604                        slave_port_layout::north_last);
00605
00606 /// Number of North slave ports
00607     static constexpr auto s_north_size =
00608       detail::enum_count(slave_port_layout::north_0,
00609                          slave_port_layout::north_last);
00610
00611 /// A range of the East slave ports
00612     static auto inline s_east_range =
00613       views::enum_type(slave_port_layout::east_0, slave_port_layout::east_last);
00614
00615 /// Number of East slave ports
00616     static constexpr auto s_east_size =
00617       detail::enum_count(slave_port_layout::east_0,
00618                          slave_port_layout::east_last);
00619
00620 /// Describe the AXI stream switch interconnection neighborhood
00621     static auto inline interconnect = boost::hana::make_tuple(
00622         // Connection topology of the NoC towards East of the switches
00623         std::tuple { 1, 0, m_east_range, s_west_range }
00624         // Connection topology of the NoC towards West of the switches
00625       , std::tuple { -1, 0, m_west_range, s_east_range }
00626         // Connection topology of the NoC towards North of the switches
00627       , std::tuple { 0, 1, m_north_range, s_south_range }
00628         // Connection topology of the NoC towards South of the switches
00629       , std::tuple { 0, -1, m_south_range, s_north_range }
00630       );
00631
00632
00633   };
00634
00635   // The organization of the AXI stream switch on a shim tile
00636   struct shim_axi_stream_switch {
00637 /// \todo factorize the 2 following values with core tiles?
00638 /// Default number of registers on default router paths
00639     static auto constexpr latency = 4;
00640
00641 /// Depth of additional FIFO on specific FIFO paths
00642     static auto constexpr fifo_depth = 16;
00643
00644 /** Layout of the AXI stream master ports in the switch
00645
00646 Revision v2.02, 6.5.3 Shim AXI-Stream Interconnect, Table 6-2, p.  291 */
00647     enum class master_port_layout :  std::int8_t {
00648       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation,
00649       fifo_0,
00650       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00651       south_0,
00652       south_1,
00653       south_2,
00654       south_3,
00655       south_4,
00656       south_5, south_last = south_5, ///< Used for shim BLI port validation
00657       west_0,
00658       west_1,
00659       west_2,
00660       west_3, west_last = west_3, ///< Used for west validation
00661       north_0,
00662       north_1,
00663       north_2,
00664       north_3,
00665       north_4,
00666       north_5, north_last = north_5, ///< Used for north validation
00667       east_0,
00668       east_1,
00669       east_2,
00670       east_3, east_last = east_3, ///< Used for east validation
```

```
00671       // To measure the enum
00672       size
00673     };
00674
00675   private:
00676
00677 /// \todo Factorize out
00678 /// Use a hidden friend to have introspection for master_port_layout
00679     friend bool constexpr is_axi_master(master_port_layout) {
00680       // Yes, master_port_layout is definitely an AXI master
00681       return true;
00682     }
00683
00684   public:
00685
00686 /// \todo factorize with a mix-in the following between master and slave
00687
00688 /// Number of AXI stream master ports in the switch
00689     static constexpr auto nb_master_port =
00690       detail::underlying_value(master_port_layout::size);
00691
00692 /// \todo Factorize out this range code with core tile
00693
00694 /// A range of the master ports connected to the tile control inputs
00695     static auto inline m_tile_ctrl_range =
00696       views::enum_type(master_port_layout::tile_ctrl,
00697                        master_port_layout::tile_ctrl_last);
00698
00699 /// Number of master ports connected to the tile control inputs
00700     static constexpr auto m_tile_ctrl_size =
00701       detail::enum_count(master_port_layout::tile_ctrl,
00702                          master_port_layout::tile_ctrl_last);
00703
00704 /// A range of the master ports connected to the FIFO inputs
00705     static auto inline m_fifo_range =
00706       views::enum_type(master_port_layout::fifo_0,
00707                        master_port_layout::fifo_last);
00708
00709 /// Number of master ports connected to the FIFO inputs
00710     static constexpr auto m_fifo_size =
00711       detail::enum_count(master_port_layout::fifo_0,
00712                          master_port_layout::fifo_last);
00713
00714 /// A range of the South master ports
00715     static auto inline m_south_range =
00716       views::enum_type(master_port_layout::south_0,
00717                        master_port_layout::south_last);
00718
00719 /// Number of South master ports
00720     static constexpr auto m_south_size =
00721       detail::enum_count(master_port_layout::south_0,
00722                          master_port_layout::south_last);
00723
00724 /// A range of the West master ports
00725     static auto inline m_west_range =
00726       views::enum_type(master_port_layout::west_0,
00727                        master_port_layout::west_last);
00728
00729 /// Number of West master ports
00730     static constexpr auto m_west_size =
00731       detail::enum_count(master_port_layout::west_0,
00732                          master_port_layout::west_last);
00733
00734 /// A range of the North master ports
00735     static auto inline m_north_range =
00736       views::enum_type(master_port_layout::north_0,
00737                        master_port_layout::north_last);
00738
00739 /// Number of North master ports
00740     static constexpr auto m_north_size =
00741       detail::enum_count(master_port_layout::north_0,
00742                          master_port_layout::north_last);
00743
00744 /// A range of the East master ports
00745     static auto inline m_east_range =
00746       views::enum_type(master_port_layout::east_0,
00747                        master_port_layout::east_last);
00748
00749 /// Number of East master ports
00750     static constexpr auto m_east_size =
00751       detail::enum_count(master_port_layout::east_0,
00752                          master_port_layout::east_last);
00753
00754 /** Layout of the AXI stream slave ports in the switch
00755
00756 Revision v2.02, 6.5.3 Shim AXI-Stream Interconnect, Table 6-3, p.  291 */
00757     enum class slave_port_layout :  std::int8_t {
```

```
00758       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation,
00759       fifo_0,
00760       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00761       south_0,
00762       south_1,
00763       south_2,
00764       south_3,
00765       south_4,
00766       south_5,
00767       south_6,
00768       south_7, south_last = south_7, ///< Used for shim BLI port validation
00769       west_0,
00770       west_1,
00771       west_2,
00772       west_3, west_last = west_3, ///< Used for west validation
00773       north_0,
00774       north_1,
00775       north_2,
00776       north_3, north_last = north_3, ///< Used for north validation
00777       east_0,
00778       east_1,
00779       east_2,
00780       east_3, east_last = east_3, ///< Used for east validation
00781       shim_trace,
00782       // To measure the enum
00783       size
00784     };
00785
00786   private:
00787
00788 /// Use a hidden friend to have introspection for slave_port_layout
00789     friend bool constexpr is_axi_master(slave_port_layout) {
00790       // No, slave_port_layout is definitely not an AXI master
00791       return false;
00792     }
00793
00794   public:
00795
00796 /// Number of master AXI stream slave ports in the switch
00797     static constexpr auto nb_slave_port =
00798       static_cast<int>(slave_port_layout::size);
00799
00800 /// A range of the slave ports connected to the tile control outputs
00801     static auto inline s_tile_ctrl_range =
00802       views::enum_type(slave_port_layout::tile_ctrl,
00803                        slave_port_layout::tile_ctrl_last);
00804
00805 /// Number of slave ports connected to the tile control outputs
00806     static constexpr auto s_tile_ctrl_size =
00807       detail::enum_count(slave_port_layout::tile_ctrl,
00808                          slave_port_layout::tile_ctrl_last);
00809
00810 /// A range of the slave ports connected to the FIFO outputs
00811     static auto inline s_fifo_range =
00812       views::enum_type(slave_port_layout::fifo_0, slave_port_layout::fifo_last);
00813
00814 /// Number of slave ports connected to the FIFO outputs
00815     static constexpr auto s_fifo_size =
00816       detail::enum_count(slave_port_layout::fifo_0,
00817                          slave_port_layout::fifo_last);
00818
00819 /// A range of the South slave ports
00820     static auto inline s_south_range =
00821       views::enum_type(slave_port_layout::south_0,
00822                        slave_port_layout::south_last);
00823
00824 /// Number of South slave ports
00825     static constexpr auto s_south_size =
00826       detail::enum_count(slave_port_layout::south_0,
00827                          slave_port_layout::south_last);
00828
00829 /// A range of the West slave ports
00830     static auto inline s_west_range =
00831       views::enum_type(slave_port_layout::west_0, slave_port_layout::west_last);
00832
00833 /// Number of West slave ports
00834     static constexpr auto s_west_size =
00835       detail::enum_count(slave_port_layout::west_0,
00836                          slave_port_layout::west_last);
00837
00838 /// A range of the North slave ports
00839     static auto inline s_north_range =
00840       views::enum_type(slave_port_layout::north_0,
00841                        slave_port_layout::north_last);
00842
00843 /// Number of North slave ports
00844     static constexpr auto s_north_size =
```

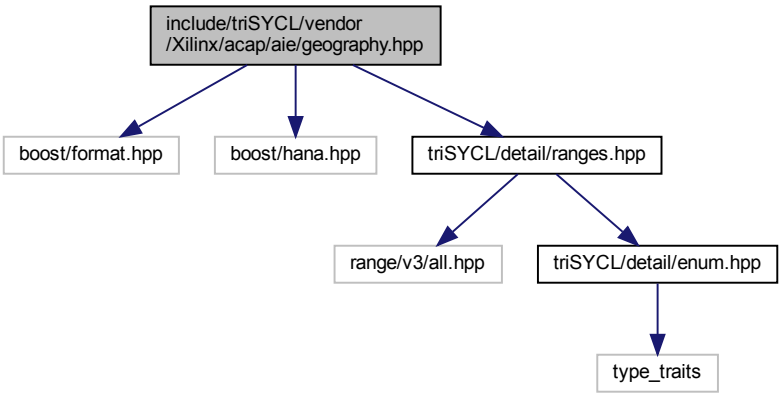
```
00845       detail::enum_count(slave_port_layout::north_0,
00846                          slave_port_layout::north_last);
00847
00848 /// A range of the East slave ports
00849     static auto inline s_east_range =
00850       views::enum_type(slave_port_layout::east_0, slave_port_layout::east_last);
00851
00852 /// Number of East slave ports
00853     static constexpr auto s_east_size =
00854       detail::enum_count(slave_port_layout::east_0,
00855                          slave_port_layout::east_last);
00856
00857   };
00858
00859 };
00860
00861 /// @} End the aie Doxygen group
00862
00863 }
00864
00865 /*
00866 # Some Emacs stuff:
00867 ### Local Variables:
00868 ### ispell-local-dictionary:  "american"
00869 ### eval:  (flyspell-prog-mode)
00870 ### End:
00871 */
00872
00873 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_GEOGRAPHY_HPP
```

13.295 include/triSYCL/vendor/Xilinx/acap/aie/geography.hpp File Reference

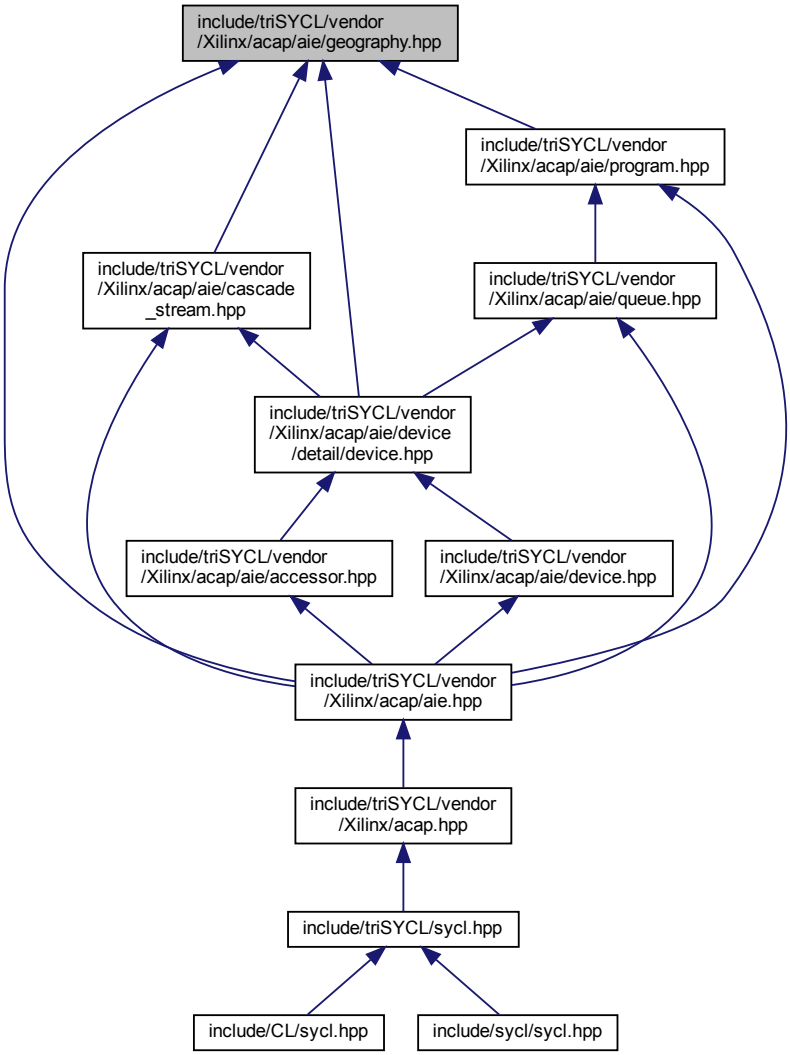
Geographic information about the AI Engine CGRA.

```
#include <boost/format.hpp>
#include <boost/hana.hpp>
#include "triSYCL/detail/ranges.hpp"
```

Include dependency graph for geography.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::geography< Layout >

  Some geographic information about the AI Engine array. More...
- struct trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch
- struct trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.295.1 Detailed Description

Geographic information about the AI Engine CGRA.

Define in a central location most of the ACAP architectural parameters.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file geography.hpp.

13.296 geography.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_GEOGRAPHY_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_GEOGRAPHY_HPP
00003
00004 /** \file
00005
00006 Geographic information about the AI Engine CGRA
00007
00008 Define in a central location most of the ACAP architectural
00009 parameters.
00010
00011 Ronan dot Keryell at Xilinx dot com
00012
00013 This file is distributed under the University of Illinois Open Source
00014 License.  See LICENSE.TXT for details.
00015 */
00016
00017 #include <boost/format.hpp>
00018 #include <boost/hana.hpp>
00019
00020 #include "triSYCL/detail/ranges.hpp"
00021
00022 namespace trisycl::vendor::xilinx::acap::aie {
00023
00024 /// \ingroup aie
00025 /// @{
00026
00027 /** Some geographic information about the AI Engine array
00028
00029 \param Layout is a class describing the basic layout of the CGRA,
00030 setting for example the x & y size
00031 */
00032 template <typename Layout>
00033 struct geography :  Layout {
00034 /// Keep the layout used for this CGRA
00035   using layout = Layout;
00036
00037 /// Minimum horizontal coordinate
00038   static auto constexpr x_min = 0;
00039
00040 /// Minimum vertical coordinate
00041   static auto constexpr y_min = 0;
00042
00043 /// Number of tile horizontally in the CGRA
00044   static auto constexpr x_size = layout::x_max + 1;
00045
00046 /// Number of tile vertically in the CGRA
00047   static auto constexpr y_size = layout::y_max + 1;
00048
00049 /// Total number of tiles in the CGRA
00050   static auto constexpr size = x_size*y_size;
00051
00052
00053 /** A tuple with the coordinate tuples of all the tiles inside the
00054 CGRA geometry */
00055   static auto constexpr tile_coordinates = boost::hana::cartesian_product(
00056     boost::hana::make_tuple(
00057         boost::hana::range_c<int, y_min, y_size>
00058       , boost::hana::range_c<int, x_min, x_size>
00059                             )
00060                                                                     );
00061
```

```
00062
00063 /** Generate a tuple by tiling the Tileable object onto the CGRA geometry
00064
00065 \param Tileable is the type to tile according to coordinate \p X
00066 and \p Y
00067 */
00068   template <template <int X, int Y> typename Tileable>
00069   static auto generate_tiles() {
00070     return boost::hana::transform(
00071         tile_coordinates
00072       , [] (auto coord) {
00073           return Tileable<boost::hana::at_c<1>(coord),
00074                           boost::hana::at_c<0>(coord)> {};
00075         }
00076                                   );
00077   }
00078
00079
00080 /** Compute the linearized coordinate of a tile
00081
00082 \param[in] x is the horizontal tile coordinate
00083
00084 \param[in] y is the vertical tile coordinate
00085 */
00086   static auto constexpr linear_id(int x, int y) {
00087     return x + x_size*y;
00088   }
00089
00090
00091 /** Compute the horizontal coordinate of a tile from linear id
00092
00093 \param[in] linear_id is a linearized coordinate
00094 */
00095   static auto constexpr linear_x(int linear_id) {
00096     return linear_id%x_size;
00097   }
00098
00099
00100 /** Compute the vertical coordinate of a tile from a linear id
00101
00102 \param[in] linear_id is a linearized coordinate
00103 */
00104   static auto constexpr linear_y(int linear_id) {
00105     return linear_id/x_size;
00106   }
00107
00108
00109 /** Test if an horizontal coordinate is valid
00110
00111 \param[in] x is the horizontal tile coordinate
00112 */
00113   static bool constexpr is_x_valid(int x) {
00114     return x_min <= x && x <= layout::x_max;
00115   }
00116
00117
00118 /** Validate an horizontal coordinate
00119
00120 Do nothing but throwing if the coordinate is not a valid tile
00121 coordinate.
00122
00123 \param[in] x is the horizontal tile coordinate
00124
00125 \throws trisycl::runtime_error if the coordinate is invalid
00126 */
00127   static void validate_x(int x) {
00128     if (!is_x_valid(x)) {
00129       throw ::trisycl::runtime_error {
00130         (boost::format { "The x coordinate %1% is not between %2% and %3%" }
00131           % x % x_min % layout::x_max).str() };
00132     }
00133   }
00134
00135
00136 /** Test if a vertical coordinate is valid
00137
00138 \param[in] y is the vertical tile coordinate
00139 */
00140   static bool constexpr is_y_valid(int y) {
00141     return y_min <= y && y <= layout::y_max;
00142   }
00143
00144
00145 /** Validate a vertical coordinate
00146
00147 Do nothing but throwing if the coordinate is not a valid tile
00148 coordinate.
```

```
00149
00150 \param[in] y is the vertical tile coordinate
00151
00152 \throws trisycl::runtime_error if the coordinate is invalid
00153 */
00154   static void validate_y(int y) {
00155     if (!is_y_valid(y)) {
00156       throw ::trisycl::runtime_error {
00157         (boost::format { "The y coordinate %1% is not between %2% and %3%" }
00158           % y % y_min % layout::y_max).str() };
00159     }
00160   }
00161
00162
00163 /** Test if horizontal and vertical coordinates are valid
00164
00165 \param[in] x is the horizontal tile coordinate
00166
00167 \param[in] y is the vertical tile coordinate
00168 */
00169   static bool constexpr is_x_y_valid(int x, int y) {
00170     return is_x_valid(x) && is_y_valid(y);
00171   }
00172
00173 /** Validate the horizontal and vertical coordinates
00174
00175 Do nothing but throwing if the coordinates are not valid tile
00176 coordinates.
00177
00178 \param[in] x is the horizontal tile coordinate
00179
00180 \param[in] y is the vertical tile coordinate
00181
00182 \throws trisycl::runtime_error if the coordinate is invalid
00183 */
00184   static void validate_x_y(int x, int y) {
00185     validate_x(x);
00186     validate_y(y);
00187   }
00188
00189
00190 /** Test if a tile is in the Western column
00191
00192 \param[in] x is the horizontal tile coordinate
00193 */
00194   static bool constexpr is_west_column(int x) {
00195     return x == x_min;
00196   }
00197
00198
00199 /** Test if a tile is in the Eastern column
00200
00201 \param[in] x is the horizontal tile coordinate
00202 */
00203   static bool constexpr is_east_column(int x) {
00204     return x == layout::x_max;
00205   }
00206
00207
00208 /** Test if a tile is in the Southern row
00209
00210 \param[in] y is the vertical tile coordinate
00211 */
00212   static bool constexpr is_south_row(int y) {
00213     return y == y_min;
00214   }
00215
00216
00217 /** Test if a tile is in the Northern row
00218
00219 \param[in] y is the vertical tile coordinate
00220 */
00221   static bool constexpr is_north_row(int y) {
00222     return y == layout::y_max;
00223   }
00224
00225
00226 /** Test if a tile owns the start of the cascade_stream
00227
00228 \param[in] x is the horizontal tile coordinate
00229
00230 \param[in] y is the vertical tile coordinate
00231 */
00232   static bool constexpr is_cascade_start(int x, int y) {
00233     return x == x_min && y == y_min;
00234   }
00235
```

```
00236
00237 /** Test if a tile owns the end of the cascade_stream
00238
00239 \param[in] x is the horizontal tile coordinate
00240
00241 \param[in] y is the vertical tile coordinate
00242 */
00243   static bool constexpr is_cascade_end(int x, int y) {
00244     // The x position depends on the parity of the last line
00245     return x == ((y & 1) ?  x_min :  layout::x_max) && y == layout::y_max;
00246   }
00247
00248
00249 /** Compute the linearized coordinate of a tile along the cascade
00250 stream
00251
00252 \param[in] x is the horizontal tile coordinate
00253
00254 \param[in] y is the vertical tile coordinate
00255 */
00256   static auto constexpr cascade_linear_id(int x, int y) {
00257     return x_size*y + ((y & 1) ?  layout::x_max - x :  x);
00258   }
00259
00260
00261 /** Compute the horizontal coordinate of a tile from a cascade
00262 linear id
00263
00264 \param[in] linear_id is a linear id along the cascade stream
00265 */
00266   static auto constexpr cascade_linear_x(int linear_id) {
00267     auto x = linear_x(linear_id);
00268     // According to the row parity, symmetrize the x coordinate
00269     return (cascade_linear_y(linear_id) & 1) ?  layout::x_max - x :  x;
00270   }
00271
00272
00273 /** Compute the vertical coordinate of a tile from a cascade
00274 linear id
00275
00276 \param[in] linear_id is a linear id along the cascade stream
00277 */
00278   static auto constexpr cascade_linear_y(int linear_id) {
00279     // Same as the linear id, actually
00280     return linear_y(linear_id);
00281   }
00282
00283
00284 /** Test if a memory module coordinate offset is plausible
00285
00286 1 of the displacement is 0 and the other one is -1 or 1
00287
00288 \param[in] dx is the horizontal displacement (-1,0,+1)
00289
00290 \param[in] dy is the vertical displacement (-1,0,+1)
00291 */
00292   static bool constexpr is_valid_memory_module_offset(int dx, int dy) {
00293     if (dx == 0)
00294      // Cannot use std::abs() because it is not constexpr in C++17
00295       return dy == -1 || dy == 1;
00296     else if (dy == 0)
00297       return dx == -1 || dx == 1;
00298     else return false;
00299   }
00300
00301
00302 /** Test if a memory module exists and is connected to a tile
00303
00304 \param[in] x is the horizontal tile coordinate
00305
00306 \param[in] y is the vertical tile coordinate
00307
00308 \param[in] dx is the horizontal displacement (-1,0,+1)
00309
00310 \param[in] dy is the vertical displacement (-1,0,+1)
00311 */
00312   static bool constexpr is_memory_module(int x, int y, int dx, int dy) {
00313     if (!is_valid_memory_module_offset(dx, dy))
00314       return false;
00315     if (y == y_min && dy == -1)
00316       // No memory module on the South of the first (Southern) line
00317       return false;
00318     else if (y == layout::y_max && dy == 1)
00319       // No memory module on the North of the last (Northern) line
00320       return false;
00321     else if (x == x_min && !(y & 1) && (dx == -1))
00322       // No memory module on the West of the first (Western) column on
```

```
00323       // even lines
00324       return false;
00325     else if (x == layout::x_max && (y & 1) && (dx == 1))
00326       // No memory module on the East of the last (Eastern) column on
00327       // odd lines
00328       return false;
00329     else return true;
00330   }
00331
00332
00333 /** Compute the memory module x coordinate delta from its tile
00334
00335 \param[in] y is the vertical tile coordinate
00336
00337 \param[in] dx is the horizontal displacement (-1,0,+1)
00338 */
00339   static auto constexpr memory_module_dx_coordinate(int y, int dx) {
00340     // Mimic Figure 2-20 on p.  65
00341     return (dx - 1)/2 + (y & 1);
00342   }
00343
00344
00345 /** Compute the linear id of a memory module attached to a tile
00346
00347 \param[in] x is the horizontal tile coordinate
00348
00349 \param[in] y is the vertical tile coordinate
00350
00351 \param[in] dx is the horizontal displacement (-1,0,+1)
00352
00353 \param[in] dy is the vertical displacement (-1,0,+1)
00354 */
00355   static auto constexpr memory_module_linear_id(int x, int y, int dx, int dy) {
00356     if (dx == 0)
00357       return linear_id(x, y + dy);
00358     else
00359       // Take into account the horizontal chessboard-like skewing
00360       return linear_id(x + memory_module_dx_coordinate(y, dx), y);
00361   }
00362
00363
00364   // The organization of the AXI stream switch on a core tile
00365   struct core_axi_stream_switch {
00366 /// Default number of registers on default router paths
00367     static auto constexpr latency = 4;
00368
00369 /// Depth of additional FIFO on specific FIFO paths
00370     static auto constexpr fifo_depth = 16;
00371
00372 /** Layout of the AXI stream master ports in the switch
00373
00374 Revision v2.02, 4.2.1 Types of AXI-Streams, Table 4-2, p.  101 */
00375     enum class master_port_layout :  std::int8_t {
00376       me_0,
00377       me_1, me_last = me_1, ///< Used for user port validation
00378       dma_0,
00379       dma_1, dma_last = dma_1, ///< Used for DMA validation
00380       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation
00381       fifo_0,
00382       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00383       south_0,
00384       south_1,
00385       south_2,
00386       south_3, south_last = south_3, ///< Used for south validation
00387       west_0,
00388       west_1,
00389       west_2,
00390       west_3, west_last = west_3, ///< Used for west validation
00391       north_0,
00392       north_1,
00393       north_2,
00394       north_3,
00395       north_4,
00396       north_5, north_last = north_5, ///< Used for north validation
00397       east_0,
00398       east_1,
00399       east_2,
00400       east_3, east_last = east_3, ///< Used for east validation
00401       // To measure the enum
00402       size
00403     };
00404
00405   private:
00406
00407 /// Use a hidden friend to have introspection for master_port_layout
00408     friend bool constexpr is_axi_master(master_port_layout) {
00409       // Yes, master_port_layout is definitely an AXI master
```

```
00410       return true;
00411     }
00412
00413   public:
00414
00415 /// \todo factorize with a mix-in the following between master and slave
00416
00417 /// Number of AXI stream master ports in the switch
00418     static constexpr auto nb_master_port =
00419       detail::underlying_value(master_port_layout::size);
00420
00421 /// A range of the master ports connected to the tile inputs
00422     static auto inline m_me_range =
00423       views::enum_type(master_port_layout::me_0, master_port_layout::me_last);
00424
00425 /// Number of master ports connected to the tile inputs
00426     static constexpr auto m_me_size =
00427       detail::enum_count(master_port_layout::me_0, master_port_layout::me_last);
00428
00429 /// A range of the master ports connected to the DMA inputs
00430     static auto inline m_dma_range =
00431       views::enum_type(master_port_layout::dma_0, master_port_layout::dma_last);
00432
00433 /// Number of master ports connected to the DMA inputs
00434     static constexpr auto m_dma_size =
00435       detail::enum_count(master_port_layout::dma_0,
00436                          master_port_layout::dma_last);
00437
00438 /// A range of the master ports connected to the tile control inputs
00439     static auto inline m_tile_ctrl_range =
00440       views::enum_type(master_port_layout::tile_ctrl,
00441                        master_port_layout::tile_ctrl_last);
00442
00443 /// Number of master ports connected to the tile control inputs
00444     static constexpr auto m_tile_ctrl_size =
00445       detail::enum_count(master_port_layout::tile_ctrl,
00446                          master_port_layout::tile_ctrl_last);
00447
00448 /// A range of the master ports connected to the FIFO inputs
00449     static auto inline m_fifo_range =
00450       views::enum_type(master_port_layout::fifo_0,
00451                        master_port_layout::fifo_last);
00452
00453 /// Number of master ports connected to the FIFO inputs
00454     static constexpr auto m_fifo_size =
00455       detail::enum_count(master_port_layout::fifo_0,
00456                          master_port_layout::fifo_last);
00457
00458 /// A range of the South master ports
00459     static auto inline m_south_range =
00460       views::enum_type(master_port_layout::south_0,
00461                        master_port_layout::south_last);
00462
00463 /// Number of South master ports
00464     static constexpr auto m_south_size =
00465       detail::enum_count(master_port_layout::south_0,
00466                          master_port_layout::south_last);
00467
00468 /// A range of the West master ports
00469     static auto inline m_west_range =
00470       views::enum_type(master_port_layout::west_0,
00471                        master_port_layout::west_last);
00472
00473 /// Number of West master ports
00474     static constexpr auto m_west_size =
00475       detail::enum_count(master_port_layout::west_0,
00476                          master_port_layout::west_last);
00477
00478 /// A range of the North master ports
00479     static auto inline m_north_range =
00480       views::enum_type(master_port_layout::north_0,
00481                        master_port_layout::north_last);
00482
00483 /// Number of North master ports
00484     static constexpr auto m_north_size =
00485       detail::enum_count(master_port_layout::north_0,
00486                          master_port_layout::north_last);
00487
00488 /// A range of the East master ports
00489     static auto inline m_east_range =
00490       views::enum_type(master_port_layout::east_0,
00491                        master_port_layout::east_last);
00492
00493 /// Number of East master ports
00494     static constexpr auto m_east_size =
00495       detail::enum_count(master_port_layout::east_0,
00496                          master_port_layout::east_last);
```

```
00497
00498 /** Layout of the AXI stream slave ports in the switch
00499
00500 Revision v2.02, 4.2.1 Types of AXI-Streams, Table 4-3, p.  102 */
00501     enum class slave_port_layout :  std::int8_t {
00502       me_0,
00503       me_1, me_last = me_1, ///< Used for user port validation
00504       dma_0,
00505       dma_1, dma_last = dma_1, ///< Used for DMA validation
00506       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation,
00507       fifo_0,
00508       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00509       south_0,
00510       south_1,
00511       south_2,
00512       south_3,
00513       south_4,
00514       south_5, south_last = south_5, ///< Used for south validation
00515       west_0,
00516       west_1,
00517       west_2,
00518       west_3, west_last = west_3, ///< Used for west validation
00519       north_0,
00520       north_1,
00521       north_2,
00522       north_3, north_last = north_3, ///< Used for north validation
00523       east_0,
00524       east_1,
00525       east_2,
00526       east_3, east_last = east_3, ///< Used for east validation
00527       core_trace,
00528       mem_trace,
00529       // To measure the enum
00530       size
00531     };
00532
00533   private:
00534
00535 /// Use a hidden friend to have introspection for slave_port_layout
00536     friend bool constexpr is_axi_master(slave_port_layout) {
00537       // No, slave_port_layout is definitely not an AXI master
00538       return false;
00539     }
00540
00541   public:
00542
00543 /// Number of AXI stream slave ports in the switch
00544     static constexpr auto nb_slave_port =
00545       detail::underlying_value(slave_port_layout::size);
00546
00547 /// A range of the slave ports connected to the tile outputs
00548     static auto inline s_me_range =
00549       views::enum_type(slave_port_layout::me_0, slave_port_layout::me_last);
00550
00551 /// Number of slave ports connected to the tile outputs
00552     static constexpr auto s_me_size =
00553       detail::enum_count(slave_port_layout::me_0, slave_port_layout::me_last);
00554
00555 /// A range of the slave ports connected to the DMA outputs
00556     static auto inline s_dma_range =
00557       views::enum_type(slave_port_layout::dma_0, slave_port_layout::dma_last);
00558
00559 /// Number of slave ports connected to the DMA outputs
00560     static constexpr auto s_dma_size =
00561       detail::enum_count(slave_port_layout::dma_0, slave_port_layout::dma_last);
00562
00563 /// A range of the slave ports connected to the tile control outputs
00564     static auto inline s_tile_ctrl_range =
00565       views::enum_type(slave_port_layout::tile_ctrl,
00566                        slave_port_layout::tile_ctrl_last);
00567
00568 /// Number of slave ports connected to the tile control outputs
00569     static constexpr auto s_tile_ctrl_size =
00570       detail::enum_count(slave_port_layout::tile_ctrl,
00571                          slave_port_layout::tile_ctrl_last);
00572
00573 /// A range of the slave ports connected to the FIFO outputs
00574     static auto inline s_fifo_range =
00575       views::enum_type(slave_port_layout::fifo_0, slave_port_layout::fifo_last);
00576
00577 /// Number of slave ports connected to the FIFO outputs
00578     static constexpr auto s_fifo_size =
00579       detail::enum_count(slave_port_layout::fifo_0,
00580                          slave_port_layout::fifo_last);
00581
00582 /// A range of the South slave ports
00583     static auto inline s_south_range =
```

```
00584       views::enum_type(slave_port_layout::south_0,
00585                        slave_port_layout::south_last);
00586
00587 /// Number of South slave ports
00588     static constexpr auto s_south_size =
00589       detail::enum_count(slave_port_layout::south_0,
00590                          slave_port_layout::south_last);
00591
00592 /// A range of the West slave ports
00593     static auto inline s_west_range =
00594       views::enum_type(slave_port_layout::west_0, slave_port_layout::west_last);
00595
00596 /// Number of West slave ports
00597     static constexpr auto s_west_size =
00598       detail::enum_count(slave_port_layout::west_0,
00599                          slave_port_layout::west_last);
00600
00601 /// A range of the North slave ports
00602     static auto inline s_north_range =
00603       views::enum_type(slave_port_layout::north_0,
00604                        slave_port_layout::north_last);
00605
00606 /// Number of North slave ports
00607     static constexpr auto s_north_size =
00608       detail::enum_count(slave_port_layout::north_0,
00609                          slave_port_layout::north_last);
00610
00611 /// A range of the East slave ports
00612     static auto inline s_east_range =
00613       views::enum_type(slave_port_layout::east_0, slave_port_layout::east_last);
00614
00615 /// Number of East slave ports
00616     static constexpr auto s_east_size =
00617       detail::enum_count(slave_port_layout::east_0,
00618                          slave_port_layout::east_last);
00619
00620 /// Describe the AXI stream switch interconnection neighborhood
00621     static auto inline interconnect = boost::hana::make_tuple(
00622         // Connection topology of the NoC towards East of the switches
00623         std::tuple { 1, 0, m_east_range, s_west_range }
00624         // Connection topology of the NoC towards West of the switches
00625       , std::tuple { -1, 0, m_west_range, s_east_range }
00626         // Connection topology of the NoC towards North of the switches
00627       , std::tuple { 0, 1, m_north_range, s_south_range }
00628         // Connection topology of the NoC towards South of the switches
00629       , std::tuple { 0, -1, m_south_range, s_north_range }
00630       );
00631
00632
00633   };
00634
00635   // The organization of the AXI stream switch on a shim tile
00636   struct shim_axi_stream_switch {
00637 /// \todo factorize the 2 following values with core tiles?
00638 /// Default number of registers on default router paths
00639     static auto constexpr latency = 4;
00640
00641 /// Depth of additional FIFO on specific FIFO paths
00642     static auto constexpr fifo_depth = 16;
00643
00644 /** Layout of the AXI stream master ports in the switch
00645
00646 Revision v2.02, 6.5.3 Shim AXI-Stream Interconnect, Table 6-2, p.  291 */
00647     enum class master_port_layout :  std::int8_t {
00648       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation,
00649       fifo_0,
00650       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00651       south_0,
00652       south_1,
00653       south_2,
00654       south_3,
00655       south_4,
00656       south_5, south_last = south_5, ///< Used for shim BLI port validation
00657       west_0,
00658       west_1,
00659       west_2,
00660       west_3, west_last = west_3, ///< Used for west validation
00661       north_0,
00662       north_1,
00663       north_2,
00664       north_3,
00665       north_4,
00666       north_5, north_last = north_5, ///< Used for north validation
00667       east_0,
00668       east_1,
00669       east_2,
00670       east_3, east_last = east_3, ///< Used for east validation
```

```
00671       // To measure the enum
00672       size
00673     };
00674
00675   private:
00676
00677 /// \todo Factorize out
00678 /// Use a hidden friend to have introspection for master_port_layout
00679     friend bool constexpr is_axi_master(master_port_layout) {
00680       // Yes, master_port_layout is definitely an AXI master
00681       return true;
00682     }
00683
00684   public:
00685
00686 /// \todo factorize with a mix-in the following between master and slave
00687
00688 /// Number of AXI stream master ports in the switch
00689     static constexpr auto nb_master_port =
00690       detail::underlying_value(master_port_layout::size);
00691
00692 /// \todo Factorize out this range code with core tile
00693
00694 /// A range of the master ports connected to the tile control inputs
00695     static auto inline m_tile_ctrl_range =
00696       views::enum_type(master_port_layout::tile_ctrl,
00697                        master_port_layout::tile_ctrl_last);
00698
00699 /// Number of master ports connected to the tile control inputs
00700     static constexpr auto m_tile_ctrl_size =
00701       detail::enum_count(master_port_layout::tile_ctrl,
00702                          master_port_layout::tile_ctrl_last);
00703
00704 /// A range of the master ports connected to the FIFO inputs
00705     static auto inline m_fifo_range =
00706       views::enum_type(master_port_layout::fifo_0,
00707                        master_port_layout::fifo_last);
00708
00709 /// Number of master ports connected to the FIFO inputs
00710     static constexpr auto m_fifo_size =
00711       detail::enum_count(master_port_layout::fifo_0,
00712                          master_port_layout::fifo_last);
00713
00714 /// A range of the South master ports
00715     static auto inline m_south_range =
00716       views::enum_type(master_port_layout::south_0,
00717                        master_port_layout::south_last);
00718
00719 /// Number of South master ports
00720     static constexpr auto m_south_size =
00721       detail::enum_count(master_port_layout::south_0,
00722                          master_port_layout::south_last);
00723
00724 /// A range of the West master ports
00725     static auto inline m_west_range =
00726       views::enum_type(master_port_layout::west_0,
00727                        master_port_layout::west_last);
00728
00729 /// Number of West master ports
00730     static constexpr auto m_west_size =
00731       detail::enum_count(master_port_layout::west_0,
00732                          master_port_layout::west_last);
00733
00734 /// A range of the North master ports
00735     static auto inline m_north_range =
00736       views::enum_type(master_port_layout::north_0,
00737                        master_port_layout::north_last);
00738
00739 /// Number of North master ports
00740     static constexpr auto m_north_size =
00741       detail::enum_count(master_port_layout::north_0,
00742                          master_port_layout::north_last);
00743
00744 /// A range of the East master ports
00745     static auto inline m_east_range =
00746       views::enum_type(master_port_layout::east_0,
00747                        master_port_layout::east_last);
00748
00749 /// Number of East master ports
00750     static constexpr auto m_east_size =
00751       detail::enum_count(master_port_layout::east_0,
00752                          master_port_layout::east_last);
00753
00754 /** Layout of the AXI stream slave ports in the switch
00755
00756 Revision v2.02, 6.5.3 Shim AXI-Stream Interconnect, Table 6-3, p.  291 */
00757     enum class slave_port_layout :  std::int8_t {
```

```
00758       tile_ctrl, tile_ctrl_last = tile_ctrl, ///< Used for validation,
00759       fifo_0,
00760       fifo_1, fifo_last = fifo_1, ///< Used for FIFO validation
00761       south_0,
00762       south_1,
00763       south_2,
00764       south_3,
00765       south_4,
00766       south_5,
00767       south_6,
00768       south_7, south_last = south_7, ///< Used for shim BLI port validation
00769       west_0,
00770       west_1,
00771       west_2,
00772       west_3, west_last = west_3, ///< Used for west validation
00773       north_0,
00774       north_1,
00775       north_2,
00776       north_3, north_last = north_3, ///< Used for north validation
00777       east_0,
00778       east_1,
00779       east_2,
00780       east_3, east_last = east_3, ///< Used for east validation
00781       shim_trace,
00782       // To measure the enum
00783       size
00784     };
00785
00786   private:
00787
00788 /// Use a hidden friend to have introspection for slave_port_layout
00789     friend bool constexpr is_axi_master(slave_port_layout) {
00790       // No, slave_port_layout is definitely not an AXI master
00791       return false;
00792     }
00793
00794   public:
00795
00796 /// Number of master AXI stream slave ports in the switch
00797     static constexpr auto nb_slave_port =
00798       static_cast<int>(slave_port_layout::size);
00799
00800 /// A range of the slave ports connected to the tile control outputs
00801     static auto inline s_tile_ctrl_range =
00802       views::enum_type(slave_port_layout::tile_ctrl,
00803                        slave_port_layout::tile_ctrl_last);
00804
00805 /// Number of slave ports connected to the tile control outputs
00806     static constexpr auto s_tile_ctrl_size =
00807       detail::enum_count(slave_port_layout::tile_ctrl,
00808                          slave_port_layout::tile_ctrl_last);
00809
00810 /// A range of the slave ports connected to the FIFO outputs
00811     static auto inline s_fifo_range =
00812       views::enum_type(slave_port_layout::fifo_0, slave_port_layout::fifo_last);
00813
00814 /// Number of slave ports connected to the FIFO outputs
00815     static constexpr auto s_fifo_size =
00816       detail::enum_count(slave_port_layout::fifo_0,
00817                          slave_port_layout::fifo_last);
00818
00819 /// A range of the South slave ports
00820     static auto inline s_south_range =
00821       views::enum_type(slave_port_layout::south_0,
00822                        slave_port_layout::south_last);
00823
00824 /// Number of South slave ports
00825     static constexpr auto s_south_size =
00826       detail::enum_count(slave_port_layout::south_0,
00827                          slave_port_layout::south_last);
00828
00829 /// A range of the West slave ports
00830     static auto inline s_west_range =
00831       views::enum_type(slave_port_layout::west_0, slave_port_layout::west_last);
00832
00833 /// Number of West slave ports
00834     static constexpr auto s_west_size =
00835       detail::enum_count(slave_port_layout::west_0,
00836                          slave_port_layout::west_last);
00837
00838 /// A range of the North slave ports
00839     static auto inline s_north_range =
00840       views::enum_type(slave_port_layout::north_0,
00841                        slave_port_layout::north_last);
00842
00843 /// Number of North slave ports
00844     static constexpr auto s_north_size =
```

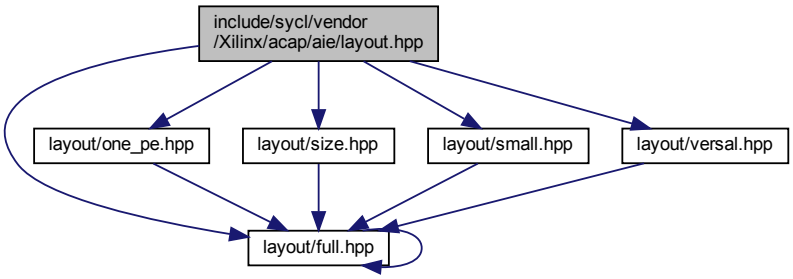
```
00845       detail::enum_count(slave_port_layout::north_0,
00846                          slave_port_layout::north_last);
00847
00848 /// A range of the East slave ports
00849     static auto inline s_east_range =
00850       views::enum_type(slave_port_layout::east_0, slave_port_layout::east_last);
00851
00852 /// Number of East slave ports
00853     static constexpr auto s_east_size =
00854       detail::enum_count(slave_port_layout::east_0,
00855                          slave_port_layout::east_last);
00856
00857   };
00858
00859 };
00860
00861 /// @} End the aie Doxygen group
00862
00863 }
00864
00865 /*
00866 # Some Emacs stuff:
00867 ### Local Variables:
00868 ### ispell-local-dictionary:  "american"
00869 ### eval:  (flyspell-prog-mode)
00870 ### End:
00871 */
00872
00873 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_GEOGRAPHY_HPP
```

13.297 include/sycl/vendor/Xilinx/acap/aie/layout.hpp File Reference

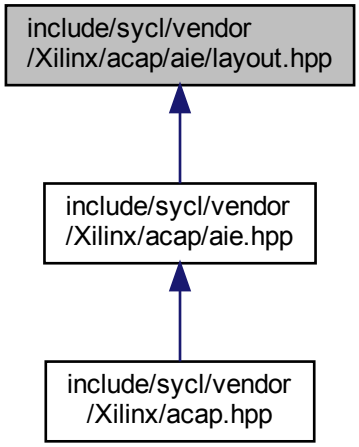
Some examples of AI Engine array layouts.

```
#include "layout/full.hpp"
#include "layout/one_pe.hpp"
#include "layout/size.hpp"
#include "layout/small.hpp"
#include "layout/versal.hpp"
```

Include dependency graph for layout.hpp:



This graph shows which files directly or indirectly include this file:



13.297.1 Detailed Description

Some examples of AI Engine array layouts.

Mainly used for unit test purposes.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file layout.hpp.

13.298 layout.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_HPP
00003
00004 /** \file
00005
00006 Some examples of AI Engine array layouts
00007
00008 Mainly used for unit test purposes.
00009
00010 Ronan dot Keryell at Xilinx dot com
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include "layout/full.hpp"
00017 #include "layout/one_pe.hpp"
00018 #include "layout/size.hpp"
00019 #include "layout/small.hpp"
00020 #include "layout/versal.hpp"
00021
00022 /*
```

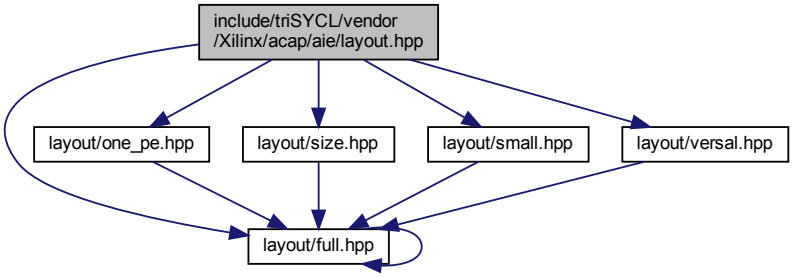
```
00023 # Some Emacs stuff:
00024 ### Local Variables:
00025 ### ispell-local-dictionary:  "american"
00026 ### eval:  (flyspell-prog-mode)
00027 ### End:
00028 */
00029
00030 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_HPP
```

13.299 include/triSYCL/vendor/Xilinx/acap/aie/layout.hpp File Reference

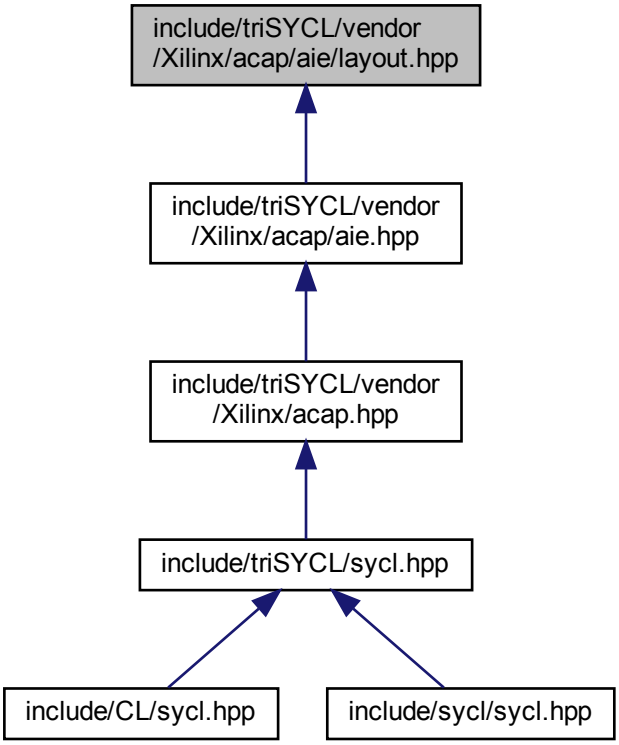
Some examples of AI Engine array layouts.

```
#include "layout/full.hpp"
#include "layout/one_pe.hpp"
#include "layout/size.hpp"
#include "layout/small.hpp"
#include "layout/versal.hpp"
```

Include dependency graph for layout.hpp:



This graph shows which files directly or indirectly include this file:



13.299.1 Detailed Description

Some examples of AI Engine array layouts.

Mainly used for unit test purposes.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file layout.hpp.

13.300 layout.hpp

Go to the documentation of this file.

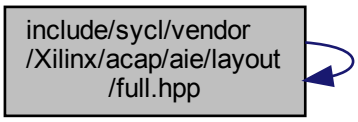
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_HPP
00003
00004 /** \file
00005
00006 Some examples of AI Engine array layouts
00007
00008 Mainly used for unit test purposes.
```

```
00009
00010 Ronan dot Keryell at Xilinx dot com
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include "layout/full.hpp"
00017 #include "layout/one_pe.hpp"
00018 #include "layout/size.hpp"
00019 #include "layout/small.hpp"
00020 #include "layout/versal.hpp"
00021
00022 /*
00023 # Some Emacs stuff:
00024 ### Local Variables:
00025 ### ispell-local-dictionary:  "american"
00026 ### eval:  (flyspell-prog-mode)
00027 ### End:
00028 */
00029
00030 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_HPP
```

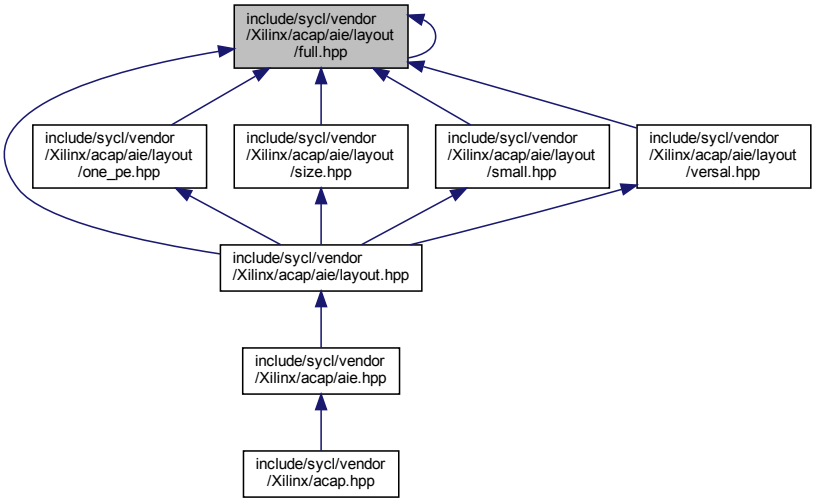
13.301 include/sycl/vendor/Xilinx/acap/aie/layout/full.hpp File Reference

Layout for a AI Engine array with 5∗6 PE.

`#include "full.hpp"`
Include dependency graph for full.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::layout::full

  Describe the layout of a "big" AI Engine array. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::layout

13.301.1 Detailed Description

Layout for a AI Engine array with 5∗6 PE.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file full.hpp.

13.302 full.hpp

Go to the documentation of this file.

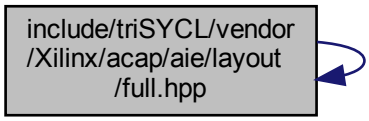
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_FULL_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_FULL_HPP
00003
00004 /** \file
00005
00006 Layout for a AI Engine array with 5*6 PE
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include "full.hpp"
00015
00016 namespace trisycl::vendor::xilinx::acap::aie::layout {
00017
00018 /// \ingroup aie
00019 /// @{
00020
00021 /** Describe the layout of a "big" AI Engine array
00022
00023 Use some constrains from Figure 2-8 "ME Array Address Map
00024 Example", p.  44
00025 */
00026 struct full {
00027   static auto constexpr x_min = 0;
00028   static auto constexpr x_max = 5;
00029   static auto constexpr y_min = 0;
00030   static auto constexpr y_max = 4;
00031
00032   static bool constexpr is_noc_tile(int x, int y) {
00033     return y == 0 && 2 <= x && x <= 3;
00034   }
00035
00036   static bool constexpr is_pl_tile(int x, int y) {
```

```
00037     return y == 0 && ((0 <= x && x <= 1) || (4 <= x && x <= 5));
00038   }
00039 };
00040
00041 /// @} End the aie Doxygen group
00042
00043 }
00044
00045 /*
00046 # Some Emacs stuff:
00047 ### Local Variables:
00048 ### ispell-local-dictionary:  "american"
00049 ### eval:  (flyspell-prog-mode)
00050 ### End:
00051 */
00052
00053 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_FULL_HPP
```

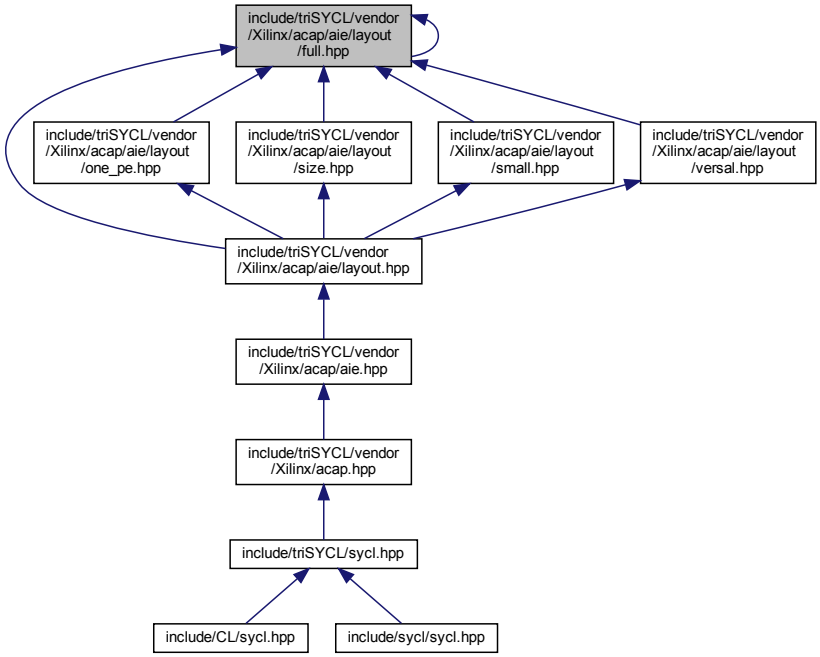
13.303 include/triSYCL/vendor/Xilinx/acap/aie/layout/full.hpp File Reference

Layout for a AI Engine array with 5∗6 PE.

`#include "full.hpp"`
Include dependency graph for full.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::layout::full

  Describe the layout of a "big" AI Engine array. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::layout

13.303.1 Detailed Description

Layout for a AI Engine array with 5∗6 PE.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file full.hpp.

13.304 full.hpp

Go to the documentation of this file.

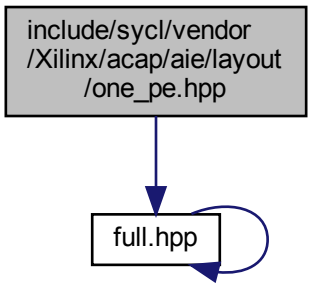
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_FULL_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_FULL_HPP
00003
00004 /** \file
00005
00006 Layout for a AI Engine array with 5*6 PE
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include "full.hpp"
00015
00016 namespace trisycl::vendor::xilinx::acap::aie::layout {
00017
00018 /// \ingroup aie
00019 /// @{
00020
00021 /** Describe the layout of a "big" AI Engine array
00022
00023 Use some constrains from Figure 2-8 "ME Array Address Map
00024 Example", p.  44
00025 */
00026 struct full {
00027   static auto constexpr x_min = 0;
00028   static auto constexpr x_max = 5;
00029   static auto constexpr y_min = 0;
00030   static auto constexpr y_max = 4;
00031
00032   static bool constexpr is_noc_tile(int x, int y) {
00033     return y == 0 && 2 <= x && x <= 3;
00034   }
00035
00036   static bool constexpr is_pl_tile(int x, int y) {
00037     return y == 0 && ((0 <= x && x <= 1) || (4 <= x && x <= 5));
00038   }
00039 };
00040
00041 /// @} End the aie Doxygen group
00042
00043 }
00044
00045 /*
00046 # Some Emacs stuff:
00047 ### Local Variables:
00048 ### ispell-local-dictionary:  "american"
00049 ### eval:  (flyspell-prog-mode)
00050 ### End:
00051 */
00052
00053 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_FULL_HPP
```

13.305 include/sycl/vendor/Xilinx/acap/aie/layout/one_pe.hpp File Reference

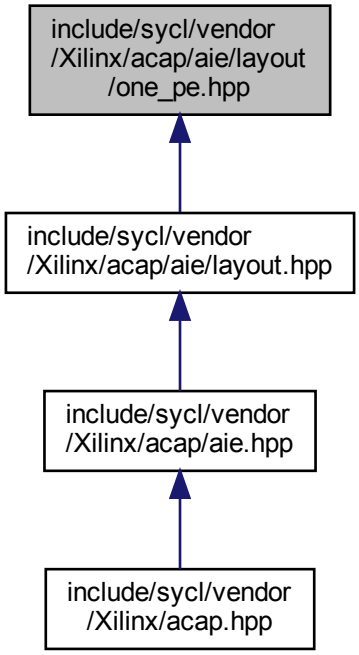
Layout for a AI Engine array with 1 PE only.

```
#include "full.hpp"
```

Include dependency graph for one_pe.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::layout::one_pe

  Describe the layout of a tiny AI Engine with only 1 tile. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::layout

13.305.1 Detailed Description

Layout for a AI Engine array with 1 PE only.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file one_pe.hpp.

13.306 one_pe.hpp

Go to the documentation of this file.

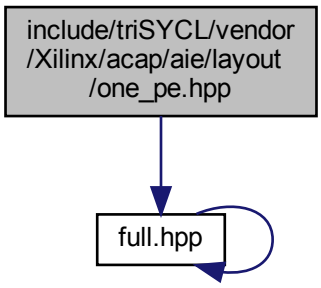
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_ONE_PE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_ONE_PE_HPP
00003
00004 /** \file
00005
00006 Layout for a AI Engine array with 1 PE only
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include "full.hpp"
00015
00016 namespace trisycl::vendor::xilinx::acap::aie::layout {
00017
00018 /// \ingroup aie
00019 /// @{
00020
00021 /** Describe the layout of a tiny AI Engine with only 1 tile
00022 */
00023 struct one_pe :  full {
00024   static auto constexpr x_max = 0;
00025   static auto constexpr y_max = 0;
00026 };
00027
00028 /// @} End the aie Doxygen group
00029
00030 }
00031
00032 /*
00033 # Some Emacs stuff:
00034 ### Local Variables:
00035 ### ispell-local-dictionary:  "american"
00036 ### eval:  (flyspell-prog-mode)
00037 ### End:
00038 */
00039
00040 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_ONE_PE_HPP
```

13.307 include/triSYCL/vendor/Xilinx/acap/aie/layout/one_pe.hpp File Reference

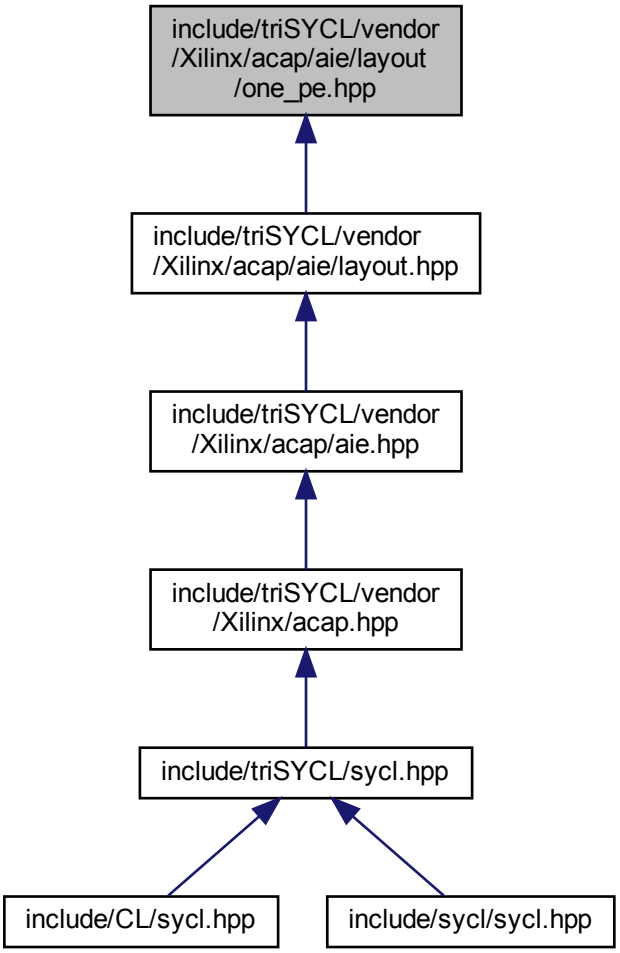
Layout for a AI Engine array with 1 PE only.

```
#include "full.hpp"
```

Include dependency graph for one_pe.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::layout::one_pe

  Describe the layout of a tiny AI Engine with only 1 tile. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::layout

13.307.1 Detailed Description

Layout for a AI Engine array with 1 PE only.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file one_pe.hpp.

13.308 one_pe.hpp

Go to the documentation of this file.

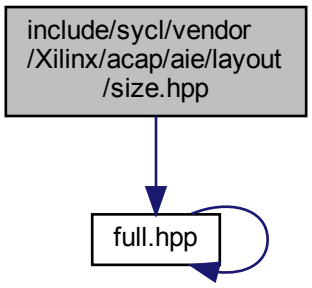
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_ONE_PE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_ONE_PE_HPP
00003
00004 /** \file
00005
00006 Layout for a AI Engine array with 1 PE only
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include "full.hpp"
00015
00016 namespace trisycl::vendor::xilinx::acap::aie::layout {
00017
00018 /// \ingroup aie
00019 /// @{
00020
00021 /** Describe the layout of a tiny AI Engine with only 1 tile
00022 */
00023 struct one_pe :  full {
00024   static auto constexpr x_max = 0;
00025   static auto constexpr y_max = 0;
00026 };
00027
00028 /// @} End the aie Doxygen group
00029
00030 }
00031
00032 /*
00033 # Some Emacs stuff:
00034 ### Local Variables:
00035 ### ispell-local-dictionary:  "american"
00036 ### eval:  (flyspell-prog-mode)
00037 ### End:
00038 */
00039
00040 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_ONE_PE_HPP
```

13.309 include/sycl/vendor/Xilinx/acap/aie/layout/size.hpp File Reference

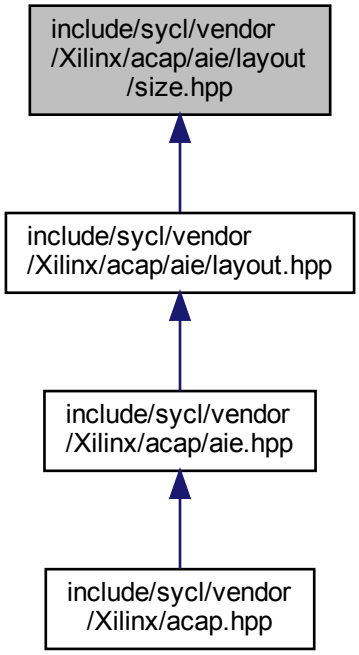
Flexible layout for a AI Engine array with any 2D size.

```
#include "full.hpp"
```

Include dependency graph for size.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::layout::size< X_size, Y_size >

  Fully specify the size of an AI Engine layout. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::layout

13.309.1 Detailed Description

Flexible layout for a AI Engine array with any 2D size.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file size.hpp.

13.310 size.hpp

Go to the documentation of this file.

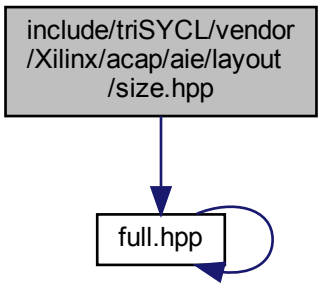
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_SIZE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_SIZE_HPP
00003
00004 /** \file
00005
00006 Flexible layout for a AI Engine array with any 2D size
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include "full.hpp"
00015
00016 // Just to include the namespace once in Doxygen documentation
00017 /// \ingroup aie
00018 namespace trisycl::vendor::xilinx::acap::aie::layout {
00019
00020 /// \ingroup aie
00021 /// @{
00022
00023 /** Fully specify the size of an AI Engine layout
00024
00025 \param X_size is the number of tile horizontally
00026
00027 \param Y_size is the number of tile vertically
00028 */
00029 template <int X_size, int Y_size>
00030 struct size :  full {
00031   static_assert(X_size >= 1 && Y_size >= 1, "Sorry, "
00032                 "we do not yet support antimatter or black matter cores :-)");
00033 /// Maximum horizontal coordinate
00034   static auto constexpr x_max = X_size - 1;
00035
00036 /// Maximum vertical coordinate
00037   static auto constexpr y_max = Y_size - 1;
00038 };
00039
00040 /// @} End the aie Doxygen group
00041
00042 }
00043
00044 /*
00045 # Some Emacs stuff:
00046 ### Local Variables:
00047 ### ispell-local-dictionary:  "american"
00048 ### eval:  (flyspell-prog-mode)
00049 ### End:
00050 */
00051
00052 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_SIZE_HPP
```

13.311 include/triSYCL/vendor/Xilinx/acap/aie/layout/size.hpp File Reference

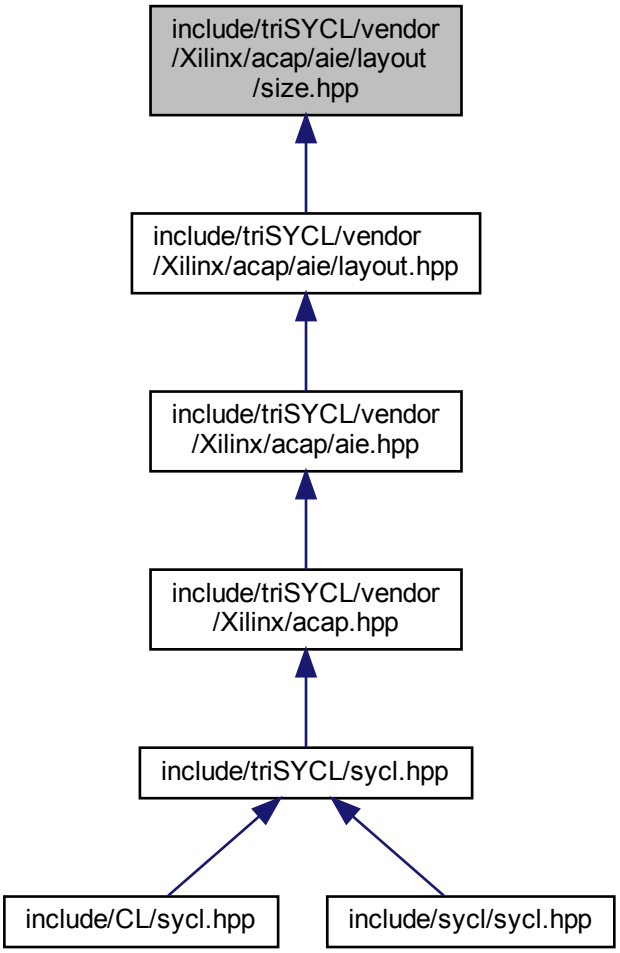
Flexible layout for a AI Engine array with any 2D size.

```
#include "full.hpp"
```

Include dependency graph for size.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::layout::size< X_size, Y_size >

  Fully specify the size of an AI Engine layout. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::layout

13.311.1 Detailed Description

Flexible layout for a AI Engine array with any 2D size.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file size.hpp.

13.312 size.hpp

Go to the documentation of this file.

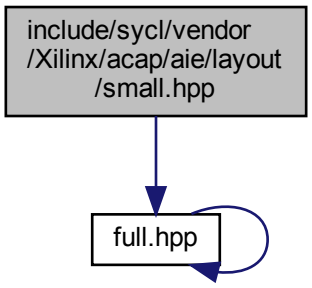
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_SIZE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_SIZE_HPP
00003
00004 /** \file
00005
00006 Flexible layout for a AI Engine array with any 2D size
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include "full.hpp"
00015
00016 // Just to include the namespace once in Doxygen documentation
00017 /// \ingroup aie
00018 namespace trisycl::vendor::xilinx::acap::aie::layout {
00019
00020 /// \ingroup aie
00021 /// @{
00022
00023 /** Fully specify the size of an AI Engine layout
00024
00025 \param X_size is the number of tile horizontally
00026
00027 \param Y_size is the number of tile vertically
00028 */
00029 template <int X_size, int Y_size>
00030 struct size :  full {
00031   static_assert(X_size >= 1 && Y_size >= 1, "Sorry, "
00032                 "we do not yet support antimatter or black matter cores :-)");
00033 /// Maximum horizontal coordinate
00034   static auto constexpr x_max = X_size - 1;
00035
00036 /// Maximum vertical coordinate
00037   static auto constexpr y_max = Y_size - 1;
00038 };
00039
00040 /// @} End the aie Doxygen group
00041
00042 }
00043
00044 /*
00045 # Some Emacs stuff:
00046 ### Local Variables:
00047 ### ispell-local-dictionary:  "american"
00048 ### eval:  (flyspell-prog-mode)
00049 ### End:
00050 */
00051
00052 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_SIZE_HPP
```

13.313 include/sycl/vendor/Xilinx/acap/aie/layout/small.hpp File Reference

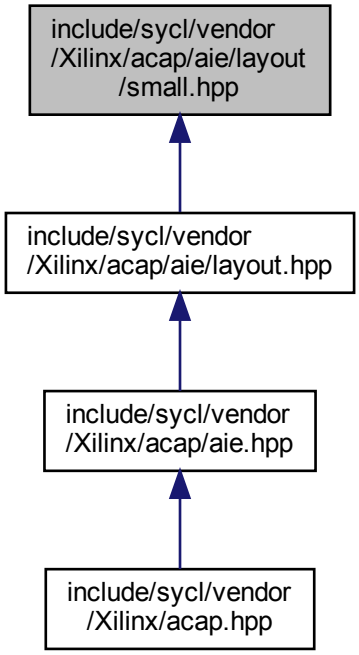
Layout for a AI Engine array with a few PE only.

```
#include "full.hpp"
```

Include dependency graph for small.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::layout::small

  Describe the layout of a small AI Engine array with only a few PE. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::layout

13.313.1 Detailed Description

Layout for a AI Engine array with a few PE only.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file small.hpp.

13.314 small.hpp

Go to the documentation of this file.

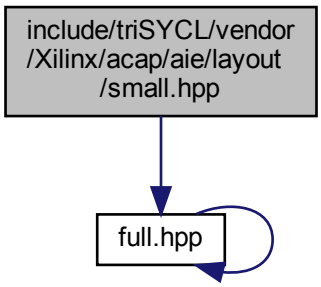
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_SMALL_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_SMALL_HPP
00003
00004 /** \file
00005
00006 Layout for a AI Engine array with a few PE only
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include "full.hpp"
00015
00016 namespace trisycl::vendor::xilinx::acap::aie::layout {
00017
00018 /// \ingroup aie
00019 /// @{
00020
00021 /** Describe the layout of a small AI Engine array with only a few PE
00022 */
00023 struct small :  full {
00024   static auto constexpr x_max = 1;
00025   static auto constexpr y_max = 2;
00026 };
00027
00028 /// @} End the aie Doxygen group
00029
00030 }
00031
00032 /*
00033 # Some Emacs stuff:
00034 ### Local Variables:
00035 ### ispell-local-dictionary:  "american"
00036 ### eval:  (flyspell-prog-mode)
00037 ### End:
00038 */
00039
00040 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_SMALL_HPP
```

13.315 include/triSYCL/vendor/Xilinx/acap/aie/layout/small.hpp File Reference

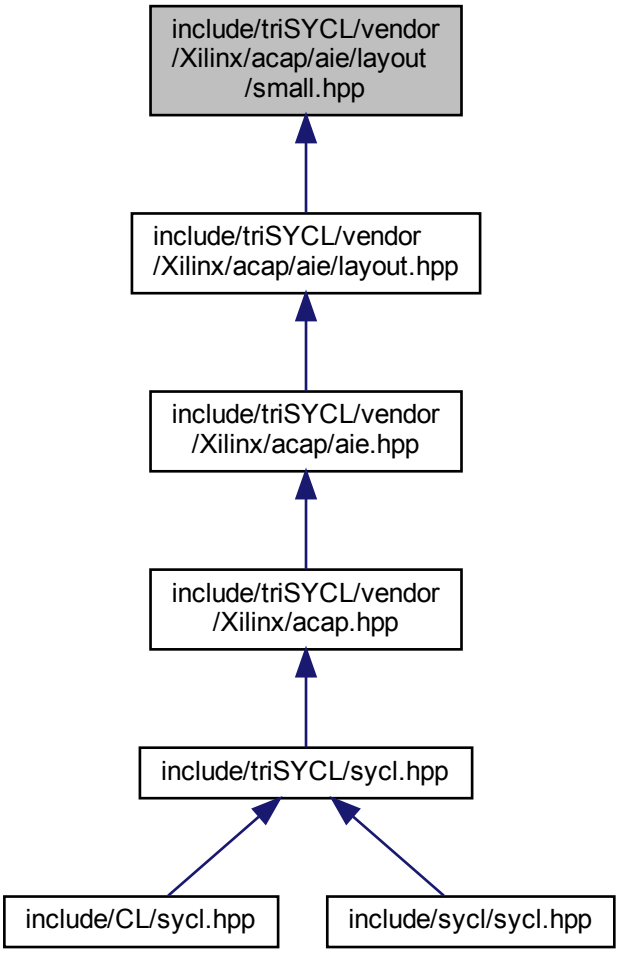
Layout for a AI Engine array with a few PE only.

```
#include "full.hpp"
```

Include dependency graph for small.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::layout::small

  Describe the layout of a small AI Engine array with only a few PE. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::layout

13.315.1 Detailed Description

Layout for a AI Engine array with a few PE only.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file small.hpp.

13.316 small.hpp

Go to the documentation of this file.

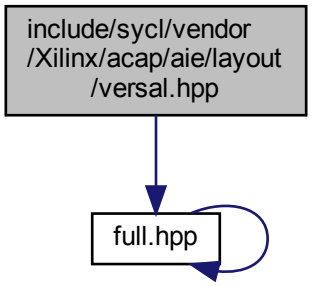
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_SMALL_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_SMALL_HPP
00003
00004 /** \file
00005
00006 Layout for a AI Engine array with a few PE only
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include "full.hpp"
00015
00016 namespace trisycl::vendor::xilinx::acap::aie::layout {
00017
00018 /// \ingroup aie
00019 /// @{
00020
00021 /** Describe the layout of a small AI Engine array with only a few PE
00022 */
00023 struct small :  full {
00024   static auto constexpr x_max = 1;
00025   static auto constexpr y_max = 2;
00026 };
00027
00028 /// @} End the aie Doxygen group
00029
00030 }
00031
00032 /*
00033 # Some Emacs stuff:
00034 ### Local Variables:
00035 ### ispell-local-dictionary:  "american"
00036 ### eval:  (flyspell-prog-mode)
00037 ### End:
00038 */
00039
00040 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_SMALL_HPP
```

13.317 include/sycl/vendor/Xilinx/acap/aie/layout/versal.hpp File Reference

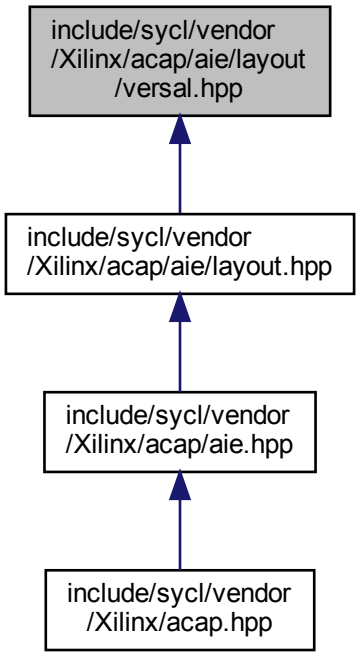
Layout for some Xilinx Versal products.

```
#include "full.hpp"
```

Include dependency graph for versal.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::layout::ve1152
- struct trisycl::vendor::xilinx::acap::aie::layout::vc1902

  Describe the layout of a Versal VC1902 device.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::layout

Typedefs

- using trisycl::vendor::xilinx::acap::aie::layout::xc10s10 = ve1152

  Alias to ve1152.
- using trisycl::vendor::xilinx::acap::aie::layout::xcve1152 = ve1152

  Alias to ve1152.
- using trisycl::vendor::xilinx::acap::aie::layout::xc10s80 = vc1902

  Alias to vc1902.
- using trisycl::vendor::xilinx::acap::aie::layout::xcvc1902 = vc1902

  Alias to vc1902.

13.317.1 Detailed Description

Layout for some Xilinx Versal products.

/proj/xbuilds/HEAD_INT_daily_latest/installs/lin64/cardano/HEAD/data/devices/S80.MEPL.json

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file versal.hpp.

13.318 versal.hpp

Go to the documentation of this file.

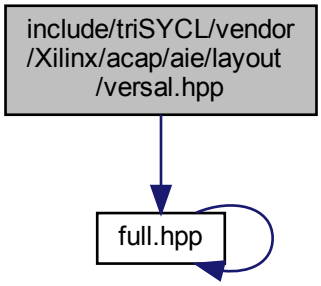
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_VERSAL_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_VERSAL_HPP
00003
00004 /** \file
00005
00006 Layout for some Xilinx Versal products
00007
00008 /proj/xbuilds/HEAD_INT_daily_latest/installs/lin64/cardano/HEAD/data/devices/S80.MEPL.json
00009
00010 Ronan dot Keryell at Xilinx dot com
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include "full.hpp"
00017
00018 namespace trisycl::vendor::xilinx::acap::aie::layout {
00019
```

```
00020 /// \ingroup aie
00021 /// @{
00022
00023 /// Describe the layout of a Versal VE1152 device
00024 struct ve1152 :  size<17, 3> {;
00025 };
00026
00027 /// Alias to ve1152
00028 using xc10s10 = ve1152;
00029
00030 /// Alias to ve1152
00031 using xcve1152 = ve1152;
00032
00033
00034 /// Describe the layout of a Versal VC1902 device
00035 struct vc1902 :  size<50, 8> {
00036 };
00037
00038 /// Alias to vc1902
00039 using xc10s80 = vc1902;
00040
00041 /// Alias to vc1902
00042 using xcvc1902 = vc1902;
00043
00044
00045 /// @} End the aie Doxygen group
00046
00047 }
00048
00049 /*
00050 # Some Emacs stuff:
00051 ### Local Variables:
00052 ### ispell-local-dictionary:  "american"
00053 ### eval:  (flyspell-prog-mode)
00054 ### End:
00055 */
00056
00057 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_VERSAL_HPP
```

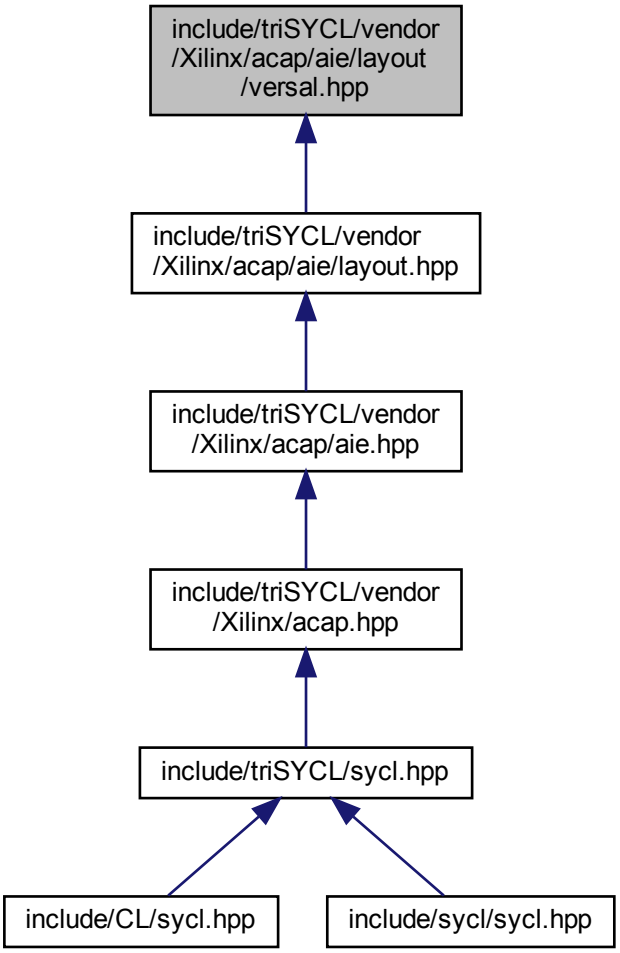
13.319 include/triSYCL/vendor/Xilinx/acap/aie/layout/versal.hpp File Reference

Layout for some Xilinx Versal products.

`#include "full.hpp"`
Include dependency graph for versal.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::layout::ve1152
- struct trisycl::vendor::xilinx::acap::aie::layout::vc1902

  Describe the layout of a Versal VC1902 device.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::layout

13.319.1 Detailed Description

Layout for some Xilinx Versal products.

/proj/xbuilds/HEAD_INT_daily_latest/installs/lin64/cardano/HEAD/data/devices/S80.MEPL.json

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file versal.hpp.

13.320 versal.hpp

Go to the documentation of this file.

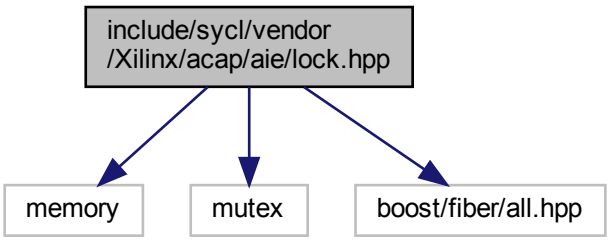
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_VERSAL_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_VERSAL_HPP
00003
00004 /** \file
00005
00006 Layout for some Xilinx Versal products
00007
00008 /proj/xbuilds/HEAD_INT_daily_latest/installs/lin64/cardano/HEAD/data/devices/S80.MEPL.json
00009
00010 Ronan dot Keryell at Xilinx dot com
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include "full.hpp"
00017
00018 namespace trisycl::vendor::xilinx::acap::aie::layout {
00019
00020 /// \ingroup aie
00021 /// @{
00022
00023 /// Describe the layout of a Versal VE1152 device
00024 struct ve1152 :  size<17, 3> {;
00025 };
00026
00027 /// Alias to ve1152
00028 using xc10s10 = ve1152;
00029
00030 /// Alias to ve1152
00031 using xcve1152 = ve1152;
00032
00033
00034 /// Describe the layout of a Versal VC1902 device
00035 struct vc1902 :  size<50, 8> {
00036 };
00037
00038 /// Alias to vc1902
00039 using xc10s80 = vc1902;
00040
00041 /// Alias to vc1902
00042 using xcvc1902 = vc1902;
00043
00044
00045 /// @} End the aie Doxygen group
00046
00047 }
00048
00049 /*
00050 # Some Emacs stuff:
00051 ### Local Variables:
00052 ### ispell-local-dictionary:  "american"
00053 ### eval:  (flyspell-prog-mode)
00054 ### End:
00055 */
00056
00057 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LAYOUT_VERSAL_HPP
```

13.321 include/sycl/vendor/Xilinx/acap/aie/lock.hpp File Reference

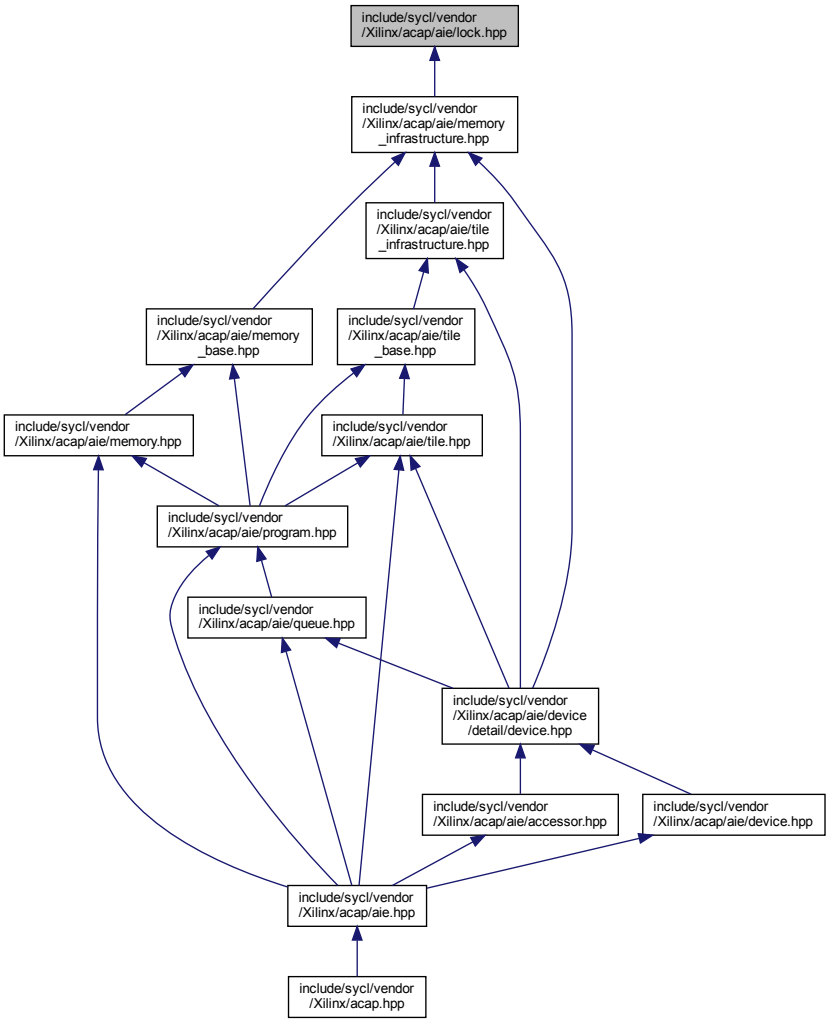
The lock mechanism used by some AI Engine tiles.

```
#include <memory>
#include <mutex>
#include <boost/fiber/all.hpp>
```

Include dependency graph for lock.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::lock_unit

  The lock infrastructure used by AI Engine memory modules and shim tiles. More...
- struct trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device

  The individual locking system.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.321.1 Detailed Description

The lock mechanism used by some AI Engine tiles.

Note that this AI Engine concept is not a pure lock, but more like a lock associated with a conditional variable, to follow the C++ jargon.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file lock.hpp.

13.322 lock.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LOCK_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LOCK_HPP
00003
00004 /** \file
00005
00006 The lock mechanism used by some AI Engine tiles
00007
00008 Note that this AI Engine concept is not a pure lock, but more
00009 like a lock associated with a conditional variable, to follow the
00010 C++ jargon.
00011
00012 Ronan dot Keryell at Xilinx dot com
00013
00014 This file is distributed under the University of Illinois Open Source
00015 License.  See LICENSE.TXT for details.
00016 */
00017
00018 #include <memory>
00019 #include <mutex>
00020
00021 #include <boost/fiber/all.hpp>
00022
00023 namespace trisycl::vendor::xilinx::acap::aie {
00024
00025 /// \ingroup aie
00026 /// @{
00027
00028 /** The lock infrastructure used by AI Engine memory modules and shim tiles
00029
00030 Based on Math Engine (ME) Architecture Specification, Revision v1.5
00031 June 2018
00032
00033 4.4.6 Lock Interface, p.  115
00034
00035 4.7 Lock Unit, p.  129
00036 */
00037 struct lock_unit {
00038   // There are 16 hardware locks per memory tile
00039   auto static constexpr lock_number = 16;
00040
00041   using lock_id_t = decltype(lock_number);
00042
00043 /// The individual locking system
00044   struct locking_device {
00045 /// The type of the value stored in the locking device
00046     using value_t = bool;
00047
00048     /* The problem here is that \c mutex and \c condition_variable
00049 are not moveable while the instantiation of a memory module uses
00050 move assignment with Boost.Hana...
00051
00052 So allocate them dynamically and keep them in a \c std::unique_ptr
00053 so globally the type is moveable */
00054
00055 /// The mutex to provide the basic protection mechanism
00056     std::unique_ptr<boost::fibers::mutex> m { new boost::fibers::mutex { } };
00057
00058 /// The condition variable to wait/notify for some value
00059     std::unique_ptr<boost::fibers::condition_variable> cv {
```

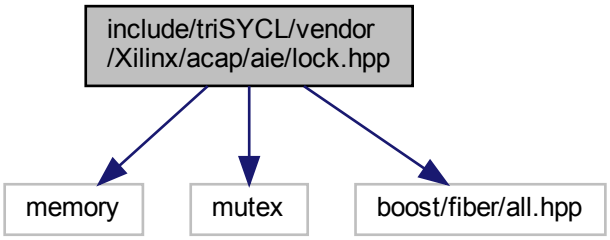
```
00060       new boost::fibers::condition_variable { } };
00061
00062 /// The value to be waited for, initialized to false on reset
00063     value_t value = false;
00064
00065 /// Lock the mutex
00066     void acquire() {
00067       m->lock();
00068     }
00069
00070
00071 /// Unlock the mutex
00072     void release() {
00073       m->unlock();
00074     }
00075
00076
00077 /// Wait until the internal value has the expectation
00078     void acquire_with_value(value_t expectation) {
00079       std::unique_lock lk { *m };
00080       cv->wait(lk, [&] { return expectation == value; });
00081     }
00082
00083
00084 /// Release and update with a new internal value
00085     void release_with_value(value_t new_value) {
00086       {
00087         std::unique_lock lk { *m };
00088         value = new_value;
00089       }
00090       // By construction there should be only one client waiting for it
00091       cv->notify_one();
00092     }
00093   };
00094
00095 /// The locking units of the locking device
00096   locking_device locks[lock_number];
00097
00098 /// Get the requested lock
00099   auto &lock(int i) {
00100     assert(0 <= i && i < lock_number);
00101     return locks[i];
00102   }
00103
00104 };
00105
00106 /// @} End the aie Doxygen group
00107
00108 }
00109
00110 /*
00111 # Some Emacs stuff:
00112 ### Local Variables:
00113 ### ispell-local-dictionary:  "american"
00114 ### eval:  (flyspell-prog-mode)
00115 ### End:
00116 */
00117
00118 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LOCK_HPP
```

13.323 include/triSYCL/vendor/Xilinx/acap/aie/lock.hpp File Reference

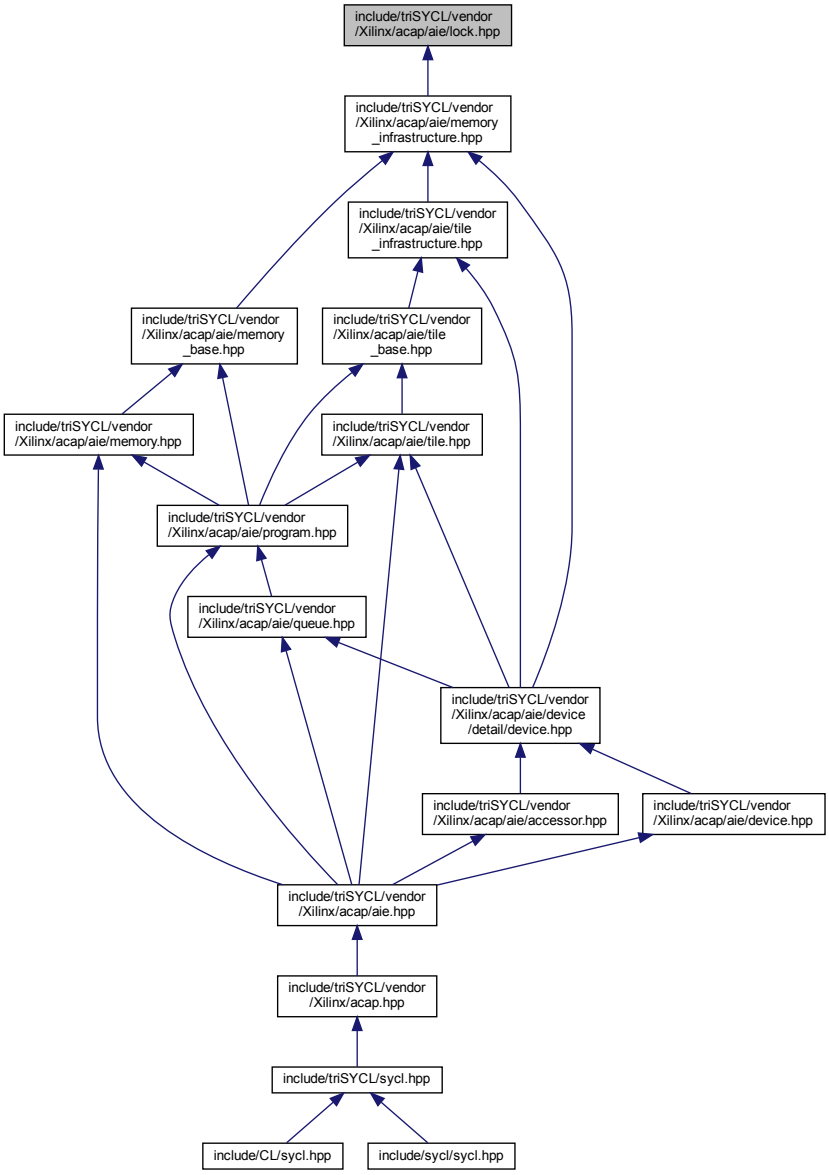
The lock mechanism used by some AI Engine tiles.

```
#include <memory>
#include <mutex>
#include <boost/fiber/all.hpp>
```

Include dependency graph for lock.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::lock_unit

  The lock infrastructure used by AI Engine memory modules and shim tiles. More...
- struct trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device

  The individual locking system.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.323.1 Detailed Description

The lock mechanism used by some AI Engine tiles.

Note that this AI Engine concept is not a pure lock, but more like a lock associated with a conditional variable, to follow the C++ jargon.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file lock.hpp.

13.324 lock.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LOCK_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LOCK_HPP
00003
00004 /** \file
00005
00006 The lock mechanism used by some AI Engine tiles
00007
00008 Note that this AI Engine concept is not a pure lock, but more
00009 like a lock associated with a conditional variable, to follow the
00010 C++ jargon.
00011
00012 Ronan dot Keryell at Xilinx dot com
00013
00014 This file is distributed under the University of Illinois Open Source
00015 License.  See LICENSE.TXT for details.
00016 */
00017
00018 #include <memory>
00019 #include <mutex>
00020
00021 #include <boost/fiber/all.hpp>
00022
00023 namespace trisycl::vendor::xilinx::acap::aie {
00024
00025 /// \ingroup aie
00026 /// @{
00027
00028 /** The lock infrastructure used by AI Engine memory modules and shim tiles
00029
00030 Based on Math Engine (ME) Architecture Specification, Revision v1.5
00031 June 2018
00032
00033 4.4.6 Lock Interface, p.  115
00034
00035 4.7 Lock Unit, p.  129
00036 */
00037 struct lock_unit {
00038   // There are 16 hardware locks per memory tile
00039   auto static constexpr lock_number = 16;
00040
00041   using lock_id_t = decltype(lock_number);
00042
00043 /// The individual locking system
00044   struct locking_device {
00045 /// The type of the value stored in the locking device
00046     using value_t = bool;
00047
00048     /* The problem here is that \c mutex and \c condition_variable
00049 are not moveable while the instantiation of a memory module uses
```

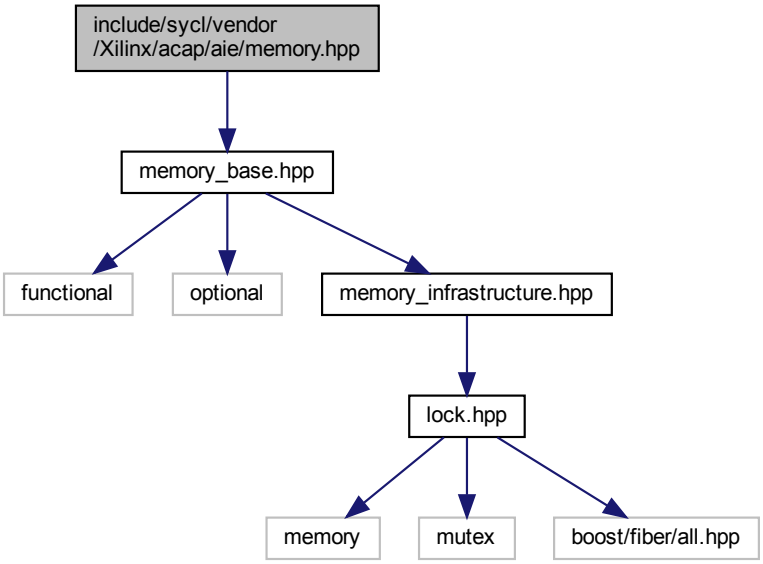
```
00050 move assignment with Boost.Hana...
00051
00052 So allocate them dynamically and keep them in a \c std::unique_ptr
00053 so globally the type is moveable */
00054
00055 /// The mutex to provide the basic protection mechanism
00056     std::unique_ptr<boost::fibers::mutex> m { new boost::fibers::mutex { } };
00057
00058 /// The condition variable to wait/notify for some value
00059     std::unique_ptr<boost::fibers::condition_variable> cv {
00060       new boost::fibers::condition_variable { } };
00061
00062 /// The value to be waited for, initialized to false on reset
00063     value_t value = false;
00064
00065 /// Lock the mutex
00066     void acquire() {
00067       m->lock();
00068     }
00069
00070
00071 /// Unlock the mutex
00072     void release() {
00073       m->unlock();
00074     }
00075
00076
00077 /// Wait until the internal value has the expectation
00078     void acquire_with_value(value_t expectation) {
00079       std::unique_lock lk { *m };
00080       cv->wait(lk, [&] { return expectation == value; });
00081     }
00082
00083
00084 /// Release and update with a new internal value
00085     void release_with_value(value_t new_value) {
00086       {
00087         std::unique_lock lk { *m };
00088         value = new_value;
00089       }
00090       // By construction there should be only one client waiting for it
00091       cv->notify_one();
00092     }
00093   };
00094
00095 /// The locking units of the locking device
00096   locking_device locks[lock_number];
00097
00098 /// Get the requested lock
00099   auto &lock(int i) {
00100     assert(0 <= i && i < lock_number);
00101     return locks[i];
00102   }
00103
00104 };
00105
00106 /// @} End the aie Doxygen group
00107
00108 }
00109
00110 /*
00111 # Some Emacs stuff:
00112 ### Local Variables:
00113 ### ispell-local-dictionary:  "american"
00114 ### eval:  (flyspell-prog-mode)
00115 ### End:
00116 */
00117
00118 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_LOCK_HPP
```

13.325 include/sycl/vendor/Xilinx/acap/aie/memory.hpp File Reference

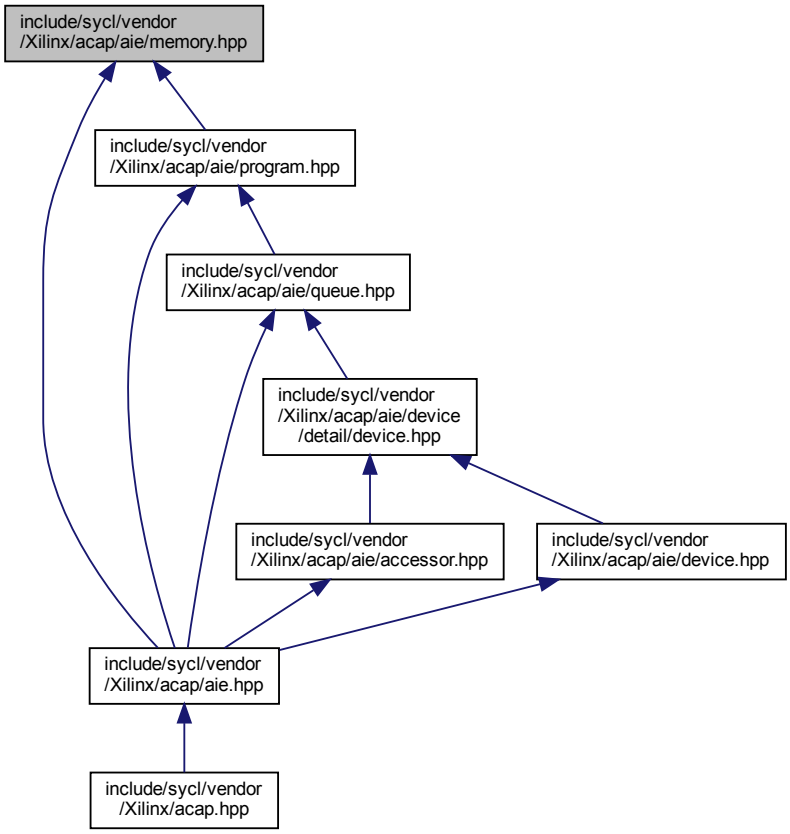
The basic AI Engine heterogeneous Memory Module (i.e.

```
#include "memory_base.hpp"
```

Include dependency graph for memory.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::memory< AIE, X, Y >

  The AI Engine Memory Module infrastructure of a tile. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.325.1 Detailed Description

The basic AI Engine heterogeneous Memory Module (i.e.

dependent of x & y coordinates)

Based on Math Engine (ME) Architecture Specification, Revision v1.4 March 2018

Section 4.5 ME Memory Module, p. 118

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file memory.hpp.

13.326 memory.hpp

Go to the documentation of this file.

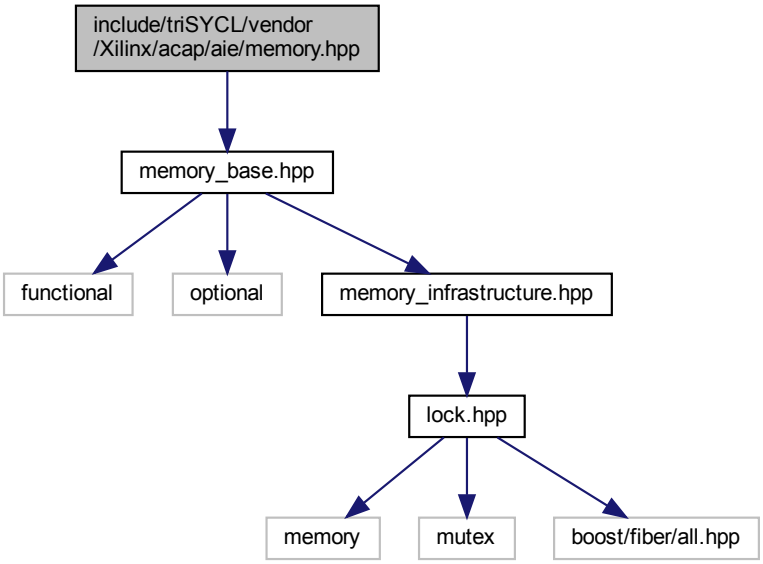
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine heterogeneous Memory Module (i.e.  dependent of
00007 x & y coordinates)
00008
00009 Based on Math Engine (ME) Architecture Specification, Revision v1.4
00010 March 2018
00011
00012 Section 4.5 ME Memory Module, p.  118
00013
00014 Ronan dot Keryell at Xilinx dot com
00015
00016 This file is distributed under the University of Illinois Open Source
00017 License.  See LICENSE.TXT for details.
00018 */
00019
00020 #include "memory_base.hpp"
00021
00022 namespace trisycl::vendor::xilinx::acap::aie {
00023
00024 /// \ingroup aie
00025 /// @{
00026
00027 /** The AI Engine Memory Module infrastructure of a tile
00028
00029 This is the type you need to inherit from to define the content of
00030 a CGRA tile memory module.
00031
00032 \param AIE is the type representing the full CGRA with the
00033 programs and memory contents
00034
00035 \param X is the horizontal coordinate of the memory module
00036
00037 \param Y is the vertical coordinate of the memory module
00038 */
00039 template <typename AIE //< The type representing the full CGRA
00040           , int X //< The horizontal coordinate of the memory module
00041           , int Y //< The vertical coordinate of the memory module
00042           >
00043 struct memory :  memory_base {
00044 /** The horizontal tile coordinates in the CGRA grid (starting at 0
00045 and increasing towards East) */
00046   static auto constexpr x = X;
00047 /** The vertical tile coordinates in the CGRA grid (starting at
00048 increasing towards North) */
00049   static auto constexpr y = Y;
00050 };
00051
00052 /// @} End the aie Doxygen group
00053
00054 } // namespace trisycl::vendor::xilinx::acap::aie
00055
00056 /*
00057 # Some Emacs stuff:
00058 ### Local Variables:
00059 ### ispell-local-dictionary:  "american"
00060 ### eval:  (flyspell-prog-mode)
00061 ### End:
00062 */
00063
00064 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_HPP
```

13.327 include/triSYCL/vendor/Xilinx/acap/aie/memory.hpp File Reference

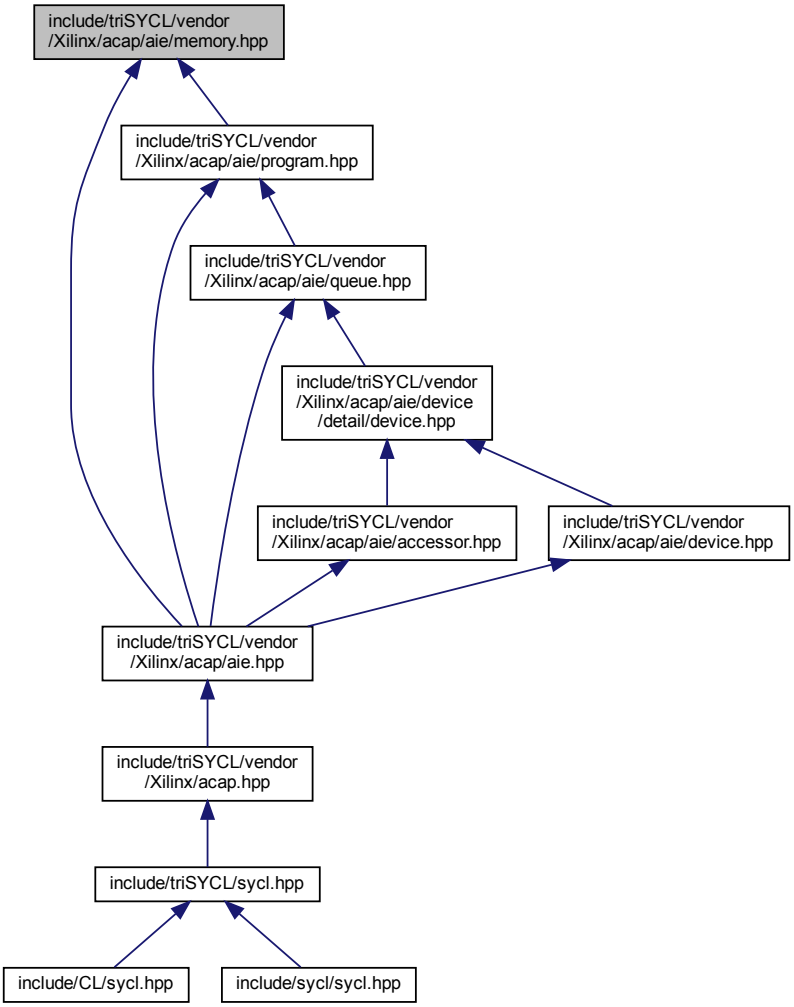
The basic AI Engine heterogeneous Memory Module (i.e.

```
#include "memory_base.hpp"
```

Include dependency graph for memory.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::memory< AIE, X, Y >

  The AI Engine Memory Module infrastructure of a tile. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.327.1 Detailed Description

The basic AI Engine heterogeneous Memory Module (i.e.

dependent of x & y coordinates)

Based on Math Engine (ME) Architecture Specification, Revision v1.4 March 2018

Section 4.5 ME Memory Module, p. 118

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file memory.hpp.

13.328 memory.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine heterogeneous Memory Module (i.e.  dependent of
00007 x & y coordinates)
00008
00009 Based on Math Engine (ME) Architecture Specification, Revision v1.4
00010 March 2018
00011
00012 Section 4.5 ME Memory Module, p.  118
00013
00014 Ronan dot Keryell at Xilinx dot com
00015
00016 This file is distributed under the University of Illinois Open Source
00017 License.  See LICENSE.TXT for details.
00018 */
00019
00020 #include "memory_base.hpp"
00021
00022 namespace trisycl::vendor::xilinx::acap::aie {
00023
00024 /// \ingroup aie
00025 /// @{
00026
00027 /** The AI Engine Memory Module infrastructure of a tile
00028
00029 This is the type you need to inherit from to define the content of
00030 a CGRA tile memory module.
00031
00032 \param AIE is the type representing the full CGRA with the
00033 programs and memory contents
00034
00035 \param X is the horizontal coordinate of the memory module
00036
00037 \param Y is the vertical coordinate of the memory module
00038 */
00039 template <typename AIE //< The type representing the full CGRA
00040           , int X //< The horizontal coordinate of the memory module
00041           , int Y //< The vertical coordinate of the memory module
00042           >
00043 struct memory :  memory_base {
00044 /** The horizontal tile coordinates in the CGRA grid (starting at 0
00045 and increasing towards East) */
00046   static auto constexpr x = X;
00047 /** The vertical tile coordinates in the CGRA grid (starting at
00048 increasing towards North) */
00049   static auto constexpr y = Y;
00050 };
00051
00052 /// @} End the aie Doxygen group
00053
00054 } // namespace trisycl::vendor::xilinx::acap::aie
00055
```

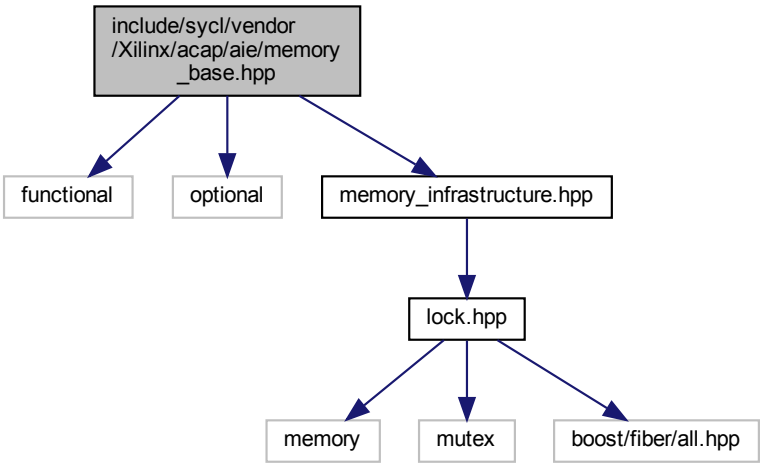
```
00056 /*
00057 # Some Emacs stuff:
00058 ### Local Variables:
00059 ### ispell-local-dictionary:  "american"
00060 ### eval:  (flyspell-prog-mode)
00061 ### End:
00062 */
00063
00064 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_HPP
```

13.329 include/sycl/vendor/Xilinx/acap/aie/memory_base.hpp File Reference

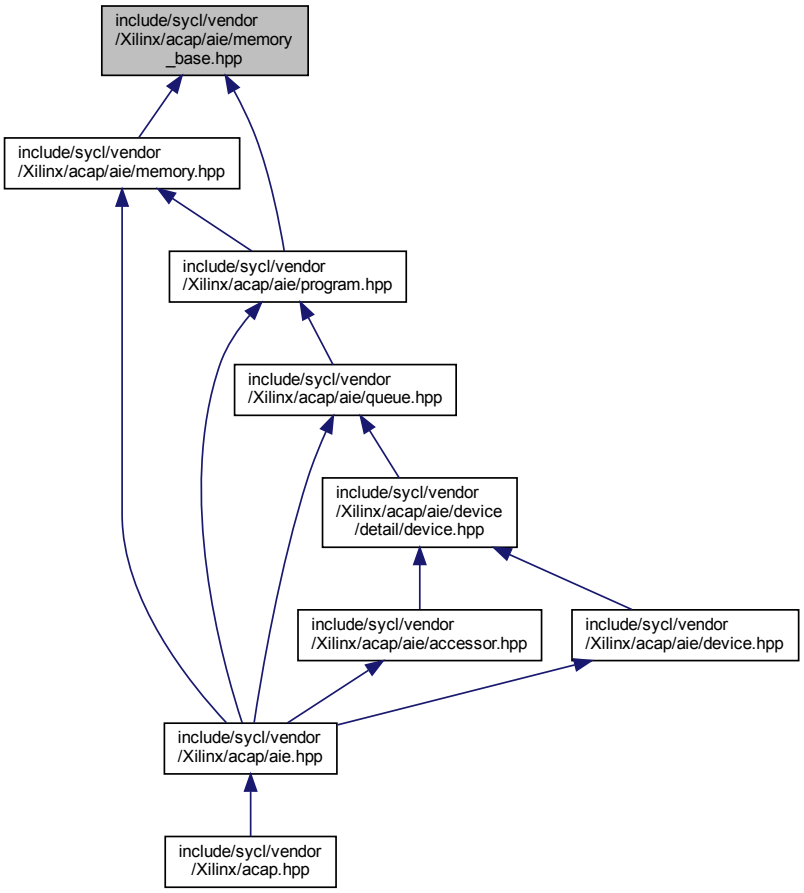
The basic AI Engine homogeneous Memory Module, with the common infrastructure to all the memory modules (i.e.

```
#include <functional>
#include <optional>
#include "memory_infrastructure.hpp"
```

Include dependency graph for memory_base.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::memory_base

  The AI Engine Memory Module infrastructure of a tile common to all the tiles. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.329.1 Detailed Description

The basic AI Engine homogeneous Memory Module, with the common infrastructure to all the memory modules (i.e.

independent of x & y coordinates)

Based on Math Engine (ME) Architecture Specification, Revision v1.4 March 2018

Section 4.5 ME Memory Module, p. 118

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file memory_base.hpp.

13.330 memory_base.hpp

Go to the documentation of this file.

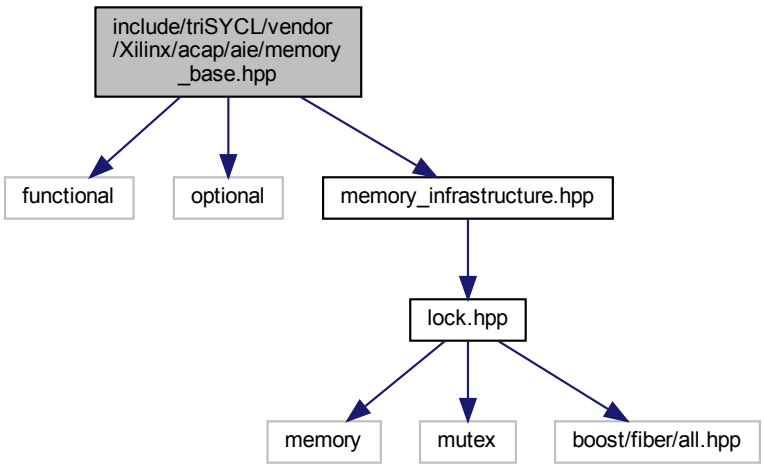
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_BASE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_BASE_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine homogeneous Memory Module, with the common
00007 infrastructure to all the memory modules (i.e.  independent of x & y
00008 coordinates)
00009
00010 Based on Math Engine (ME) Architecture Specification, Revision v1.4
00011 March 2018
00012
00013 Section 4.5 ME Memory Module, p.  118
00014
00015 Ronan dot Keryell at Xilinx dot com
00016
00017 This file is distributed under the University of Illinois Open Source
00018 License.  See LICENSE.TXT for details.
00019 */
00020
00021 #include <functional>
00022 #include <optional>
00023
00024 #include "memory_infrastructure.hpp"
00025
00026 namespace trisycl::vendor::xilinx::acap::aie {
00027
00028 /// \ingroup aie
00029 /// @{
00030
00031 /** The AI Engine Memory Module infrastructure of a tile common to all
00032 the tiles
00033
00034 This allows some type erasure while accessing the common memory
00035 module infrastructure.
00036 */
00037 struct memory_base {
00038 /// Keep a reference to the memory_infrastructure hardware features
00039   std::optional<std::reference_wrapper<memory_infrastructure» mi;
00040
00041 /// Get an access to the a specific lock
00042   auto& lock(int i) {
00043     // value() will throw if there is some missed initialization
00044     return mi.value().get().lock(i);
00045   }
00046
00047 /// Store a way to access to hardware infrastructure of the tile
00048   void set_memory_infrastructure(memory_infrastructure& m) { mi = m; }
00049 };
00050
00051 /// @} End the aie Doxygen group
00052
00053 } // namespace trisycl::vendor::xilinx::acap::aie
00054
00055 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_BASE_HPP
```

13.331 include/triSYCL/vendor/Xilinx/acap/aie/memory_base.hpp File Reference

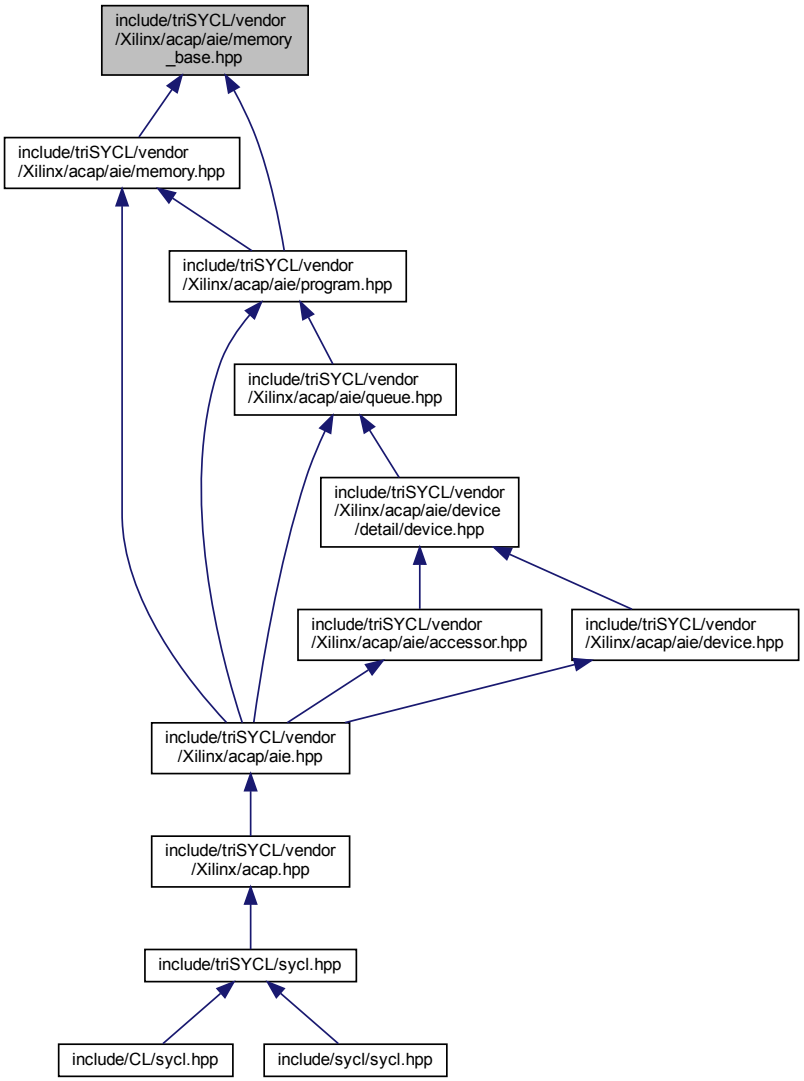
The basic AI Engine homogeneous Memory Module, with the common infrastructure to all the memory modules (i.e.

```
#include <functional>
#include <optional>
#include "memory_infrastructure.hpp"
```

Include dependency graph for memory_base.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::memory_base

  The AI Engine Memory Module infrastructure of a tile common to all the tiles. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.331.1 Detailed Description

The basic AI Engine homogeneous Memory Module, with the common infrastructure to all the memory modules (i.e.

independent of x & y coordinates)

Based on Math Engine (ME) Architecture Specification, Revision v1.4 March 2018

Section 4.5 ME Memory Module, p. 118

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file memory_base.hpp.

13.332 memory_base.hpp

Go to the documentation of this file.

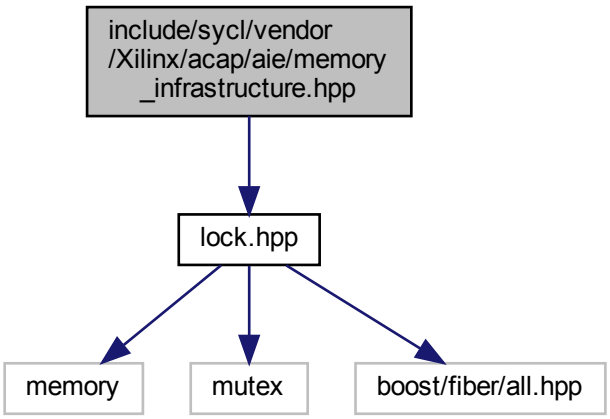
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_BASE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_BASE_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine homogeneous Memory Module, with the common
00007 infrastructure to all the memory modules (i.e.  independent of x & y
00008 coordinates)
00009
00010 Based on Math Engine (ME) Architecture Specification, Revision v1.4
00011 March 2018
00012
00013 Section 4.5 ME Memory Module, p.  118
00014
00015 Ronan dot Keryell at Xilinx dot com
00016
00017 This file is distributed under the University of Illinois Open Source
00018 License.  See LICENSE.TXT for details.
00019 */
00020
00021 #include <functional>
00022 #include <optional>
00023
00024 #include "memory_infrastructure.hpp"
00025
00026 namespace trisycl::vendor::xilinx::acap::aie {
00027
00028 /// \ingroup aie
00029 /// @{
00030
00031 /** The AI Engine Memory Module infrastructure of a tile common to all
00032 the tiles
00033
00034 This allows some type erasure while accessing the common memory
00035 module infrastructure.
00036 */
00037 struct memory_base {
00038 /// Keep a reference to the memory_infrastructure hardware features
00039   std::optional<std::reference_wrapper<memory_infrastructure» mi;
00040
00041 /// Get an access to the a specific lock
00042   auto& lock(int i) {
00043     // value() will throw if there is some missed initialization
00044     return mi.value().get().lock(i);
00045   }
00046
00047 /// Store a way to access to hardware infrastructure of the tile
00048   void set_memory_infrastructure(memory_infrastructure& m) { mi = m; }
00049 };
00050
00051 /// @} End the aie Doxygen group
00052
00053 } // namespace trisycl::vendor::xilinx::acap::aie
00054
00055 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_BASE_HPP
```

13.333 include/sycl/vendor/Xilinx/acap/aie/memory_infrastructure.hpp File Reference

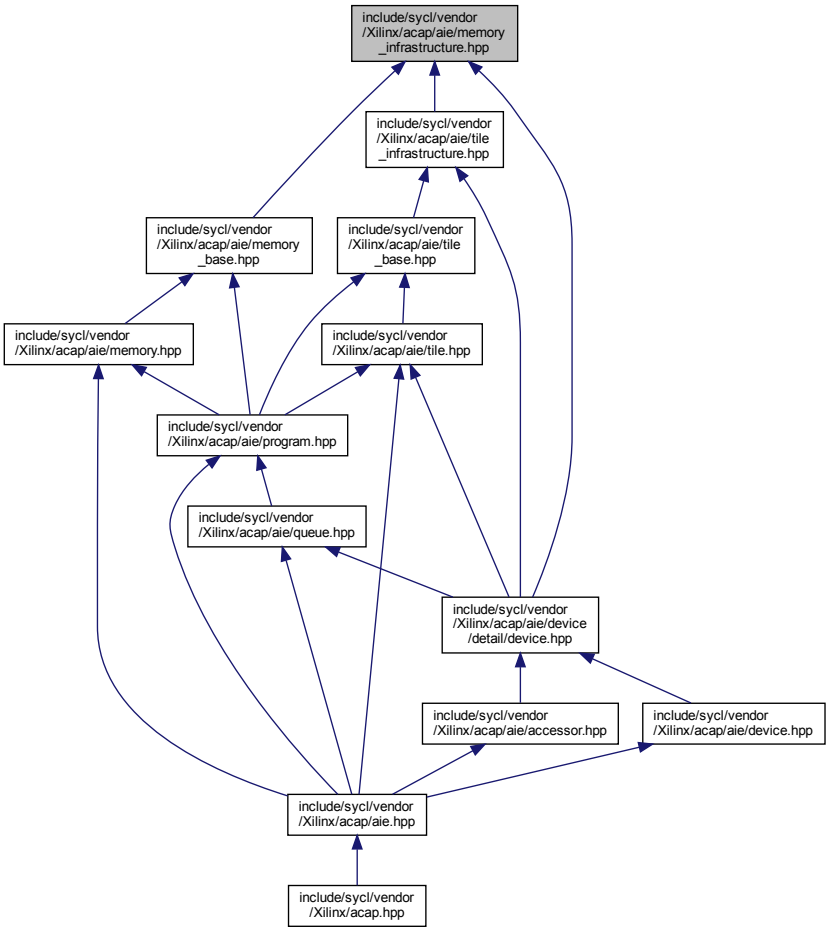
The basic AI Engine homogeneous memory infrastructure, i.e.

```
#include "lock.hpp"
```

Include dependency graph for memory_infrastructure.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::memory_infrastructure

  The AI Engine memory infrastructure common to all the tiles. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.333.1 Detailed Description

The basic AI Engine homogeneous memory infrastructure, i.e.

independent of x & y coordinates.

This is owned by a device tile, so for example the lock configuration can survive to some program changes.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file memory_infrastructure.hpp.

13.334 memory_infrastructure.hpp

Go to the documentation of this file.

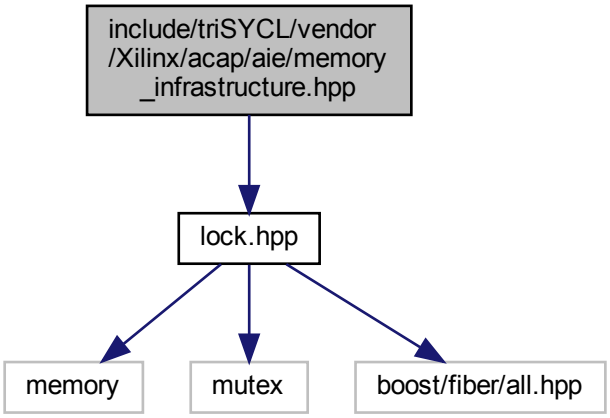
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_INFRASTRUCTURE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_INFRASTRUCTURE_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine homogeneous memory infrastructure,
00007 i.e.  independent of x & y coordinates.
00008
00009 This is owned by a device tile, so for example the lock
00010 configuration can survive to some program changes.
00011
00012 Ronan dot Keryell at Xilinx dot com
00013
00014 This file is distributed under the University of Illinois Open Source
00015 License.  See LICENSE.TXT for details.
00016 */
00017
00018 #include "lock.hpp"
00019
00020 namespace trisycl::vendor::xilinx::acap::aie {
00021
00022 /// \ingroup aie
00023 /// @{
00024
00025 /** The AI Engine memory infrastructure common to all the tiles
00026
00027 This allows some type erasure while accessing the common
00028 memory infrastructure.
00029
00030 \param Geography is the \c geography type representing the full
00031 layout of the architecture
00032 */
00033 class memory_infrastructure {
00034 /// The lock unit of the memory tile
00035   lock_unit memory_locking_unit;
00036
00037 /** Keep track of the aie::detail::device for hardware resource
00038 control in device mode or for debugging purpose for better
00039 messages.
00040
00041 Use void* for now to avoid cyclic header dependencies for now
00042 instead of the aie::detail::device */
00043   void* dev [[maybe_unused]];
00044
00045  public:
00046 /** Start the memory infrastructure associated to the AIE device tile
00047
00048 \param[in] dev is the aie::detail::device used to control
00049 hardware when using real hardware and provide some debug
00050 information from inside the tile_infrastructure.
00051
00052 Use auto concept here to avoid explicit type causing circular
00053 dependency
00054 */
00055   memory_infrastructure(auto& dev)
00056       :  dev { &dev } {}
00057
00058 /// Get access to a specific lock in this memory module
```

```
00059   auto& lock(int i) { return memory_locking_unit.lock(i); }
00060 };
00061
00062 /// @} End the aie Doxygen group
00063
00064 } // namespace trisycl::vendor::xilinx::acap::aie
00065
00066 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_INFRASTRUCTURE_HPP
```

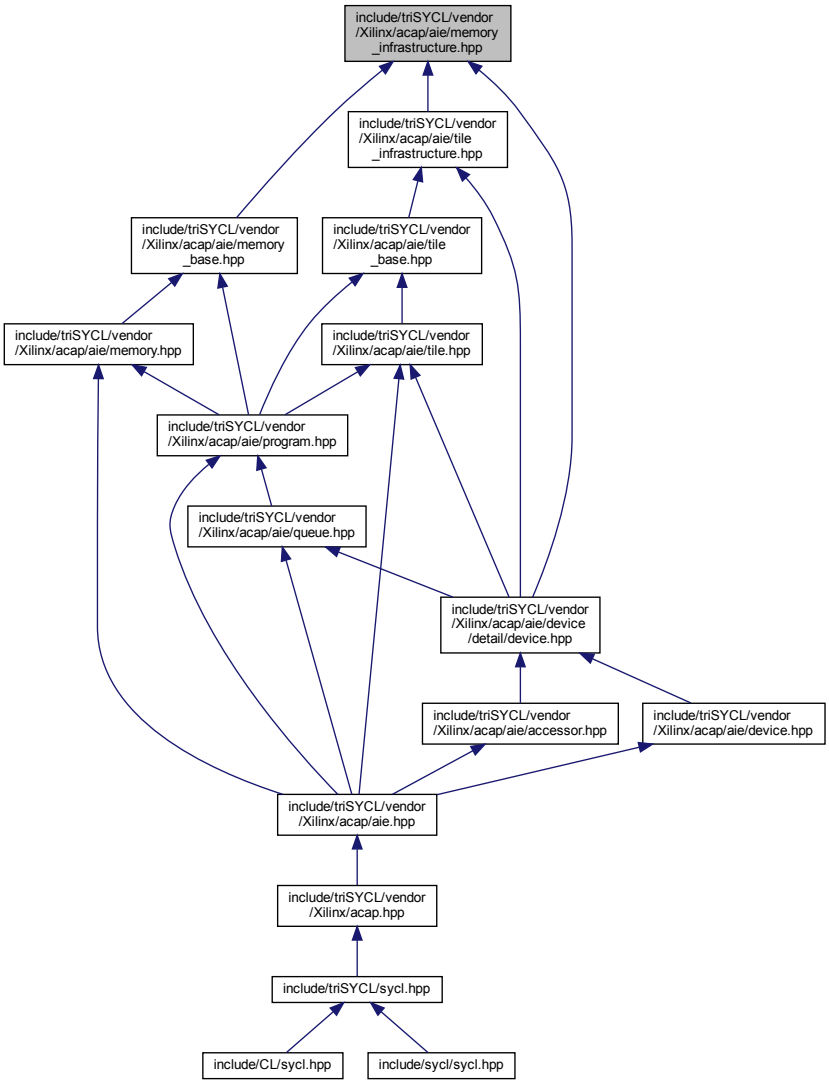
13.335 include/triSYCL/vendor/Xilinx/acap/aie/memory_↩ infrastructure.hpp File Reference

The basic AI Engine homogeneous memory infrastructure, i.e.

`#include "lock.hpp"`
Include dependency graph for memory_infrastructure.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::memory_infrastructure

  The AI Engine memory infrastructure common to all the tiles. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.335.1 Detailed Description

The basic AI Engine homogeneous memory infrastructure, i.e.

independent of x & y coordinates.

This is owned by a device tile, so for example the lock configuration can survive to some program changes.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file memory_infrastructure.hpp.

13.336 memory_infrastructure.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_INFRASTRUCTURE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_INFRASTRUCTURE_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine homogeneous memory infrastructure,
00007 i.e.  independent of x & y coordinates.
00008
00009 This is owned by a device tile, so for example the lock
00010 configuration can survive to some program changes.
00011
00012 Ronan dot Keryell at Xilinx dot com
00013
00014 This file is distributed under the University of Illinois Open Source
00015 License.  See LICENSE.TXT for details.
00016 */
00017
00018 #include "lock.hpp"
00019
00020 namespace trisycl::vendor::xilinx::acap::aie {
00021
00022 /// \ingroup aie
00023 /// @{
00024
00025 /** The AI Engine memory infrastructure common to all the tiles
00026
00027 This allows some type erasure while accessing the common
00028 memory infrastructure.
00029
00030 \param Geography is the \c geography type representing the full
00031 layout of the architecture
00032 */
00033 class memory_infrastructure {
00034 /// The lock unit of the memory tile
00035   lock_unit memory_locking_unit;
00036
00037 /** Keep track of the aie::detail::device for hardware resource
00038 control in device mode or for debugging purpose for better
00039 messages.
00040
00041 Use void* for now to avoid cyclic header dependencies for now
00042 instead of the aie::detail::device */
00043   void* dev [[maybe_unused]];
00044
00045  public:
00046 /** Start the memory infrastructure associated to the AIE device tile
00047
00048 \param[in] dev is the aie::detail::device used to control
00049 hardware when using real hardware and provide some debug
00050 information from inside the tile_infrastructure.
00051
00052 Use auto concept here to avoid explicit type causing circular
00053 dependency
00054 */
00055   memory_infrastructure(auto& dev)
00056       :  dev { &dev } {}
00057
00058 /// Get access to a specific lock in this memory module
```

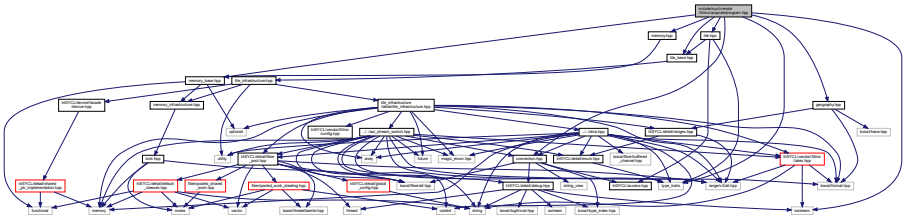
```
00059   auto& lock(int i) { return memory_locking_unit.lock(i); }
00060 };
00061
00062 /// @} End the aie Doxygen group
00063
00064 } // namespace trisycl::vendor::xilinx::acap::aie
00065
00066 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_MEMORY_INFRASTRUCTURE_HPP
```

13.337 include/sycl/vendor/Xilinx/acap/aie/program.hpp File Reference

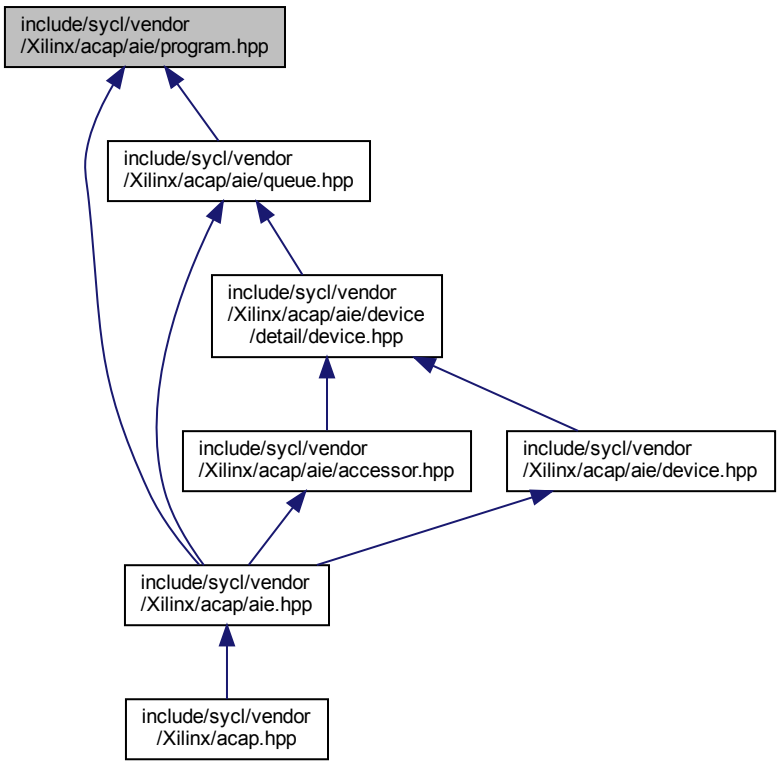
Model of an AI Engine program, that weaves the program of each tile with the memory of each tile for a given device.

```
#include <iostream>
#include <type_traits>
#include "connection.hpp"
#include "geography.hpp"
#include "memory.hpp"
#include "memory_base.hpp"
#include "tile.hpp"
#include "tile_base.hpp"
```

Include dependency graph for program.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >

  Define an AI Engine CGRA program with its code and memory per core. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.337.1 Detailed Description

Model of an AI Engine program, that weaves the program of each tile with the memory of each tile for a given device.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file program.hpp.

13.338 program.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_PROGRAM_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_PROGRAM_HPP
00003
00004 /** \file
00005
00006 Model of an AI Engine program, that weaves the program of each tile
00007 with the memory of each tile for a given device
00008
00009 Ronan dot Keryell at Xilinx dot com
00010
00011 This file is distributed under the University of Illinois Open Source
00012 License.  See LICENSE.TXT for details.
00013 */
00014
00015 #include <iostream>
00016 #include <type_traits>
00017
00018 #include "connection.hpp"
00019 #include "geography.hpp"
00020 #include "memory.hpp"
00021 #include "memory_base.hpp"
00022 #include "tile.hpp"
00023 #include "tile_base.hpp"
00024
00025 /// \ingroup acap
00026 ///  @{
00027
00028 /** \defgroup aie AI Engine CGRA
00029
00030 Extensions to support explicit AI Engine system-wide programming in C++
00031 @{
00032 */
00033
00034 namespace trisycl::vendor::xilinx::acap::aie {
00035
00036 /** Define an AI Engine CGRA program with its code and memory per core
00037
00038 \param AIEDevice is the device description of the machine to
00039 instantiate with the physical size
00040
00041 \param Tile is the description of the program tiles to
00042 instantiate.  By default each tile will run an empty program.
00043
00044 \param Memory is the description of the machine memory modules.  By
00045 default the machine has empty memory modules.
00046 */
00047 template <typename AIEDevice,
00048           template <typename AIE,
00049                     int X,
00050                     int Y> typename Tile = acap::aie::tile,
00051           template <typename AIE,
00052                     int X,
00053                     int Y> typename Memory = acap::aie::memory>
00054 struct program {
00055
00056 /// The geography of the CGRA
00057   using geo = typename AIEDevice::geo;
00058   using device = AIEDevice;
00059
00060 /// The device running this program
00061   AIEDevice aie_d;
00062
00063 /// Type describing all the memory modules of the CGRA
00064   template <int X, int Y>
00065   using tileable_memory = Memory<program, X, Y>;
00066
00067 /** The tiled memory modules of the CGRA
00068
00069 Unfortunately it is not possible to use \c auto here...
00070 Otherwise it could be just:  \code static inline auto \endcode */
00071   decltype(geo::template generate_tiles<tileable_memory>())
00072   memory_modules = geo::template generate_tiles<tileable_memory>();
00073
00074 /** Keep track of all the tiled memory modules as a type-erased
00075 memory_modules_base type to have a simpler access to the basic
00076 position-independent memory module features */
00077   memory_base *memory_modules_bases[geo::y_size][geo::x_size];
00078
00079 /// Type describing the programs of all the cores in the CGRA
00080   template <int X, int Y>
00081   using tileable_tile = Tile<program, X, Y>;
00082
```

```
00083 /** The tiled programs of the CGRA
00084
00085 Unfortunately it is not possible to use \c auto here...
00086 Otherwise it could be just:  \code static inline auto \endcode */
00087   decltype(geo::template generate_tiles<tileable_tile>()) tiles =
00088     geo::template generate_tiles<tileable_tile>();
00089
00090 /** Keep track of all the tiles as a type-erased tile_base type to
00091 have a simpler access to the basic position-independent tile
00092 features */
00093   tile_base<program> *tile_bases[geo::y_size][geo::x_size];
00094
00095 /** Access to the common infrastructure part of a memory module
00096
00097 \param[in] x is the horizontal memory module coordinate
00098
00099 \param[in] y is the vertical memory module coordinate
00100 */
00101   memory_base &memory_module(int x, int y) {
00102     geo::validate_x_y(x, y);
00103     return *memory_modules_bases[y][x];
00104   }
00105
00106
00107 /** Access to a heterogeneous memory module by its linear id
00108
00109 \param[in] LinearId is the linear id
00110 */
00111   template <int LinearId>
00112   auto &memory_module() {
00113     return boost::hana::at_c<LinearId>(memory_modules);
00114   }
00115
00116
00117 /** Access to a heterogeneous memory module by its coordinates
00118
00119 \param[in] X is the horizontal memory module coordinate
00120
00121 \param[in] Y is the vertical memory module coordinate
00122 */
00123   template <int X, int Y>
00124   auto &memory_module() {
00125     return memory_module<geo::linear_id(X, Y)>();
00126   }
00127
00128
00129 /** Iterate on all the memory module bases of the AIE in an
00130 homogeneous way
00131
00132 \param[in] F is the function to apply on each memory module base
00133 */
00134   template <typename F>
00135   void for_each_memory_base(F && f) {
00136     for (auto y = 0; y != geo::y_size; ++y)
00137       for (auto x = 0; x != geo::x_size; ++x)
00138         f(*memory_modules_bases[y][x]);
00139   }
00140
00141
00142 /** Access to a heterogeneous tile by linear id
00143
00144 \param[in] LinearId is the linear id
00145 */
00146   template <int LinearId>
00147   auto &tile() {
00148     return boost::hana::at_c<LinearId>(tiles);
00149   }
00150
00151
00152 /** Access to a heterogeneous tile by its coordinates
00153
00154 \param[in] X is the horizontal tile coordinate
00155
00156 \param[in] Y is the vertical tile coordinate
00157 */
00158   template <int X, int Y>
00159   auto &tile() {
00160     return tile<geo::linear_id(X, Y)>();
00161   }
00162
00163
00164 /** Iterate on all the tile bases of the AIE in an homogeneous way
00165
00166 \param[in] F is the function to apply on each tile base
00167 */
00168   template <typename F>
00169   void for_each_tile_base(F && f) {
```

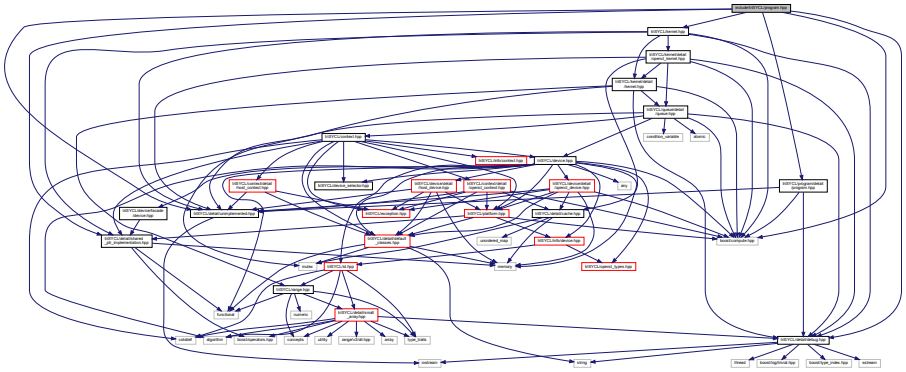
```
00170     for (auto y = 0; y != geo::y_size; ++y)
00171       for (auto x = 0; x != geo::x_size; ++x)
00172         f(*tile_bases[y][x]);
00173   }
00174
00175
00176 /// Create the AIE program with the tiles and memory modules
00177   program(AIEDevice &aie_d) :  aie_d { aie_d } {
00178     boost::hana::for_each(tiles, [&] (auto& t) {
00179         // Inform each tile about its program
00180         t.set_program(*this);
00181         // Inform each tile about their tile infrastructure
00182         t.set_tile_infrastructure(aie_d.tile(t.x, t.y));
00183         // Keep track of each base tile
00184         tile_bases[t.y][t.x] = &t;
00185       });
00186     // Connect each memory to its infrastructure
00187     boost::hana::for_each(memory_modules, [&] (auto& m) {
00188         // Inform each tile about their tile infrastructure
00189         m.set_memory_infrastructure(aie_d.mem(m.x, m.y));
00190         // Keep track of each base tile
00191         memory_modules_bases[m.y][m.x] = &m;
00192       });
00193   }
00194
00195
00196 /** Instantiate a kernel in a form that can be outlined by the SYCL
00197 device compiler
00198
00199 \param[in] KernelName k is the kernel name type required by SYCL
00200
00201 \param[in] k is the kernel functor
00202 */
00203   template <typename KernelName, typename KernelType>
00204 #ifdef __SYCL_DEVICE_ONLY__
00205   __attribute__((sycl_kernel))
00206 #endif
00207   void kernel_outliner(KernelType k) {
00208     k();
00209   }
00210
00211 /// Wait for the end of the execution of each tile
00212   void wait() {
00213     boost::hana::for_each(tiles, [&] (auto& t) {
00214         TRISYCL_DUMP_T("Joining AIE tile (" « t.x « ',' « t.y « ")...");
00215         t.wait();
00216         TRISYCL_DUMP_T("Joined AIE tile (" « t.x « ',' « t.y « ')');
00217       });
00218   }
00219
00220 /** Launch the programs of all the tiles of the CGRA in their own
00221 executor (CPU thread, fiber...)  and wait for their completion.
00222
00223 This is the main member function to use to launch the execution.
00224 */
00225   void run() {
00226     // Start each tile program in its own executor
00227     boost::hana::for_each(tiles, [&] (auto& t) {
00228         t.single_task([&] {
00229             TRISYCL_DUMP_T("Starting AIE tile (" « t.x « ',' « t.y
00230                            « ") linear id = " « t.linear_id());
00231             /* Just use a capture by reference in the following
00232 because there is direct execution here */
00233             auto kernel = [&] {
00234                               // If the tile has an operator(), use it
00235                               if constexpr (requires { t(); })
00236                                 return [&] { t(); };
00237                               /* Else the kernel should have a run
00238 member function and use it.  */
00239                               else
00240                                 return [&] { t.run(); };
00241             }();
00242             using kernel_type = decltype(kernel);
00243             // Use the kernel type as its SYCL name too
00244             kernel_outliner<kernel_type, kernel_type>(kernel);
00245             TRISYCL_DUMP_T("Stopping AIE tile (" « t.x « ',' « t.y « ')');
00246           });
00247       });
00248     wait();
00249   }
00250
00251 /** Run synchronously an heterogeneous invocable collectively on the device
00252
00253 \param f is an invocable taking an heterogeneous tile handler
00254
00255 \todo Factorize out the 2 run functions
00256 */
```

```
00257   template <typename Invocable>
00258   void run(Invocable&& f) {
00259     // Start each tile program in its own executor
00260     boost::hana::for_each(tiles, [&] (auto& t) {
00261         t.single_task([&] {
00262             TRISYCL_DUMP_T("Starting AIE tile (" « t.x « ',' « t.y
00263                            « ") linear id = " « t.linear_id());
00264 /// Each tile gets its own copy of work
00265             auto kernel = [&, work = f] { work(t); };
00266             using kernel_type = decltype(kernel);
00267             // Use the kernel type as its SYCL name too
00268             kernel_outliner<kernel_type, kernel_type>(kernel);
00269             TRISYCL_DUMP_T("Stopping AIE tile (" « t.x « ',' « t.y « ')');
00270           });
00271       });
00272     wait();
00273   }
00274
00275 /// Access the cascade connections
00276   auto &cascade() {
00277     return aie_d.cascade();
00278   }
00279
00280 };
00281
00282 /// @} End the aie Doxygen group
00283 /// @} End the acap Doxygen group
00284
00285 }
00286
00287 /*
00288 # Some Emacs stuff:
00289 ### Local Variables:
00290 ### ispell-local-dictionary:  "american"
00291 ### eval:  (flyspell-prog-mode)
00292 ### End:
00293 */
00294
00295 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_PROGRAM_HPP
```

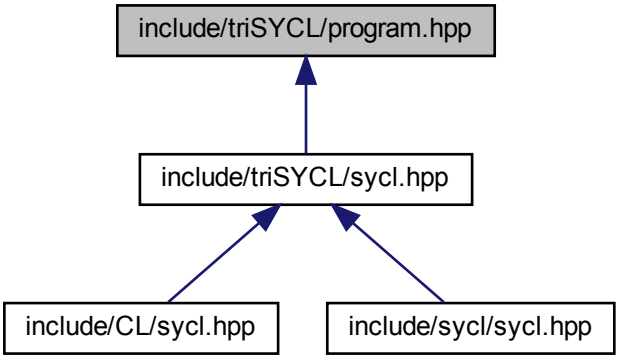
13.339 include/triSYCL/program.hpp File Reference

```
#include <boost/compute.hpp>
#include "triSYCL/detail/debug.hpp"
#include "triSYCL/detail/shared_ptr_implementation.hpp"
#include "triSYCL/detail/unimplemented.hpp"
#include "triSYCL/kernel.hpp"
#include "triSYCL/program/detail/program.hpp"
```

Include dependency graph for program.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::program

  SYCL program.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

Enumerations

- enum class trisycl::program_state { trisycl::none , trisycl::compiled , trisycl::linked }

13.340 program.hpp

Go to the documentation of this file.

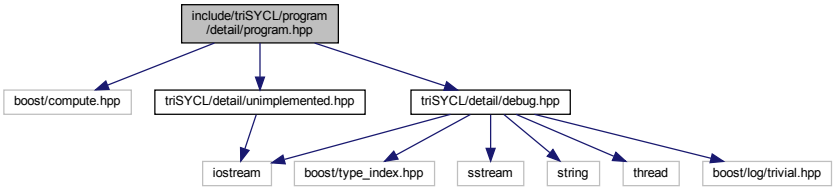
```
00001 #ifndef TRISYCL_SYCL_PROGRAM_HPP
00002 #define TRISYCL_SYCL_PROGRAM_HPP
00003
00004 /** \file The OpenCL SYCL program
00005
00006 Dave Airlie
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #ifdef TRISYCL_OPENCL
00013 #include <boost/compute.hpp>
00014 #endif
00015
00016 #include "triSYCL/detail/debug.hpp"
00017 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00018 #include "triSYCL/detail/unimplemented.hpp"
00019 #include "triSYCL/kernel.hpp"
```

```
00020 #include "triSYCL/program/detail/program.hpp"
00021
00022 namespace trisycl {
00023
00024 enum class program_state {
00025   none,
00026   compiled,
00027   linked
00028 };
00029
00030 /** SYCL program
00031
00032 \todo To be implemented
00033
00034 */
00035
00036 class program
00037     /* Use the underlying kernel implementation that can be shared in
00038 the SYCL model */
00039   :  public detail::shared_ptr_implementation<program, detail::program> {
00040
00041   // The type encapsulating the implementation
00042   using implementation_t = typename program::shared_ptr_implementation;
00043
00044   // The handler class uses the implementation
00045   friend class handler;
00046
00047   // Allows the comparison operation to access the implementation
00048   friend implementation_t;
00049
00050  public:
00051
00052   // Make the implementation member directly accessible in this class
00053   using implementation_t::implementation;
00054
00055   program() = delete;
00056
00057 #ifdef TRISYCL_OPENCL
00058
00059   program(const context &context, cl_program p) {}
00060
00061   cl_program get()const {
00062     return implementation->get();
00063   }
00064
00065   template <typename kernelT>
00066   bool has_kernel() const;
00067
00068   bool has_kernel(string_class kernelName) const;
00069
00070   template <typename kernelT>
00071   void compile_with_kernel_type(string_class compileOptions = "")
00072   {
00073
00074   }
00075
00076   template <typename kernelT>
00077   void build_with_kernel_type(string_class buildOptions = "")
00078   {
00079   }
00080
00081   template <typename kernelT>
00082   ::trisycl::kernel get_kernel() const;
00083
00084   kernel get_kernel(string_class kernelName)const {
00085     return NULL;
00086   }
00087 #endif
00088 };
00089
00090 }
00091
00092 /*
00093 # Some Emacs stuff:
00094 ### Local Variables:
00095 ### ispell-local-dictionary:  "american"
00096 ### eval:  (flyspell-prog-mode)
00097 ### End:
00098 */
00099
00100 #endif // TRISYCL_SYCL_PROGRAM_HPP
```

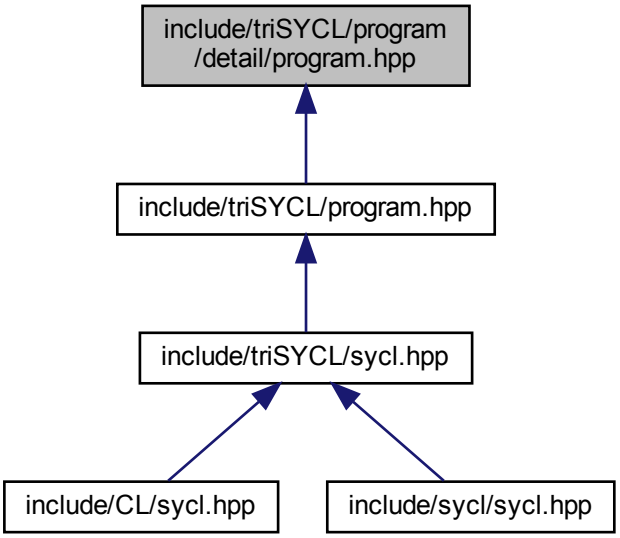
13.341 include/triSYCL/program/detail/program.hpp File Reference

```
#include <boost/compute.hpp>
#include "triSYCL/detail/debug.hpp"
#include "triSYCL/detail/unimplemented.hpp"
```

Include dependency graph for program.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::detail::program

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.342 program.hpp

Go to the documentation of this file.

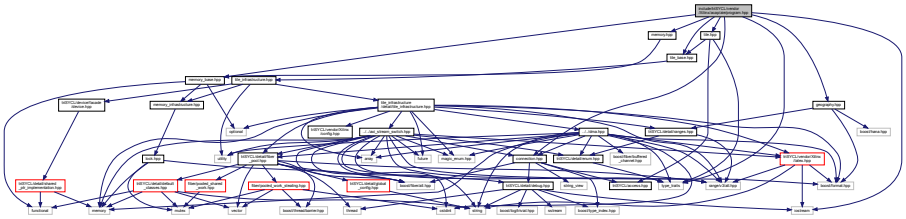
```
00001 #ifndef TRISYCL_SYCL_PROGRAM_DETAIL_PROGRAM_HPP
00002 #define TRISYCL_SYCL_PROGRAM_DETAIL_PROGRAM_HPP
00003
00004 /** \file The OpenCL SYCL program
00005
00006 Dave Airlie
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011 #ifdef TRISYCL_OPENCL
00012 #include <boost/compute.hpp>
00013 #endif
00014
00015 #include "triSYCL/detail/debug.hpp"
00016 #include "triSYCL/detail/unimplemented.hpp"
00017
00018 namespace trisycl::detail {
00019
00020 class program :  detail::debug<detail::program> {
00021
00022   public:
00023
00024 #ifdef TRISYCL_OPENCL
00025 /** Return the OpenCL kernel object for this kernel
00026
00027 Retains a reference to the returned cl_kernel object.  Caller
00028 should release it when finished.
00029 */
00030   virtual cl_program get() const = 0;
00031 #endif
00032
00033   virtual ~program() {}
00034 };
00035
00036 }
00037
00038 /*
00039 # Some Emacs stuff:
00040 ### Local Variables:
00041 ### ispell-local-dictionary:  "american"
00042 ### eval:  (flyspell-prog-mode)
00043 ### End:
00044 */
00045
00046 #endif // TRISYCL_SYCL_DETAIL_PROGRAM_PROGRAM_HPP
00047
00048
```

13.343 include/triSYCL/vendor/Xilinx/acap/aie/program.hpp File Reference

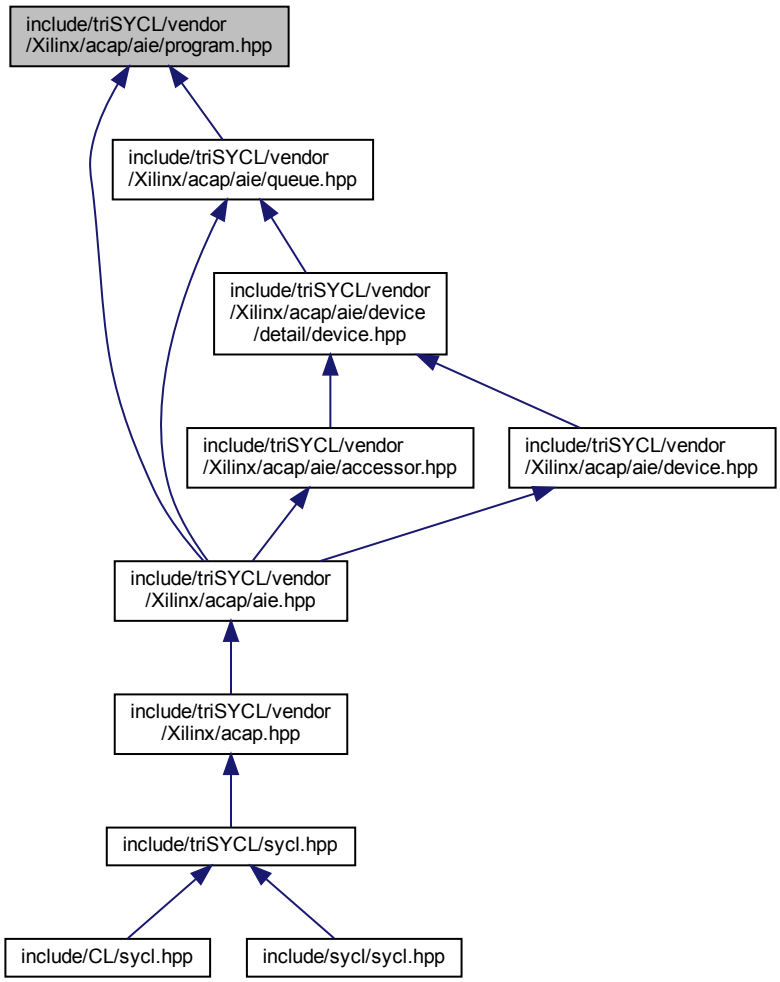
Model of an AI Engine program, that weaves the program of each tile with the memory of each tile for a given device.

```
#include <iostream>
#include <type_traits>
#include "connection.hpp"
#include "geography.hpp"
#include "memory.hpp"
#include "memory_base.hpp"
#include "tile.hpp"
#include "tile_base.hpp"
```

Include dependency graph for program.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >

  Define an AI Engine CGRA program with its code and memory per core. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.343.1 Detailed Description

Model of an AI Engine program, that weaves the program of each tile with the memory of each tile for a given device.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file program.hpp.

13.344 program.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_PROGRAM_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_PROGRAM_HPP
00003
00004 /** \file
00005
00006 Model of an AI Engine program, that weaves the program of each tile
00007 with the memory of each tile for a given device
00008
00009 Ronan dot Keryell at Xilinx dot com
00010
00011 This file is distributed under the University of Illinois Open Source
00012 License.  See LICENSE.TXT for details.
00013 */
00014
00015 #include <iostream>
00016 #include <type_traits>
00017
00018 #include "connection.hpp"
00019 #include "geography.hpp"
00020 #include "memory.hpp"
00021 #include "memory_base.hpp"
00022 #include "tile.hpp"
00023 #include "tile_base.hpp"
00024
00025 /// \ingroup acap
00026 ///  @{
00027
00028 /** \defgroup aie AI Engine CGRA
00029
00030 Extensions to support explicit AI Engine system-wide programming in C++
00031 @{
00032 */
00033
00034 namespace trisycl::vendor::xilinx::acap::aie {
00035
00036 /** Define an AI Engine CGRA program with its code and memory per core
00037
00038 \param AIEDevice is the device description of the machine to
00039 instantiate with the physical size
00040
00041 \param Tile is the description of the program tiles to
00042 instantiate.  By default each tile will run an empty program.
00043
00044 \param Memory is the description of the machine memory modules.  By
00045 default the machine has empty memory modules.
00046 */
00047 template <typename AIEDevice,
```

```
00048           template <typename AIE,
00049                     int X,
00050                     int Y> typename Tile = acap::aie::tile,
00051           template <typename AIE,
00052                     int X,
00053                     int Y> typename Memory = acap::aie::memory>
00054 struct program {
00055
00056 /// The geography of the CGRA
00057   using geo = typename AIEDevice::geo;
00058   using device = AIEDevice;
00059
00060 /// The device running this program
00061   AIEDevice aie_d;
00062
00063 /// Type describing all the memory modules of the CGRA
00064   template <int X, int Y>
00065   using tileable_memory = Memory<program, X, Y>;
00066
00067 /** The tiled memory modules of the CGRA
00068
00069 Unfortunately it is not possible to use \c auto here...
00070 Otherwise it could be just:  \code static inline auto \endcode */
00071   decltype(geo::template generate_tiles<tileable_memory>())
00072   memory_modules = geo::template generate_tiles<tileable_memory>();
00073
00074 /** Keep track of all the tiled memory modules as a type-erased
00075 memory_modules_base type to have a simpler access to the basic
00076 position-independent memory module features */
00077   memory_base *memory_modules_bases[geo::y_size][geo::x_size];
00078
00079 /// Type describing the programs of all the cores in the CGRA
00080   template <int X, int Y>
00081   using tileable_tile = Tile<program, X, Y>;
00082
00083 /** The tiled programs of the CGRA
00084
00085 Unfortunately it is not possible to use \c auto here...
00086 Otherwise it could be just:  \code static inline auto \endcode */
00087   decltype(geo::template generate_tiles<tileable_tile>()) tiles =
00088     geo::template generate_tiles<tileable_tile>();
00089
00090 /** Keep track of all the tiles as a type-erased tile_base type to
00091 have a simpler access to the basic position-independent tile
00092 features */
00093   tile_base<program> *tile_bases[geo::y_size][geo::x_size];
00094
00095 /** Access to the common infrastructure part of a memory module
00096
00097 \param[in] x is the horizontal memory module coordinate
00098
00099 \param[in] y is the vertical memory module coordinate
00100 */
00101   memory_base &memory_module(int x, int y) {
00102     geo::validate_x_y(x, y);
00103     return *memory_modules_bases[y][x];
00104   }
00105
00106
00107 /** Access to a heterogeneous memory module by its linear id
00108
00109 \param[in] LinearId is the linear id
00110 */
00111   template <int LinearId>
00112   auto &memory_module() {
00113     return boost::hana::at_c<LinearId>(memory_modules);
00114   }
00115
00116
00117 /** Access to a heterogeneous memory module by its coordinates
00118
00119 \param[in] X is the horizontal memory module coordinate
00120
00121 \param[in] Y is the vertical memory module coordinate
00122 */
00123   template <int X, int Y>
00124   auto &memory_module() {
00125     return memory_module<geo::linear_id(X, Y)>();
00126   }
00127
00128
00129 /** Iterate on all the memory module bases of the AIE in an
00130 homogeneous way
00131
00132 \param[in] F is the function to apply on each memory module base
00133 */
00134   template <typename F>
```

```
00135   void for_each_memory_base(F && f) {
00136     for (auto y = 0; y != geo::y_size; ++y)
00137       for (auto x = 0; x != geo::x_size; ++x)
00138         f(*memory_modules_bases[y][x]);
00139   }
00140
00141
00142 /** Access to a heterogeneous tile by linear id
00143
00144 \param[in] LinearId is the linear id
00145 */
00146   template <int LinearId>
00147   auto &tile() {
00148     return boost::hana::at_c<LinearId>(tiles);
00149   }
00150
00151
00152 /** Access to a heterogeneous tile by its coordinates
00153
00154 \param[in] X is the horizontal tile coordinate
00155
00156 \param[in] Y is the vertical tile coordinate
00157 */
00158   template <int X, int Y>
00159   auto &tile() {
00160     return tile<geo::linear_id(X, Y)>();
00161   }
00162
00163
00164 /** Iterate on all the tile bases of the AIE in an homogeneous way
00165
00166 \param[in] F is the function to apply on each tile base
00167 */
00168   template <typename F>
00169   void for_each_tile_base(F && f) {
00170     for (auto y = 0; y != geo::y_size; ++y)
00171       for (auto x = 0; x != geo::x_size; ++x)
00172         f(*tile_bases[y][x]);
00173   }
00174
00175
00176 /// Create the AIE program with the tiles and memory modules
00177   program(AIEDevice &aie_d) :  aie_d { aie_d } {
00178     boost::hana::for_each(tiles, [&] (auto& t) {
00179         // Inform each tile about its program
00180         t.set_program(*this);
00181         // Inform each tile about their tile infrastructure
00182         t.set_tile_infrastructure(aie_d.tile(t.x, t.y));
00183         // Keep track of each base tile
00184         tile_bases[t.y][t.x] = &t;
00185       });
00186     // Connect each memory to its infrastructure
00187     boost::hana::for_each(memory_modules, [&] (auto& m) {
00188         // Inform each tile about their tile infrastructure
00189         m.set_memory_infrastructure(aie_d.mem(m.x, m.y));
00190         // Keep track of each base tile
00191         memory_modules_bases[m.y][m.x] = &m;
00192       });
00193   }
00194
00195
00196 /** Instantiate a kernel in a form that can be outlined by the SYCL
00197 device compiler
00198
00199 \param[in] KernelName k is the kernel name type required by SYCL
00200
00201 \param[in] k is the kernel functor
00202 */
00203   template <typename KernelName, typename KernelType>
00204 #ifdef __SYCL_DEVICE_ONLY__
00205   __attribute__((sycl_kernel))
00206 #endif
00207   void kernel_outliner(KernelType k) {
00208     k();
00209   }
00210
00211 /// Wait for the end of the execution of each tile
00212   void wait() {
00213     boost::hana::for_each(tiles, [&] (auto& t) {
00214         TRISYCL_DUMP_T("Joining AIE tile (" « t.x « ',' « t.y « ")...");
00215         t.wait();
00216         TRISYCL_DUMP_T("Joined AIE tile (" « t.x « ',' « t.y « ')');
00217       });
00218   }
00219
00220 /** Launch the programs of all the tiles of the CGRA in their own
00221 executor (CPU thread, fiber...)  and wait for their completion.
```

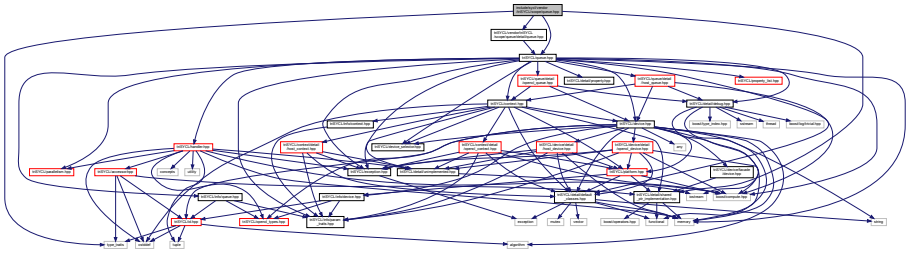
```
00222
00223 This is the main member function to use to launch the execution.
00224 */
00225   void run() {
00226     // Start each tile program in its own executor
00227     boost::hana::for_each(tiles, [&] (auto& t) {
00228         t.single_task([&] {
00229             TRISYCL_DUMP_T("Starting AIE tile (" « t.x « ',' « t.y
00230                            « ") linear id = " « t.linear_id());
00231             /* Just use a capture by reference in the following
00232 because there is direct execution here */
00233             auto kernel = [&] {
00234                               // If the tile has an operator(), use it
00235                               if constexpr (requires { t(); })
00236                                 return [&] { t(); };
00237                               /* Else the kernel should have a run
00238 member function and use it.  */
00239                               else
00240                                 return [&] { t.run(); };
00241             }();
00242             using kernel_type = decltype(kernel);
00243             // Use the kernel type as its SYCL name too
00244             kernel_outliner<kernel_type, kernel_type>(kernel);
00245             TRISYCL_DUMP_T("Stopping AIE tile (" « t.x « ',' « t.y « ')');
00246           });
00247       });
00248     wait();
00249   }
00250
00251 /** Run synchronously an heterogeneous invocable collectively on the device
00252
00253 \param f is an invocable taking an heterogeneous tile handler
00254
00255 \todo Factorize out the 2 run functions
00256 */
00257   template <typename Invocable>
00258   void run(Invocable&& f) {
00259     // Start each tile program in its own executor
00260     boost::hana::for_each(tiles, [&] (auto& t) {
00261         t.single_task([&] {
00262             TRISYCL_DUMP_T("Starting AIE tile (" « t.x « ',' « t.y
00263                            « ") linear id = " « t.linear_id());
00264 /// Each tile gets its own copy of work
00265             auto kernel = [&, work = f] { work(t); };
00266             using kernel_type = decltype(kernel);
00267             // Use the kernel type as its SYCL name too
00268             kernel_outliner<kernel_type, kernel_type>(kernel);
00269             TRISYCL_DUMP_T("Stopping AIE tile (" « t.x « ',' « t.y « ')');
00270           });
00271       });
00272     wait();
00273   }
00274
00275 /// Access the cascade connections
00276   auto &cascade() {
00277     return aie_d.cascade();
00278   }
00279
00280 };
00281
00282 /// @} End the aie Doxygen group
00283 /// @} End the acap Doxygen group
00284
00285 }
00286
00287 /*
00288 # Some Emacs stuff:
00289 ### Local Variables:
00290 ### ispell-local-dictionary:  "american"
00291 ### eval:  (flyspell-prog-mode)
00292 ### End:
00293 */
00294
00295 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_PROGRAM_HPP
```

13.345 include/sycl/vendor/triSYCL/scope/queue.hpp File Reference

```
#include <type_traits>
#include "triSYCL/detail/shared_ptr_implementation.hpp"
#include "triSYCL/queue.hpp"
```

```
#include "triSYCL/vendor/triSYCL/scope/queue/detail/queue.hpp"
```

Include dependency graph for queue.hpp:



Classes

- struct trisycl::vendor::trisycl::scope::empty_queue_scope

  An empty queue scope to make compilation errors clearer. More...
- class trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >

  A conceptual queue implementing some queue-scoped storage constructed from a device with some device-scope storage. More...
- struct trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >

  The command group redefining some member functions to be able to pass the scope and so on to the kernel.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope

Macros

- #define SYCL_VENDOR_TRISYCL_QUEUE_SCOPE 1

  This is an extension providing scope storage for queues.

13.345.1 Macro Definition Documentation

13.345.1.1 SYCL_VENDOR_TRISYCL_QUEUE_SCOPE

```
#define SYCL_VENDOR_TRISYCL_QUEUE_SCOPE 1
```

This is an extension providing scope storage for queues.

Definition at line 27 of file queue.hpp.

13.346 queue.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_QUEUE_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_QUEUE_HPP
00003
00004 /** \file The SYCL conceptual queue with a queue scope storage
00005
00006
00007 \todo Add some caching or prevent multiple creation of storage on
00008 the queue ?  Or is this a feature to have different scopes on
00009 the same queue?
00010
00011 \todo The device and queue are very similar.  Use a common
00012 mix-in to implement both of them?
00013
00014 Ronan at Keryell point FR
00015
00016 This file is distributed under the University of Illinois Open Source
00017 License.  See LICENSE.TXT for details.
00018 */
00019
00020 #include <type_traits>
00021
00022 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00023 #include "triSYCL/queue.hpp"
00024 #include "triSYCL/vendor/triSYCL/scope/queue/detail/queue.hpp"
00025
00026 /// This is an extension providing scope storage for queues
00027 #define SYCL_VENDOR_TRISYCL_QUEUE_SCOPE 1
00028
00029 namespace trisycl::vendor::trisycl::scope {
00030
00031 /** \addtogroup vendor_trisycl_scope triSYCL extension for storage scopes
00032 @{
00033 */
00034
00035 /// An empty queue scope to make compilation errors clearer
00036 struct empty_queue_scope {
00037 };
00038
00039
00040 /** A conceptual queue implementing some queue-scoped storage
00041 constructed from a device with some device-scope storage
00042 */
00043 template <typename Device,
00044           typename QueueStorage = empty_queue_scope>
00045 class queue
00046   /* Use the underlying queue implementation that can be shared in the
00047 SYCL model */
00048   :  public
00049     ::trisycl::detail::shared_ptr_implementation<queue<Device, QueueStorage>,
00050                                                  detail::queue
00051                                                  <Device, QueueStorage>> {
00052
00053   using spi =
00054     ::trisycl::detail::shared_ptr_implementation<queue<Device, QueueStorage>,
00055                                                  detail::queue
00056                                                  <Device, QueueStorage>>;
00057
00058   // Allows the comparison operation to access the implementation
00059   friend spi;
00060
00061 public:
00062
00063   // Make the implementation member directly accessible in this class
00064   using spi::implementation;
00065
00066 /// The type of the scoped storage in the queue
00067   using storage_type = QueueStorage;
00068
00069 /// True if there is some storage in the queue
00070   static constexpr bool has_some_storage_p =
00071     not std::is_same_v<QueueStorage, empty_queue_scope>;
00072
00073 /// The device type targeted by this queue
00074   using device_type = Device;
00075
00076 /// The platform type targeted by this queue
00077   using platform_type = typename device_type::platform_type;
00078
00079 /// The queue scope storage type
00080   using queue_scope_type = storage_type;
00081
00082 /// The device scope storage type
```

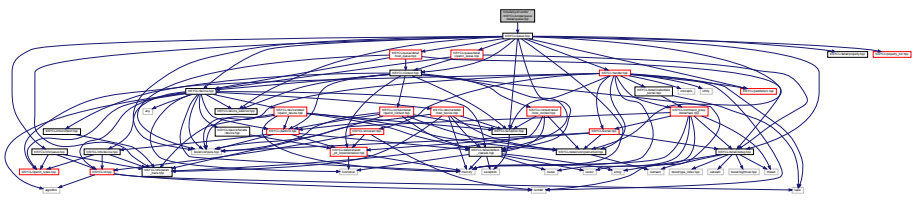
```
00083   using device_scope_type = detail::storage_type_trait_t<device_type>;
00084
00085 /// The platform scope storage type
00086   using platform_scope_type = detail::storage_type_trait_t<platform_type>;
00087
00088 /** Construct the queue with some queue-scoped storage on top
00089 of a SYCL queue
00090
00091 \param[in] q is the real queue to use
00092 \param[in] d is the a scoped device to use
00093 */
00094   queue(const ::trisycl::queue &q, const Device &d)
00095     :  spi { new detail::queue<Device, QueueStorage> { q, d } } {}
00096
00097
00098 /** Construct the queue with some queue-scoped storage on top
00099 of a scoped device
00100
00101 \param[in] d is the a scoped device to use
00102 */
00103   queue(const Device &d) :  queue { ::trisycl::queue { d }, d } {}
00104
00105
00106 /// It is still copyable
00107   queue(const queue&) = default;
00108
00109
00110 /// It is still movable
00111   queue(queue&&) = default;
00112
00113
00114 /// Access to the queue-scoped storage
00115   auto& get_storage()const {
00116     return implementation->get_storage();
00117   }
00118
00119
00120 /// Access to the underlying scoped device
00121   auto& get_device()const {
00122     return implementation->get_device();
00123   }
00124
00125
00126 /// Access to the underlying scoped platform
00127   auto& get_platform()const {
00128     return get_device().get_platform();
00129   }
00130
00131
00132 /// Provide access to the device scope from the host
00133   auto& device_scope() { return get_device().get_storage(); }
00134
00135 /// Provide access to the platform scope from the host, if any
00136   template<typename PS = platform_type,
00137            typename E = std::enable_if_t<PS::has_some_storage_p»
00138   auto& platform_scope() { return get_platform().get_storage(); }
00139
00140
00141 /** Add a conversion to \c trisycl::queue& so it can be used as
00142 a normal queue */
00143   operator ::trisycl::queue&() const {
00144     return implementation->get_underlying_queue();
00145   }
00146
00147 /** The command group redefining some member functions to be able to
00148 pass the scope and so on to the kernel */
00149   template <typename CGH>
00150   struct command_group  {
00151
00152 /// The plain SYCL command-group handler
00153     ::trisycl::handler &cgh;
00154     Device &d;
00155
00156     command_group(::trisycl::handler &cgh, Device &d)
00157       :  cgh { cgh }, d { d } {}
00158
00159
00160 /** Add a conversion to \c trisycl::handler& so the usual methods
00161 such as \c get_access can still work */
00162     operator ::trisycl::handler&() {
00163       return cgh;
00164     }
00165
00166
00167 /// Sequential kernel invocation
00168     template <typename KernelName, typename Kernel>
00169     void single_task(Kernel k) {
```

```
00170       cgh.single_task<KernelName>([=, *this] () mutable {
00171           k(*this);
00172         });
00173     }
00174
00175
00176 /// Parallel kernel invocation method on a range iteration space
00177     template <typename KernelName, int Rank, typename Kernel>
00178     void parallel_for(::trisycl::range<Rank> num_work_items,
00179                       Kernel k) {
00180       cgh.parallel_for<KernelName>(num_work_items,
00181                                    [=, *this] (::trisycl::id<Rank> i) mutable {
00182                                      k(i, *this);
00183                                    });
00184     }
00185
00186
00187 /// Provide access to the device scope from inside the kernel
00188     auto& device_scope() { return d.get_storage(); }
00189
00190 /** Provide access to the platform scope from inside the kernel,
00191 if any */
00192     template<typename PS = std::remove_reference_t<decltype(d.get_platform())>,
00193              typename E = std::enable_if_t<PS::has_some_storage_p>>
00194     auto& platform_scope() { return d.get_platform().get_storage(); }
00195   };
00196
00197
00198 /// Submit a command-group lambda function
00199   template <typename CG>
00200   void submit(CG cgh) {
00201     implementation->get_underlying_queue()
00202       .submit([&] (::trisycl::handler &cgh_generic) {
00203           command_group<CG> cg { cgh_generic, get_device() };
00204           cgh(cg);
00205         });
00206   }
00207
00208
00209 /** Performs a blocking wait for the completion all enqueued tasks in
00210 the queue */
00211   void wait() {
00212     implementation->get_underlying_queue().wait();
00213   }
00214
00215 };
00216
00217 /// @} to end the vendor_trisycl_scope Doxygen group
00218
00219 }
00220
00221
00222 /*
00223 # Some Emacs stuff:
00224 ### Local Variables:
00225 ### ispell-local-dictionary:  "american"
00226 ### eval:  (flyspell-prog-mode)
00227 ### End:
00228 */
00229
00230 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_QUEUE_HPP
```

13.347 include/sycl/vendor/triSYCL/scope/queue/detail/queue.hpp File Reference

`#include "triSYCL/queue.hpp"`

Include dependency graph for queue.hpp:



Classes

- class trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >

  A conceptual queue implementing some queue-scoped storage constructed from a device with some device-scope storage. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope
- namespace trisycl::vendor::trisycl::scope::detail

13.348 queue.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_QUEUE_DETAIL_QUEUE_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_QUEUE_DETAIL_QUEUE_HPP
00003
00004 /** \file Implementation details of the SYCL conceptual queue with a
00005 queue scope storage
00006
00007 \todo The device and queue are very similar.  Use a common
00008 mix-in to implement both of them?
00009
00010 Ronan at Keryell point FR
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include "triSYCL/queue.hpp"
00017
00018 namespace trisycl::vendor::trisycl::scope::detail {
00019
00020 /** \addtogroup vendor_trisycl_scope triSYCL extension for storage scopes
00021 @{
00022 */
00023
00024 /** A conceptual queue implementing some queue-scoped storage
00025 constructed from a device with some device-scope storage
00026 */
00027 template <typename Device,
00028           typename QueueStorage>
00029 class queue {
00030 /// The storage-less queue behind the scene
00031   ::trisycl::queue q;
00032
00033 /** The queue-scoped storage default-initialized
00034
00035 \todo For now it is allocated on the host in this CPU emulation
00036 but a device compiler and run time can create this on a real
00037 device
00038 */
00039   QueueStorage scope_storage;
00040
00041 /// The scoped device the scoped queue is built into
00042   Device device_with_scope;
00043
00044 public:
00045
00046 /** Construct the queue with some queue-scoped storage on top
00047 of a SYCL queue
00048
00049 \param[in] q is the real queue to use
00050 */
00051   queue(const ::trisycl::queue &q) :  q { q } {}
00052
00053
00054 /** Construct the queue with some queue-scoped storage on top
```

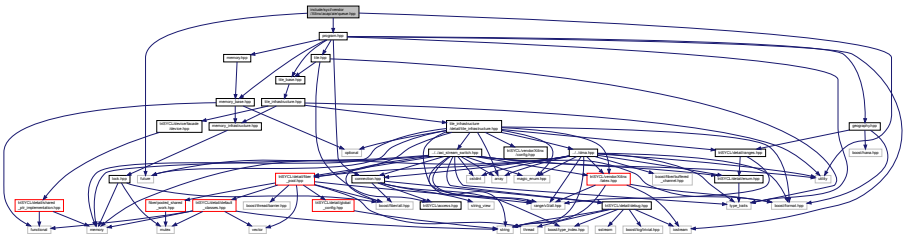
```
00055 of a SYCL queue
00056
00057 \param[in] q is the real queue to use
00058 \param[in] d is the a scoped device to use
00059 */
00060   queue(const ::trisycl::queue &q, const Device &d)
00061     :  q { q }, device_with_scope { d } {}
00062
00063
00064 /// Get the queue behind the curtain
00065   auto get_underlying_queue() {
00066     return q;
00067   }
00068
00069
00070 /// Access to the queue-scoped storage
00071   auto& get_storage() {
00072     return scope_storage;
00073   }
00074
00075
00076 /// Access to the underlying scoped device
00077   auto& get_device() {
00078     return device_with_scope;
00079   }
00080
00081 };
00082
00083 /// @} to end the vendor_trisycl_scope Doxygen group
00084
00085 }
00086
00087
00088 /*
00089 # Some Emacs stuff:
00090 ### Local Variables:
00091 ### ispell-local-dictionary:  "american"
00092 ### eval:  (flyspell-prog-mode)
00093 ### End:
00094 */
00095
00096 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_QUEUE_DETAIL_QUEUE_HPP
```

13.349 include/sycl/vendor/Xilinx/acap/aie/queue.hpp File Reference

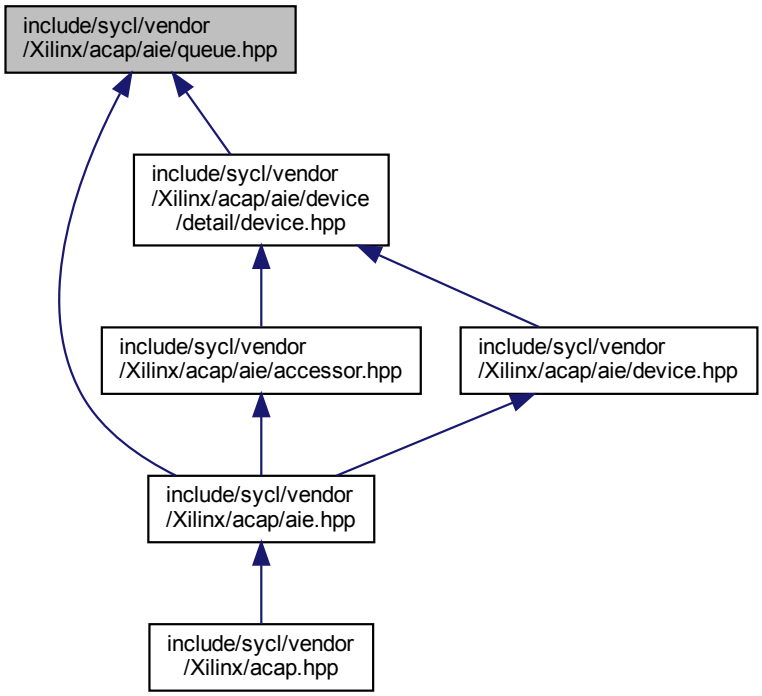
SYCL-friendly queue for AIE cooperative program including all the tiles.

```
#include <future>
#include <utility>
#include "program.hpp"
```

Include dependency graph for queue.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >

  The AI Engine minimal queue. More...
- struct trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter

  Provide a wait() function that actually can throw an exception instead of ignoring it by the std::future::wait()

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.349.1 Detailed Description

SYCL-friendly queue for AIE cooperative program including all the tiles.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file queue.hpp.

13.350 queue.hpp

Go to the documentation of this file.

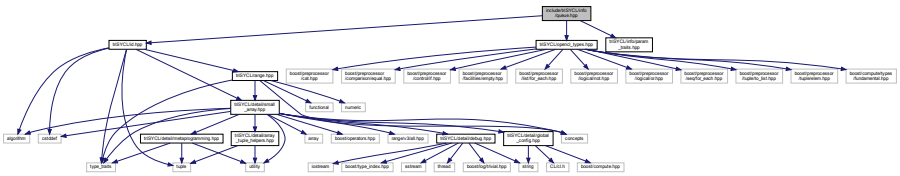
```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_QUEUE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_QUEUE_HPP
00003
00004 /** \file
00005
00006 SYCL-friendly queue for AIE cooperative program including all the tiles
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include <future>
00015 #include <utility>
00016
00017 #include "program.hpp"
00018
00019 namespace trisycl::vendor::xilinx::acap::aie {
00020
00021 /// \ingroup aie
00022 /// @{
00023
00024 /** The AI Engine minimal queue
00025
00026 \todo Make a real queue implementation instead of this mock-up
00027
00028 \param AIEDevice is the AIE device targeted by the queue
00029 */
00030 template <typename AIEDevice> struct queue {
00031   using device = AIEDevice;
00032   using geo = typename device::geo;
00033   using layout = typename geo::layout;
00034
00035 /// \todo Why using a reference since now a device is a shared_ptr?
00036   device& aie_d;
00037
00038   queue(device& d)
00039       :  aie_d { d } {}
00040
00041 /** Provide a wait() function that actually can throw an exception
00042 instead of ignoring it by the std::future::wait() */
00043   struct queue_waiter :  std::future<void> {
00044     void wait() {
00045       // Just call the function unwrapping the promise, it will
00046       // rethrow if there is any exception pending
00047       get();
00048     }
00049   };
00050
00051 /** Run synchronously a program execution on this queue
00052
00053 \param Tile is the description of the program tiles to
00054 instantiate.  By default each tile will run an empty program.
00055
00056 \param Memory is the description of the machine memory modules.  By
00057 default the machine has empty memory modules.
00058 */
00059   template <template <typename Device, int X, int Y>
00060             typename Tile = acap::aie::tile,
00061             template <typename Device, int X, int Y>
00062             typename Memory = acap::aie::memory>
00063   void run()const {
00064     program<device, Tile, Memory> { aie_d }.run();
00065   }
00066
00067 /** Run synchronously an heterogeneous invocable on this queue
00068
00069 \param f is an invocable taking an heterogeneous tile handler
00070 */
00071   template <typename Invocable> void run(Invocable&& f)const {
00072     program<device> { aie_d }.run(std::forward<Invocable>(f));
00073   }
00074
00075 /// Wait for all the device tiles on this queue to finish
00076   void wait()const {
00077     aie_d.for_each_tile([](auto& t) { t.wait(); });
00078   }
00079
00080 /** Run synchronously a uniform invocable on this queue
00081
00082 \param f is an invocable taking a uniform tile handler
```

```
00083 */
00084   template <typename Invocable> void uniform_run(Invocable&& f)const {
00085     aie_d.for_each_tile(
00086         [work = std::forward<Invocable>(f)](auto& t) {
00087           t.single_task(work);
00088         });
00089     wait();
00090   }
00091
00092 ///\todo SUBMIT
00093
00094 /** Submit a program execution on this queue
00095
00096 \param Tile is the description of the program tiles to
00097 instantiate.  By default each tile will run an empty program.
00098
00099 \param Memory is the description of the machine memory modules.  By
00100 default the machine has empty memory modules.
00101 */
00102   template <template <typename Device, int X, int Y>
00103             typename Tile = acap::aie::tile,
00104             template <typename Device, int X, int Y>
00105             typename Memory = acap::aie::memory>
00106   auto submit()const {
00107     return queue_waiter { std::async(std::launch::async,
00108 /** \fixme implement a real queue with
00109 lifetime support because here it assume
00110 the device will last...  */
00111                                      [*this] { this->run<Tile, Memory>(); }) };
00112   }
00113 };
00114
00115 /// @} End the aie Doxygen group
00116
00117 } // namespace trisycl::vendor::xilinx::acap::aie
00118
00119 /*
00120 # Some Emacs stuff:
00121 ### Local Variables:
00122 ### ispell-local-dictionary:  "american"
00123 ### eval:  (flyspell-prog-mode)
00124 ### End:
00125 */
00126
00127 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_QUEUE_HPP
```

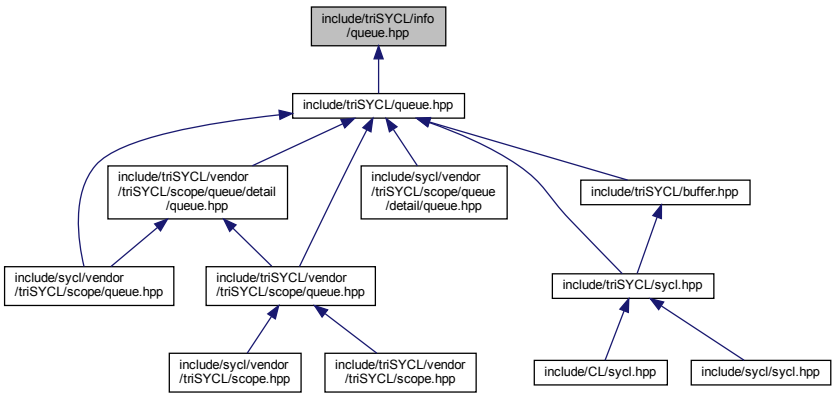
13.351 include/triSYCL/info/queue.hpp File Reference

```
#include "triSYCL/id.hpp"
#include "triSYCL/opencl_types.hpp"
#include "triSYCL/info/param_traits.hpp"
```

Include dependency graph for queue.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::info

Typedefs

- using trisycl::info::queue_profiling = bool

Enumerations

- enum class trisycl::info::queue : int { trisycl::info::context , trisycl::info::device , trisycl::info::reference_count }

  Queue information descriptors.

13.352 queue.hpp

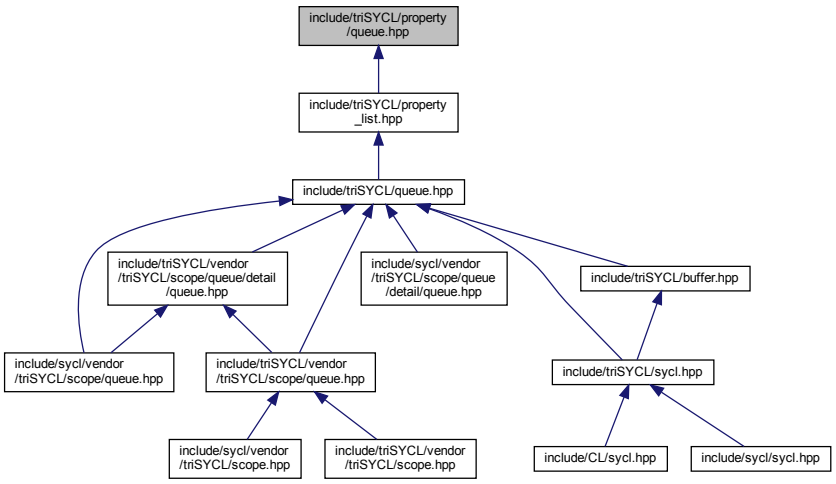
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_INFO_QUEUE_HPP
00002 #define TRISYCL_SYCL_INFO_QUEUE_HPP
00003
00004 /** \file The OpenCL SYCL queue information parameters
00005
00006 Dave Airlie
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include "triSYCL/id.hpp"
00013 #include "triSYCL/opencl_types.hpp"
00014 #include "triSYCL/info/param_traits.hpp"
00015
00016 namespace trisycl::info {
00017
00018 using queue_profiling = bool;
00019
00020 /** Queue information descriptors
00021 */
```

```
00022 enum class queue :  int {
00023   context,
00024   device,
00025   reference_count,
00026 };
00027
00028 TRISYCL_INFO_PARAM_TRAITS(queue::context, trisycl::context)
00029 TRISYCL_INFO_PARAM_TRAITS(queue::device, trisycl::device)
00030 TRISYCL_INFO_PARAM_TRAITS(queue::reference_count, trisycl::cl_uint)
00031
00032 }
00033 #endif
```

13.353 include/triSYCL/property/queue.hpp File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::property::queue::enable_profiling

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::property
- namespace trisycl::property::queue

13.354 queue.hpp

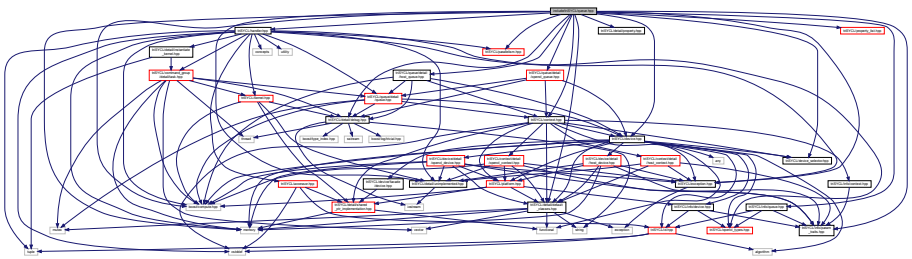
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_PROPERTY_QUEUE_HPP
00002 #define TRISYCL_SYCL_PROPERTY_QUEUE_HPP
00003
00004 /** \file Properties for queue objects.
00005
00006 Dave Airlie
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011 namespace trisycl::property::queue {
00012
00013 class enable_profiling :  public detail::property {
00014 public:
00015   enable_profiling() {}
00016 };
00017
00018 }
00019
00020 #endif // TRISYCL_SYCL_PROPERTY_QUEUE_HPP
```

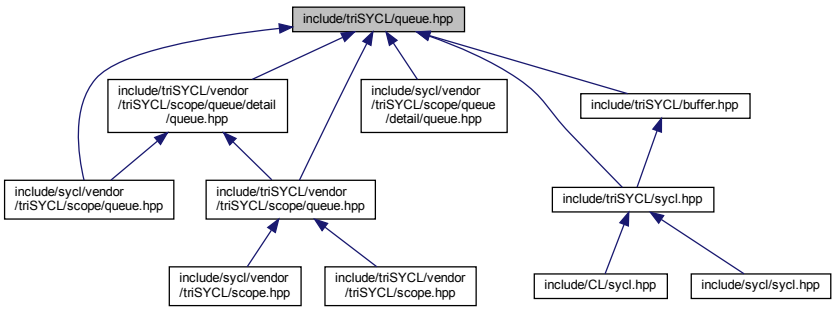
13.355 include/triSYCL/queue.hpp File Reference

```
#include <memory>
#include <boost/compute.hpp>
#include "triSYCL/context.hpp"
#include "triSYCL/detail/debug.hpp"
#include "triSYCL/detail/default_classes.hpp"
#include "triSYCL/detail/unimplemented.hpp"
#include "triSYCL/detail/property.hpp"
#include "triSYCL/device.hpp"
#include "triSYCL/device_selector.hpp"
#include "triSYCL/exception.hpp"
#include "triSYCL/handler.hpp"
#include "triSYCL/info/param_traits.hpp"
#include "triSYCL/info/queue.hpp"
#include "triSYCL/parallelism.hpp"
#include "triSYCL/property_list.hpp"
#include "triSYCL/queue/detail/host_queue.hpp"
#include "triSYCL/queue/detail/opencl_queue.hpp"
```

Include dependency graph for queue.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::queue

  SYCL queue, similar to the OpenCL queue concept. More...
- struct std::hash< trisycl::queue >

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace std

Functions

- template<> auto trisycl::queue::get_info< info::queue::device > () const
- template<> auto trisycl::queue::get_info< info::queue::context > () const
- template<> auto trisycl::queue::get_info< info::queue::reference_count > () const

13.356 queue.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_QUEUE_HPP
00002 #define TRISYCL_SYCL_QUEUE_HPP
00003
00004 /** \file The OpenCL SYCL queue
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <memory>
00013
00014 #ifdef TRISYCL_OPENCL
00015 #include <boost/compute.hpp>
00016 #endif
00017
00018 #include "triSYCL/context.hpp"
00019 #include "triSYCL/detail/debug.hpp"
00020 #include "triSYCL/detail/default_classes.hpp"
```

```
00021 #include "triSYCL/detail/unimplemented.hpp"
00022 #include "triSYCL/detail/property.hpp"
00023 #include "triSYCL/device.hpp"
00024 #include "triSYCL/device_selector.hpp"
00025 #include "triSYCL/exception.hpp"
00026 #include "triSYCL/handler.hpp"
00027 #include "triSYCL/info/param_traits.hpp"
00028 #include "triSYCL/info/queue.hpp"
00029 #include "triSYCL/parallelism.hpp"
00030 #include "triSYCL/property_list.hpp"
00031 #include "triSYCL/queue/detail/host_queue.hpp"
00032 #ifdef TRISYCL_OPENCL
00033 #include "triSYCL/queue/detail/opencl_queue.hpp"
00034 #endif
00035
00036 namespace trisycl {
00037
00038 class context;
00039 class device_selector;
00040
00041 /** \addtogroup execution Platforms, contexts, devices and queues
00042 @{
00043 */
00044
00045 /** SYCL queue, similar to the OpenCL queue concept.
00046
00047 \todo The implementation is quite minimal for now.  :-)
00048
00049 \todo All the queue methods should return a queue& instead of void
00050 to it is possible to chain opoerations
00051 */
00052 class queue
00053     /* Use the underlying queue implementation that can be shared in
00054 the SYCL model */
00055   :  public detail::shared_ptr_implementation<queue, detail::queue>,
00056     public property_list,
00057     detail::debug<queue>
00058 {
00059
00060   // The type encapsulating the implementation
00061   using implementation_t = typename queue::shared_ptr_implementation;
00062
00063   /* Allows the comparison operation to sneak in
00064
00065 Required from Clang++ 3.9 and G++ 6
00066 */
00067   friend implementation_t;
00068
00069 public:
00070
00071   // Make the implementation member directly accessible in this class
00072   using implementation_t::implementation;
00073
00074 /** Default constructor for queue from the default queue selector
00075
00076 Returns errors via the SYCL exception class.
00077 */
00078   explicit queue(const property_list &propList = {});
00079
00080 /** This constructor creates a SYCL queue from an OpenCL queue
00081
00082 At construction it does a retain on the queue memory object.
00083
00084 Retain a reference to the cl_command_queue object.  Caller should
00085 release the passed cl_command_queue object when it is no longer
00086 needed.
00087
00088 Return synchronous errors regarding the creation of the queue and
00089 report asynchronous errors via the async_handler callback function
00090 in conjunction with the synchronization and throw methods.
00091
00092 Note that the default case asyncHandler = nullptr is handled by the
00093 default constructor.
00094
00095 */
00096   queue(const async_handler &asyncHandler, const property_list &propList = {});
00097
00098 /** Creates a queue for the device provided by the device selector
00099
00100 If no device is selected, an error is reported.
00101
00102 Return synchronous errors regarding the creation of the queue and
00103 report asynchronous errors via the \c async_handler callback
00104 function if and only if there is an \c async_handler provided.
00105 */
00106   queue(const device_selector &deviceSelector,
00107         const property_list &propList = {})
```

```
00108     // Just create the queue from the selected device
00109     :  queue { device { deviceSelector }, nullptr, propList } { }
00110
00111 /** Creates a queue for the device provided by the device selector.
00112
00113 Return synchronous errors regarding the creation of the queue and
00114 report asynchronous errors via the \c async_handler callback
00115 function if and only if there is an \c async_handler provided.
00116 */
00117   queue(const device_selector &deviceSelector,
00118         const async_handler &asyncHandler,
00119         const property_list &propList = {})
00120     // Just create the queue from the selected device
00121     :  queue { device { deviceSelector }, asyncHandler, propList } { }
00122
00123 /** A queue is created for a SYCL device
00124
00125 Return asynchronous errors via the \c async_handler callback
00126 function.
00127 */
00128   queue(const device &d,
00129         const async_handler &asyncHandler,
00130         const property_list &propList = {}) :  implementation_t {
00131 #ifdef TRISYCL_OPENCL
00132     d.is_host()
00133       ?  std::shared_ptr<detail::queue>{ new detail::host_queue }
00134       :  detail::opencl_queue::instance(d)
00135 #else
00136     new detail::host_queue
00137 #endif
00138   }, property_list { propList } { }
00139
00140 /** A queue is created for a SYCL device
00141
00142 */
00143   queue(const device &d,
00144         const property_list &propList = {})
00145     :  queue { d, {}, propList } {}
00146
00147 /** This constructor chooses a device based on the provided
00148 device_selector, which needs to be in the given context.
00149
00150 If no device is selected, an error is reported.
00151
00152 Return synchronous errors regarding the creation of the queue.
00153
00154 If and only if there is an asyncHandler provided, it reports
00155 asynchronous errors via the async_handler callback function in
00156 conjunction with the synchronization and throw methods.
00157 */
00158   queue(const context &syclContext,
00159         const device_selector &deviceSelector,
00160         const async_handler &asyncHandler,
00161         const property_list &propList = {})
00162     :  property_list { propList } {
00163     device d = deviceSelector.select_device();
00164     vector_class<device> ctx_devs = syclContext.get_devices();
00165     if (std::find(ctx_devs.begin(), ctx_devs.end(), d) == ctx_devs.end())
00166       throw trisycl::invalid_object_error("Device doesn't belong to context\n");
00167     implementation =
00168 #ifdef TRISYCL_OPENCL
00169       d.is_host() ?  std::shared_ptr<detail::queue>{ new detail::host_queue }
00170     :  detail::opencl_queue::instance(d);
00171 #else
00172     std::shared_ptr<detail::queue>{ new detail::host_queue };
00173 #endif
00174   }
00175
00176 /** This constructor chooses a device based on the provided
00177 device_selector, which needs to be in the given context.
00178
00179 If no device is selected, an error is reported.
00180
00181 Return synchronous errors regarding the creation of the queue.
00182 */
00183   queue(const context &syclContext,
00184         const device_selector &deviceSelector,
00185         const property_list &propList = {})
00186     :  queue { syclContext, deviceSelector, {}, propList } {}
00187
00188 #ifdef TRISYCL_OPENCL
00189 /** This constructor creates a SYCL queue from an OpenCL queue
00190
00191 At construction it does a retain on the queue memory object.
00192
00193 Return synchronous errors regarding the creation of the queue.  If
00194 and only if there is an async_handler provided, it reports
```

```
00195 asynchronous errors via the async_handler callback function in
00196 conjunction with the synchronization and throw methods.
00197 */
00198   queue(const cl_command_queue &q, const context &syclContext,
00199         const async_handler &ah = {})
00200     :  queue { boost::compute::command_queue { q }, ah } {}
00201
00202
00203 /** Construct a queue instance using a boost::compute::command_queue
00204
00205 This is a triSYCL extension for boost::compute interoperation.
00206
00207 Return synchronous errors via the SYCL exception class.
00208
00209 \todo Deal with handler
00210 */
00211   queue(const boost::compute::command_queue &q, async_handler ah = nullptr)
00212     :  implementation_t { detail::opencl_queue::instance(q) }, property_list {} {}
00213 #endif
00214
00215
00216 #ifdef TRISYCL_OPENCL
00217 /** Return the underlying OpenCL command queue after doing a retain
00218
00219 This memory object is expected to be released by the developer.
00220
00221 Retain a reference to the returned cl_command_queue object.
00222
00223 Caller should release it when finished.
00224
00225 If the queue is a SYCL host queue then an exception is thrown.
00226 */
00227   cl_command_queue get()const {
00228     return implementation->get();
00229   }
00230
00231
00232 /** Return the underlying Boost.Compute command queue if it is an
00233 OpenCL queue
00234
00235 This is a triSYCL extension
00236 */
00237   boost::compute::command_queue get_boost_compute()const {
00238     return implementation->get_boost_compute();
00239   }
00240 #endif
00241
00242
00243 /** Return the SYCL queue's context
00244
00245 Report errors using SYCL exception classes.
00246 */
00247   context get_context()const {
00248     return implementation->get_context();
00249   }
00250
00251
00252 /** Return the SYCL device the queue is associated with
00253
00254 Report errors using SYCL exception classes.
00255 */
00256   device get_device()const {
00257     return implementation->get_device();
00258   }
00259
00260
00261 /// Return whether the queue is executing on a SYCL host device
00262   bool is_host()const {
00263     return implementation->is_host();
00264   }
00265
00266
00267 /** Performs a blocking wait for the completion all enqueued tasks in
00268 the queue
00269
00270 Synchronous errors will be reported through SYCL exceptions.
00271 */
00272   void wait() {
00273     implementation->wait_for_kernel_execution();
00274   }
00275
00276
00277 /** Perform a blocking wait for the completion all enqueued tasks in the queue
00278
00279 Synchronous errors will be reported via SYCL exceptions.
00280
00281 Asynchronous errors will be passed to the async_handler passed to the
```

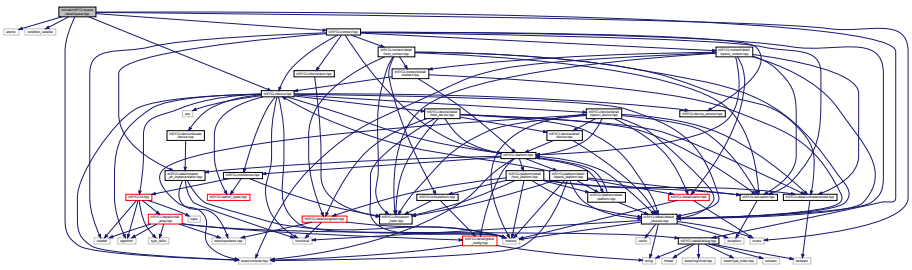
```
00282 queue on construction.
00283
00284 If no async_handler was provided then asynchronous exceptions will
00285 be lost.
00286 */
00287   void wait_and_throw() {
00288     // \todo Implement the throw part of wait_and_throw
00289     wait();
00290     TRISYCL_UNIMPL;
00291   }
00292
00293
00294 /** Checks to see if any asynchronous errors have been produced by the
00295 queue and if so reports them by passing them to the async_handler
00296 passed to the queue on construction
00297
00298 If no async_handler was provided then asynchronous exceptions will
00299 be lost.
00300 */
00301   void throw_asynchronous() {
00302     TRISYCL_UNIMPL;
00303   }
00304
00305
00306 /// Queries the platform for cl_command_queue info
00307   template <info::queue param>
00308   inline auto get_info() const;
00309
00310 /** Submit a command group functor to the queue, in order to be
00311 scheduled for execution on the device
00312
00313 Use an explicit functor parameter taking a handler& so we can use
00314 "auto" in submit() lambda parameter.
00315
00316 \todo Add in the spec an implicit conversion of event to
00317 queue& so it is possible to chain operations on the queue
00318 */
00319   template <typename Handler_Functor>
00320   event submit(Handler_Functor cgf) {
00321     handler command_group_handler { implementation };
00322     cgf(command_group_handler);
00323     return {};
00324   }
00325
00326
00327 /** Submit a command group functor to the queue, in order to be
00328 scheduled for execution on the device
00329
00330 On kernel error, this command group functor, then it is scheduled
00331 for execution on the secondary queue.
00332
00333 Return a command group functor event, which is corresponds to the
00334 queue the command group functor is being enqueued on.
00335 */
00336   template <typename Handler_Functor>
00337   event submit(Handler_Functor cgf, queue &secondaryQueue) {
00338     TRISYCL_UNIMPL;
00339     // Since it is not implemented, always submit on the main queue
00340     return submit(cgf);
00341   }
00342
00343 /** Check if the queue was constructed with the specified
00344 property.
00345 */
00346   template <typename propertyT>
00347   bool has_property()const {
00348     return property_list::has_property<propertyT>();
00349   }
00350
00351 /** Return a copy of the property that the queue was
00352 constructed with.
00353 */
00354   template <typename propertyT>
00355   propertyT get_property()const {
00356     return property_list::get_property<propertyT>();
00357   }
00358 };
00359
00360 template<>
00361 inline auto queue::get_info<info::queue::device>()const {
00362   return implementation->get_device();
00363 }
00364
00365 template<>
00366 inline auto queue::get_info<info::queue::context>()const {
00367   return implementation->get_context();
00368 }
```

```
00369
00370 template<>
00371 inline auto queue::get_info<info::queue::reference_count>()const {
00372   return trisycl::cl_uint {0};
00373 }
00374 /// @} to end the execution Doxygen group
00375
00376 }
00377
00378 /* Inject a custom specialization of std::hash to have the buffer
00379 usable into an unordered associative container
00380
00381 \todo Add this to the spec
00382 */
00383 namespace std {
00384
00385 template <> struct hash<trisycl::queue> {
00386
00387   auto operator()(const trisycl::queue &q)const {
00388     // Forward the hashing to the implementation
00389     return q.hash();
00390   }
00391
00392 };
00393
00394 }
00395
00396 /*
00397 # Some Emacs stuff:
00398 ### Local Variables:
00399 ### ispell-local-dictionary:  "american"
00400 ### eval:  (flyspell-prog-mode)
00401 ### End:
00402 */
00403
00404 #endif // TRISYCL_SYCL_QUEUE_HPP
```

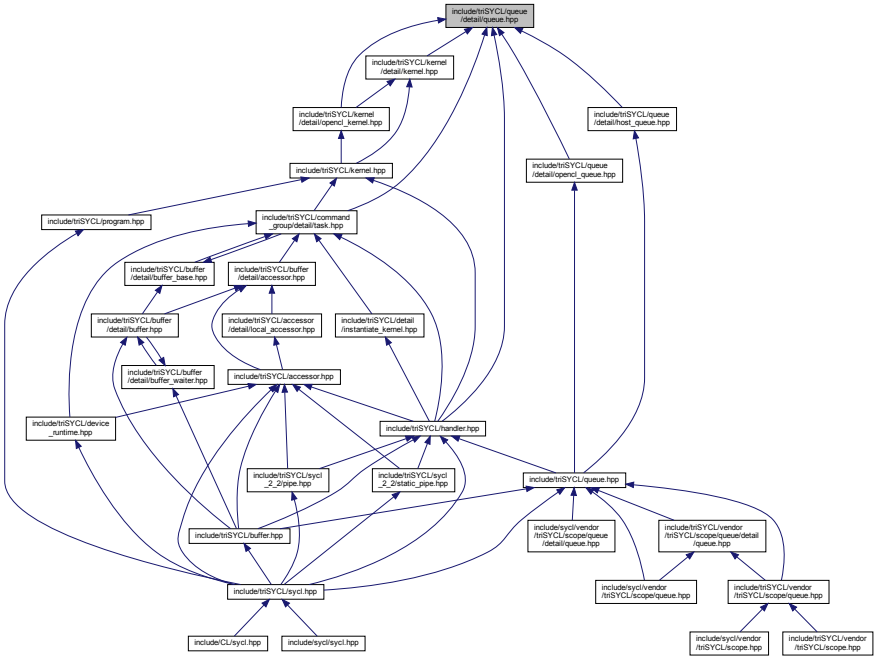
13.357 include/triSYCL/queue/detail/queue.hpp File Reference

```
#include <atomic>
#include <condition_variable>
#include <mutex>
#include <boost/compute.hpp>
#include "triSYCL/context.hpp"
#include "triSYCL/device.hpp"
#include "triSYCL/detail/debug.hpp"
```

Include dependency graph for queue.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::detail::queue

  Some implementation details about the SYCL queue.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

13.358 queue.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_QUEUE_DETAIL_QUEUE_HPP
00002 #define TRISYCL_SYCL_QUEUE_DETAIL_QUEUE_HPP
00003
00004 /** \file Some implementation details of queue.
00005
00006 Ronan at Keryell point FR
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 #include <atomic>
00013 #include <condition_variable>
00014 #include <mutex>
00015
00016 #ifdef TRISYCL_OPENCL
```

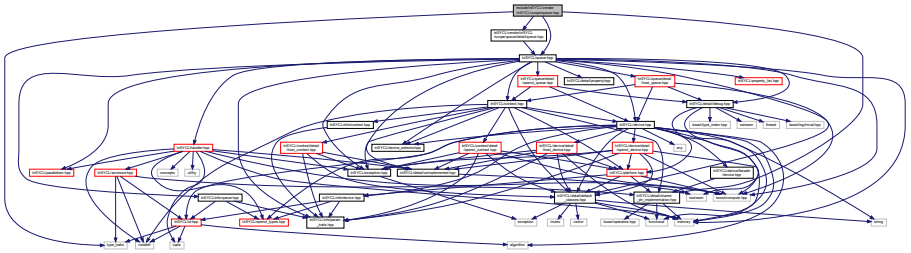
```
00017 #include <boost/compute.hpp>
00018 #endif
00019
00020 #include "triSYCL/context.hpp"
00021 #include "triSYCL/device.hpp"
00022 #include "triSYCL/detail/debug.hpp"
00023
00024 namespace trisycl::detail {
00025
00026 /** Some implementation details about the SYCL queue
00027 */
00028 struct queue :  detail::debug<detail::queue> {
00029 /// Track the number of kernels still running to wait for their completion
00030   std::atomic<size_t> running_kernels;
00031
00032 /// To signal when all the kernels have completed
00033   std::condition_variable finished;
00034 /// To protect the access to the condition variable
00035   std::mutex finished_mutex;
00036
00037
00038 /// Initialize the queue with 0 running kernel
00039   queue() :  running_kernels { 0 } {}
00040
00041 /// Wait for all kernel completion
00042   void wait_for_kernel_execution() {
00043     TRISYCL_DUMP_T("Queue waiting for kernel completion");
00044     std::unique_lock<std::mutex> ul { finished_mutex };
00045     finished.wait(ul, [&] {
00046         // When there is no kernel running in this queue, we are ready to go
00047         return running_kernels == 0;
00048       });
00049   }
00050
00051
00052 /// Signal that a new kernel started on this queue
00053   void kernel_start() {
00054     TRISYCL_DUMP_T("A kernel has been added to the queue");
00055     // One more kernel
00056     ++running_kernels;
00057   }
00058
00059
00060 /// Signal that a new kernel finished on this queue
00061   void kernel_end() {
00062     TRISYCL_DUMP_T("A kernel of the queue ended");
00063     std::unique_lock<std::mutex> ul { finished_mutex };
00064     if (--running_kernels == 0) {
00065       // Micro-optimization:  unlock before the notification
00066       // https://en.cppreference.com/w/cpp/thread/condition_variable/notify_all
00067       ul.unlock();
00068       /* It was the last kernel running, so signal the queue just in
00069 case it was working for it for completion
00070
00071 In some cases several threads might want to wait for the
00072 same queue, because of this \c notify_one is not be enough
00073 and a \c notify_all is needed
00074 */
00075       finished.notify_all();
00076     }
00077   }
00078
00079
00080 #ifdef TRISYCL_OPENCL
00081 /** Return the underlying OpenCL command queue after doing a retain
00082
00083 This memory object is expected to be released by the developer.
00084
00085 Retain a reference to the returned cl_command_queue object.
00086
00087 Caller should release it when finished.
00088
00089 If the queue is a SYCL host queue then an exception is thrown.
00090 */
00091   virtual cl_command_queue get() const = 0;
00092
00093 /// Return the underlying Boost.Compute command queue
00094   virtual boost::compute::command_queue &get_boost_compute() = 0;
00095 #endif
00096
00097
00098 /** Return the SYCL queue's context
00099
00100 Report errors using SYCL exception classes.
00101 */
00102   virtual trisycl::context get_context() const = 0;
00103
```

```
00104
00105 /** Return the SYCL device the queue is associated with
00106
00107 Report errors using SYCL exception classes.
00108 */
00109   virtual trisycl::device get_device() const = 0;
00110
00111
00112 /// Return whether the queue is executing on a SYCL host device
00113   virtual bool is_host() const = 0;
00114
00115
00116 /// Wait for all kernel completion before the queue destruction
00117 /// \todo Update according spec since queue destruction is non blocking
00118   virtual ~queue() {
00119     wait_for_kernel_execution();
00120   }
00121
00122 };
00123
00124 }
00125
00126 /*
00127 # Some Emacs stuff:
00128 ### Local Variables:
00129 ### ispell-local-dictionary:  "american"
00130 ### eval:  (flyspell-prog-mode)
00131 ### End:
00132 */
00133
00134 #endif // TRISYCL_SYCL_QUEUE_DETAIL_QUEUE_HPP
```

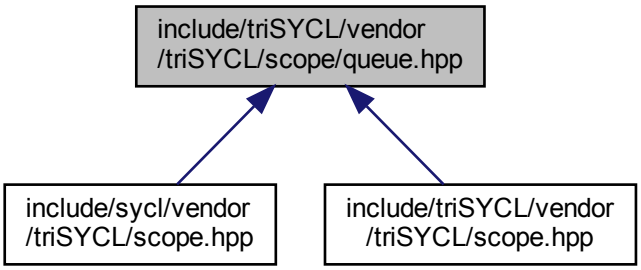
13.359 include/triSYCL/vendor/triSYCL/scope/queue.hpp File Reference

```
#include <type_traits>
#include "triSYCL/detail/shared_ptr_implementation.hpp"
#include "triSYCL/queue.hpp"
#include "triSYCL/vendor/triSYCL/scope/queue/detail/queue.hpp"
```

Include dependency graph for queue.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::trisycl::scope::empty_queue_scope

  An empty queue scope to make compilation errors clearer. More...
- class trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >

  A conceptual queue implementing some queue-scoped storage constructed from a device with some device-scope storage. More...
- struct trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >

  The command group redefining some member functions to be able to pass the scope and so on to the kernel.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope

Macros

- #define SYCL_VENDOR_TRISYCL_QUEUE_SCOPE 1

  This is an extension providing scope storage for queues.

13.359.1 Macro Definition Documentation

13.359.1.1 SYCL_VENDOR_TRISYCL_QUEUE_SCOPE

```
#define SYCL_VENDOR_TRISYCL_QUEUE_SCOPE 1
```

This is an extension providing scope storage for queues.

Definition at line 27 of file queue.hpp.

13.360 queue.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_QUEUE_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_QUEUE_HPP
00003
00004 /** \file The SYCL conceptual queue with a queue scope storage
00005
00006
00007 \todo Add some caching or prevent multiple creation of storage on
00008 the queue ?  Or is this a feature to have different scopes on
00009 the same queue?
00010
00011 \todo The device and queue are very similar.  Use a common
00012 mix-in to implement both of them?
00013
00014 Ronan at Keryell point FR
00015
00016 This file is distributed under the University of Illinois Open Source
00017 License.  See LICENSE.TXT for details.
00018 */
00019
00020 #include <type_traits>
00021
00022 #include "triSYCL/detail/shared_ptr_implementation.hpp"
00023 #include "triSYCL/queue.hpp"
00024 #include "triSYCL/vendor/triSYCL/scope/queue/detail/queue.hpp"
00025
00026 /// This is an extension providing scope storage for queues
00027 #define SYCL_VENDOR_TRISYCL_QUEUE_SCOPE 1
00028
00029 namespace trisycl::vendor::trisycl::scope {
00030
00031 /** \addtogroup vendor_trisycl_scope triSYCL extension for storage scopes
00032 @{
00033 */
00034
00035 /// An empty queue scope to make compilation errors clearer
00036 struct empty_queue_scope {
00037 };
00038
00039
00040 /** A conceptual queue implementing some queue-scoped storage
00041 constructed from a device with some device-scope storage
00042 */
00043 template <typename Device,
00044           typename QueueStorage = empty_queue_scope>
00045 class queue
00046   /* Use the underlying queue implementation that can be shared in the
00047 SYCL model */
00048   :  public
00049     ::trisycl::detail::shared_ptr_implementation<queue<Device, QueueStorage>,
00050                                                  detail::queue
00051                                                  <Device, QueueStorage>> {
00052
00053   using spi =
00054     ::trisycl::detail::shared_ptr_implementation<queue<Device, QueueStorage>,
00055                                                  detail::queue
00056                                                  <Device, QueueStorage>>;
00057
00058   // Allows the comparison operation to access the implementation
00059   friend spi;
00060
00061 public:
00062
00063   // Make the implementation member directly accessible in this class
00064   using spi::implementation;
00065
00066 /// The type of the scoped storage in the queue
00067   using storage_type = QueueStorage;
00068
00069 /// True if there is some storage in the queue
00070   static constexpr bool has_some_storage_p =
00071     not std::is_same_v<QueueStorage, empty_queue_scope>;
00072
00073 /// The device type targeted by this queue
00074   using device_type = Device;
00075
00076 /// The platform type targeted by this queue
00077   using platform_type = typename device_type::platform_type;
00078
00079 /// The queue scope storage type
00080   using queue_scope_type = storage_type;
00081
00082 /// The device scope storage type
```

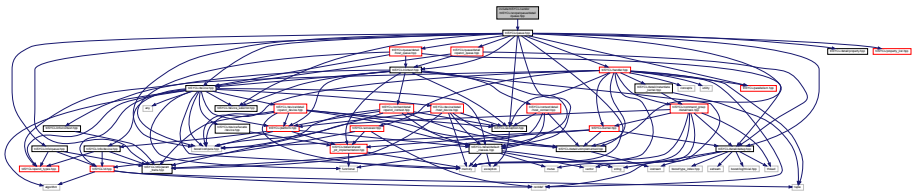
```
00083   using device_scope_type = detail::storage_type_trait_t<device_type>;
00084
00085 /// The platform scope storage type
00086   using platform_scope_type = detail::storage_type_trait_t<platform_type>;
00087
00088 /** Construct the queue with some queue-scoped storage on top
00089 of a SYCL queue
00090
00091 \param[in] q is the real queue to use
00092 \param[in] d is the a scoped device to use
00093 */
00094   queue(const ::trisycl::queue &q, const Device &d)
00095     :  spi { new detail::queue<Device, QueueStorage> { q, d } } {}
00096
00097
00098 /** Construct the queue with some queue-scoped storage on top
00099 of a scoped device
00100
00101 \param[in] d is the a scoped device to use
00102 */
00103   queue(const Device &d) :  queue { ::trisycl::queue { d }, d } {}
00104
00105
00106 /// It is still copyable
00107   queue(const queue&) = default;
00108
00109
00110 /// It is still movable
00111   queue(queue&&) = default;
00112
00113
00114 /// Access to the queue-scoped storage
00115   auto& get_storage()const {
00116     return implementation->get_storage();
00117   }
00118
00119
00120 /// Access to the underlying scoped device
00121   auto& get_device()const {
00122     return implementation->get_device();
00123   }
00124
00125
00126 /// Access to the underlying scoped platform
00127   auto& get_platform()const {
00128     return get_device().get_platform();
00129   }
00130
00131
00132 /// Provide access to the device scope from the host
00133   auto& device_scope() { return get_device().get_storage(); }
00134
00135 /// Provide access to the platform scope from the host, if any
00136   template<typename PS = platform_type,
00137            typename E = std::enable_if_t<PS::has_some_storage_p»
00138   auto& platform_scope() { return get_platform().get_storage(); }
00139
00140
00141 /** Add a conversion to \c trisycl::queue& so it can be used as
00142 a normal queue */
00143   operator ::trisycl::queue&() const {
00144     return implementation->get_underlying_queue();
00145   }
00146
00147 /** The command group redefining some member functions to be able to
00148 pass the scope and so on to the kernel */
00149   template <typename CGH>
00150   struct command_group  {
00151
00152 /// The plain SYCL command-group handler
00153     ::trisycl::handler &cgh;
00154     Device &d;
00155
00156     command_group(::trisycl::handler &cgh, Device &d)
00157       :  cgh { cgh }, d { d } {}
00158
00159
00160 /** Add a conversion to \c trisycl::handler& so the usual methods
00161 such as \c get_access can still work */
00162     operator ::trisycl::handler&() {
00163       return cgh;
00164     }
00165
00166
00167 /// Sequential kernel invocation
00168     template <typename KernelName, typename Kernel>
00169     void single_task(Kernel k) {
```

```
00170       cgh.single_task<KernelName>([=, *this] () mutable {
00171           k(*this);
00172         });
00173     }
00174
00175
00176 /// Parallel kernel invocation method on a range iteration space
00177     template <typename KernelName, int Rank, typename Kernel>
00178     void parallel_for(::trisycl::range<Rank> num_work_items,
00179                       Kernel k) {
00180       cgh.parallel_for<KernelName>(num_work_items,
00181                                    [=, *this] (::trisycl::id<Rank> i) mutable {
00182                                      k(i, *this);
00183                                    });
00184     }
00185
00186
00187 /// Provide access to the device scope from inside the kernel
00188     auto& device_scope() { return d.get_storage(); }
00189
00190 /** Provide access to the platform scope from inside the kernel,
00191 if any */
00192     template<typename PS = std::remove_reference_t<decltype(d.get_platform())>,
00193              typename E = std::enable_if_t<PS::has_some_storage_p»
00194     auto& platform_scope() { return d.get_platform().get_storage(); }
00195   };
00196
00197
00198 /// Submit a command-group lambda function
00199   template <typename CG>
00200   void submit(CG cgh) {
00201     implementation->get_underlying_queue()
00202       .submit([&] (::trisycl::handler &cgh_generic) {
00203           command_group<CG> cg { cgh_generic, get_device() };
00204           cgh(cg);
00205         });
00206   }
00207
00208
00209 /** Performs a blocking wait for the completion all enqueued tasks in
00210 the queue */
00211   void wait() {
00212     implementation->get_underlying_queue().wait();
00213   }
00214
00215 };
00216
00217 /// @} to end the vendor_trisycl_scope Doxygen group
00218
00219 }
00220
00221
00222 /*
00223 # Some Emacs stuff:
00224 ### Local Variables:
00225 ### ispell-local-dictionary:  "american"
00226 ### eval:  (flyspell-prog-mode)
00227 ### End:
00228 */
00229
00230 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_QUEUE_HPP
```

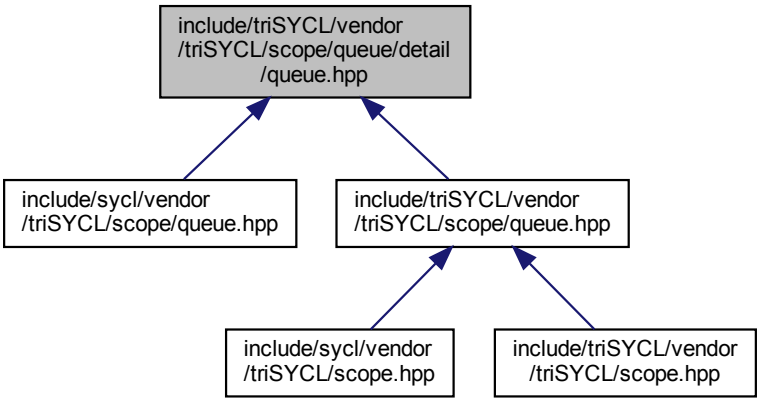
13.361 include/triSYCL/vendor/triSYCL/scope/queue/detail/queue.hpp File Reference

`#include "triSYCL/queue.hpp"`

Include dependency graph for queue.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >

  A conceptual queue implementing some queue-scoped storage constructed from a device with some device-scope storage. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::trisycl
- namespace trisycl::vendor::trisycl::scope
- namespace trisycl::vendor::trisycl::scope::detail

13.362 queue.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_QUEUE_DETAIL_QUEUE_HPP
00002 #define TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_QUEUE_DETAIL_QUEUE_HPP
00003
00004 /** \file Implementation details of the SYCL conceptual queue with a
00005 queue scope storage
00006
00007 \todo The device and queue are very similar.  Use a common
00008 mix-in to implement both of them?
00009
00010 Ronan at Keryell point FR
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include "triSYCL/queue.hpp"
00017
00018 namespace trisycl::vendor::trisycl::scope::detail {
00019
```

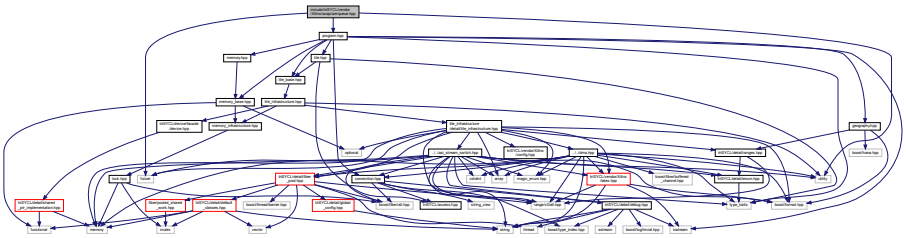
```
00020 /** \addtogroup vendor_trisycl_scope triSYCL extension for storage scopes
00021 @{
00022 */
00023
00024 /** A conceptual queue implementing some queue-scoped storage
00025 constructed from a device with some device-scope storage
00026 */
00027 template <typename Device,
00028           typename QueueStorage>
00029 class queue {
00030 /// The storage-less queue behind the scene
00031   ::trisycl::queue q;
00032
00033 /** The queue-scoped storage default-initialized
00034
00035 \todo For now it is allocated on the host in this CPU emulation
00036 but a device compiler and run time can create this on a real
00037 device
00038 */
00039   QueueStorage scope_storage;
00040
00041 /// The scoped device the scoped queue is built into
00042   Device device_with_scope;
00043
00044 public:
00045
00046 /** Construct the queue with some queue-scoped storage on top
00047 of a SYCL queue
00048
00049 \param[in] q is the real queue to use
00050 */
00051   queue(const ::trisycl::queue &q) :  q { q } {}
00052
00053
00054 /** Construct the queue with some queue-scoped storage on top
00055 of a SYCL queue
00056
00057 \param[in] q is the real queue to use
00058 \param[in] d is the a scoped device to use
00059 */
00060   queue(const ::trisycl::queue &q, const Device &d)
00061     :  q { q }, device_with_scope { d } {}
00062
00063
00064 /// Get the queue behind the curtain
00065   auto get_underlying_queue() {
00066     return q;
00067   }
00068
00069
00070 /// Access to the queue-scoped storage
00071   auto& get_storage() {
00072     return scope_storage;
00073   }
00074
00075
00076 /// Access to the underlying scoped device
00077   auto& get_device() {
00078     return device_with_scope;
00079   }
00080
00081 };
00082
00083 /// @} to end the vendor_trisycl_scope Doxygen group
00084
00085 }
00086
00087
00088 /*
00089 # Some Emacs stuff:
00090 ### Local Variables:
00091 ### ispell-local-dictionary:  "american"
00092 ### eval:  (flyspell-prog-mode)
00093 ### End:
00094 */
00095
00096 #endif // TRISYCL_SYCL_VENDOR_TRISYCL_SCOPE_QUEUE_DETAIL_QUEUE_HPP
```

13.363 include/triSYCL/vendor/Xilinx/acap/aie/queue.hpp File Reference

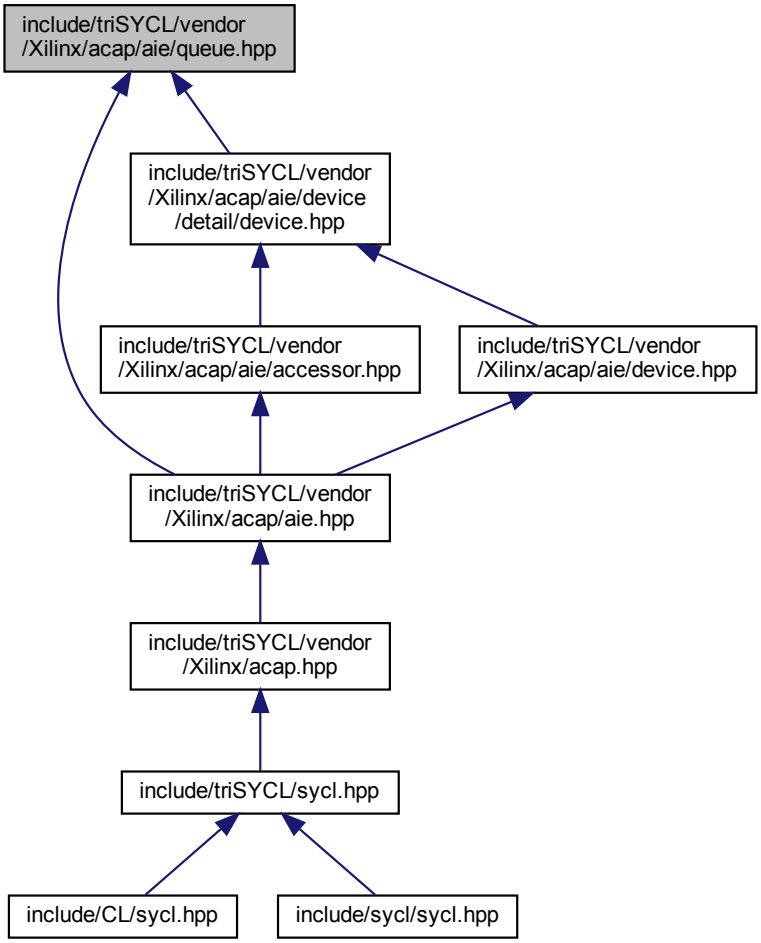
SYCL-friendly queue for AIE cooperative program including all the tiles.

```
#include <future>
#include <utility>
#include "program.hpp"
```

Include dependency graph for queue.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >

The AI Engine minimal queue. More...

- struct trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter

  Provide a wait() function that actually can throw an exception instead of ignoring it by the std::future::wait()

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.363.1 Detailed Description

SYCL-friendly queue for AIE cooperative program including all the tiles.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file queue.hpp.

13.364 queue.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_QUEUE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_QUEUE_HPP
00003
00004 /** \file
00005
00006 SYCL-friendly queue for AIE cooperative program including all the tiles
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include <future>
00015 #include <utility>
00016
00017 #include "program.hpp"
00018
00019 namespace trisycl::vendor::xilinx::acap::aie {
00020
00021 /// \ingroup aie
00022 /// @{
00023
00024 /** The AI Engine minimal queue
00025
00026 \todo Make a real queue implementation instead of this mock-up
00027
00028 \param AIEDevice is the AIE device targeted by the queue
00029 */
00030 template <typename AIEDevice> struct queue {
00031   using device = AIEDevice;
00032   using geo = typename device::geo;
00033   using layout = typename geo::layout;
00034
00035 /// \todo Why using a reference since now a device is a shared_ptr?
00036   device& aie_d;
00037
00038   queue(device& d)
00039       :  aie_d { d } {}
```

```
00040
00041 /** Provide a wait() function that actually can throw an exception
00042 instead of ignoring it by the std::future::wait() */
00043   struct queue_waiter :  std::future<void> {
00044     void wait() {
00045       // Just call the function unwrapping the promise, it will
00046       // rethrow if there is any exception pending
00047       get();
00048     }
00049   };
00050
00051 /** Run synchronously a program execution on this queue
00052
00053 \param Tile is the description of the program tiles to
00054 instantiate.  By default each tile will run an empty program.
00055
00056 \param Memory is the description of the machine memory modules.  By
00057 default the machine has empty memory modules.
00058 */
00059   template <template <typename Device, int X, int Y>
00060             typename Tile = acap::aie::tile,
00061             template <typename Device, int X, int Y>
00062             typename Memory = acap::aie::memory>
00063   void run()const {
00064     program<device, Tile, Memory> { aie_d }.run();
00065   }
00066
00067 /** Run synchronously an heterogeneous invocable on this queue
00068
00069 \param f is an invocable taking an heterogeneous tile handler
00070 */
00071   template <typename Invocable> void run(Invocable&& f)const {
00072     program<device> { aie_d }.run(std::forward<Invocable>(f));
00073   }
00074
00075 /// Wait for all the device tiles on this queue to finish
00076   void wait()const {
00077     aie_d.for_each_tile([](auto& t) { t.wait(); });
00078   }
00079
00080 /** Run synchronously a uniform invocable on this queue
00081
00082 \param f is an invocable taking a uniform tile handler
00083 */
00084   template <typename Invocable> void uniform_run(Invocable&& f)const {
00085     aie_d.for_each_tile(
00086         [work = std::forward<Invocable>(f)](auto& t) {
00087           t.single_task(work);
00088         });
00089     wait();
00090   }
00091
00092 ///\todo SUBMIT
00093
00094 /** Submit a program execution on this queue
00095
00096 \param Tile is the description of the program tiles to
00097 instantiate.  By default each tile will run an empty program.
00098
00099 \param Memory is the description of the machine memory modules.  By
00100 default the machine has empty memory modules.
00101 */
00102   template <template <typename Device, int X, int Y>
00103             typename Tile = acap::aie::tile,
00104             template <typename Device, int X, int Y>
00105             typename Memory = acap::aie::memory>
00106   auto submit()const {
00107     return queue_waiter { std::async(std::launch::async,
00108 /** \fixme implement a real queue with
00109 lifetime support because here it assume
00110 the device will last...  */
00111                                      [*this] { this->run<Tile, Memory>(); }) };
00112   }
00113 };
00114
00115 /// @} End the aie Doxygen group
00116
00117 } // namespace trisycl::vendor::xilinx::acap::aie
00118
00119 /*
00120 # Some Emacs stuff:
00121 ### Local Variables:
00122 ### ispell-local-dictionary:  "american"
00123 ### eval:  (flyspell-prog-mode)
00124 ### End:
00125 */
00126
```

```
00127 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_QUEUE_HPP
```

13.365 include/sycl/vendor/Xilinx/acap/aie/shim_tile.hpp File Reference

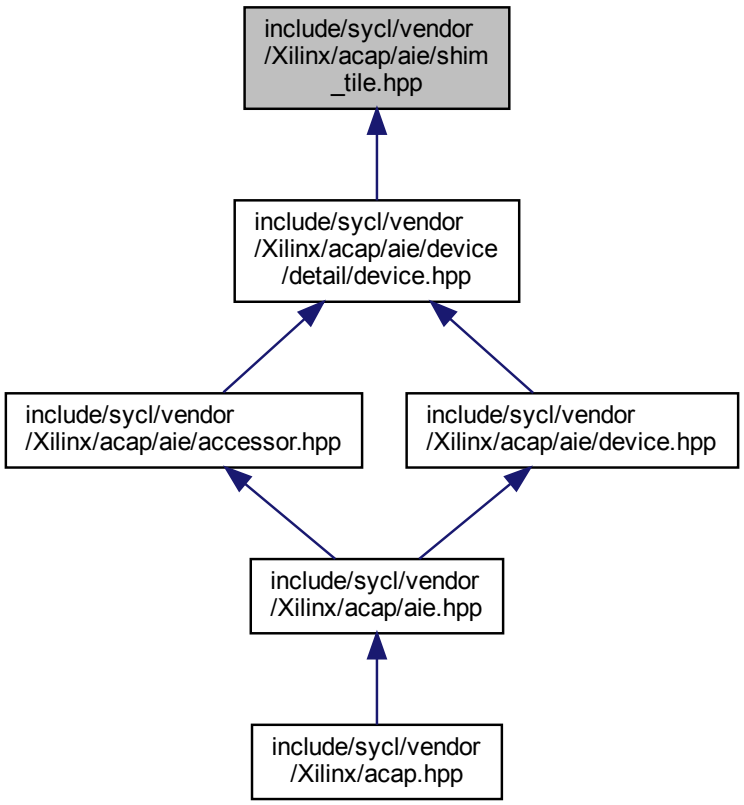
Model of a shim tile between the AI array and the rest of the circuit.

```
#include "axi_stream_switch.hpp"
```

Include dependency graph for shim_tile.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >

  The AI Engine shim tile infrastructure. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.365.1 Detailed Description

Model of a shim tile between the AI array and the rest of the circuit.

Based on Math Engine (ME) Architecture Specification Revision v2.01 March 2019, section "6 ME Shim Architecture".

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file shim_tile.hpp.

13.366 shim_tile.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_SHIM_TILE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_SHIM_TILE_HPP
00003
00004 /** \file
00005
00006 Model of a shim tile between the AI array and the rest of the circuit
00007
00008 Based on Math Engine (ME) Architecture Specification Revision
00009 v2.01 March 2019, section "6 ME Shim Architecture".
00010
00011 Ronan dot Keryell at Xilinx dot com
00012
00013 This file is distributed under the University of Illinois Open Source
00014 License.  See LICENSE.TXT for details.
00015 */
00016
00017 #include "axi_stream_switch.hpp"
00018
00019 namespace trisycl::vendor::xilinx::acap::aie {
00020
00021 /// \ingroup aie
00022 /// @{
00023
00024 /** The AI Engine shim tile infrastructure
00025
00026 \param AIE is the type representing the full CGRA with the
00027 programs and memory contents
00028
00029 \todo Assume only AIE Shim PL tiles for now
00030 */
00031 template <typename AIEDevice>
00032 class shim_tile {
00033   using geo = typename AIEDevice::geo;
00034   using axi_ss_geo = typename geo::shim_axi_stream_switch;
00035   using mpl = typename axi_ss_geo::master_port_layout;
00036   using spl = typename axi_ss_geo::slave_port_layout;
00037   using axi_ss_t = axi_stream_switch<axi_ss_geo>;
00038
00039 /// Keep the horizontal coordinate for debug purpose
00040   int x_coordinate;
00041
00042 /// \todo Remove?
00043 /// Keep the vertical coordinate for debug purpose
```

```
00044   int y_coordinate = -1;
00045
00046 /// The AXI stream switch of the core tile
00047   axi_ss_t axi_ss;
00048
00049 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00050 /// Keep track of the fiber executor
00051   ::trisycl::detail::fiber_pool *fe;
00052
00053 /// To shepherd the working fiber
00054   ::trisycl::detail::fiber_pool::future<void> future_work;
00055 #else
00056 /// Keep track of the std::thread execution in this tile
00057   std::future<void> future_work;
00058 #endif
00059
00060 public:
00061
00062 /// Construct the shim tile infrastructure
00063   shim_tile() {
00064     // Connect the core receivers to its AXI stream switch
00065     for (auto p :  axi_ss_geo::m_south_range)
00066       output(p) = std::make_shared<port_receiver<axi_ss_t>>
00067         (axi_ss, "BLI_shim_receiver");
00068   }
00069
00070
00071 /** Start the shim tile infrastructure associated to the AIE device
00072
00073 \param[in] x is the horizontal coordinate for this tile
00074
00075 \param[in] fiber_executor is the executor used to run
00076 infrastructure details
00077 */
00078   void start(int x, ::trisycl::detail::fiber_pool &fiber_executor) {
00079     x_coordinate = x;
00080 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00081     fe = &fiber_executor;
00082 #endif
00083     axi_ss.start(x, -1, fiber_executor);
00084   }
00085
00086
00087 /// \todo Factorize some code between core tile and shim tile
00088 /** Get the input router port of the AXI stream switch
00089
00090 \param p is the slave_port_layout for the stream
00091 */
00092   auto& input(spl p) {
00093     // No index validation required because of type safety
00094     return axi_ss.input(p);
00095   }
00096
00097
00098 /** Get the output router port of the AXI stream switch
00099
00100 \param p is the master_port_layout for the stream
00101 */
00102   auto& output(mpl p) {
00103     // No index validation required because of type safety
00104     return axi_ss.output(p);
00105   }
00106
00107
00108 /** Map the input BLI id/port to the shim AXI stream switch port
00109
00110 \param[in] port is the BLI id/port to use
00111 */
00112   static auto translate_input_port(int port) {
00113     return axi_ss_t::translate_port(port, spl::south_0, spl::south_last,
00114                                     "The BLI input port is out of range");
00115   }
00116
00117
00118 /** Map the input BLI id/port to the shim AXI stream switch port
00119
00120 \param[in] port is the BLI id/port to use
00121 */
00122   static auto translate_output_port(int port) {
00123     return axi_ss_t::translate_port(port, mpl::south_0, mpl::south_last,
00124                                     "The BLI output port is out of range");
00125   }
00126
00127 public:
00128
00129 /** Get the BLI input connection from the shim tile
00130
```

```
00131 \param[in] port is the BLI id/port to use
00132 */
00133   auto& bli_in_connection(int port) {
00134     // The input is actually the output of the switch
00135     return axi_ss.out_connection(translate_output_port(port));
00136   }
00137
00138
00139 /** Get the BLI output connection to the shim tile
00140
00141 \param[in] port is the BLI id/port to use
00142 */
00143   auto& bli_out_connection(int port) {
00144     // The output is actually the input of the switch
00145     return axi_ss.in_connection(translate_input_port(port));
00146   }
00147
00148
00149 /** Get the BLI (Boundary Logic Interface) input port from the PL
00150 (Programmable Logic in the FPGA) to read data from
00151
00152 \param[in] port is the BLI id/port to use
00153
00154 */
00155   auto& bli_in(int port) {
00156     return *bli_in_connection(port);
00157   }
00158
00159
00160 /** Get the BLI (Boundary Logic Interface) output port to the PL
00161 (Programmable Logic in the FPGA) to write data to
00162
00163 \param[in] port is the BLI id/port to use
00164 */
00165   auto& bli_out(int port) {
00166     return *bli_out_connection(port);
00167   }
00168
00169
00170 /// Configure a connection of the shim tile AXI stream switch
00171   void connect(typename geo::shim_axi_stream_switch::slave_port_layout sp,
00172                typename geo::shim_axi_stream_switch::master_port_layout mp) {
00173     axi_ss.connect(sp, mp);
00174   }
00175
00176
00177 };
00178
00179 /// @} End the aie Doxygen group
00180
00181 }
00182
00183 /*
00184 # Some Emacs stuff:
00185 ### Local Variables:
00186 ### ispell-local-dictionary:  "american"
00187 ### eval:  (flyspell-prog-mode)
00188 ### End:
00189 */
00190
00191 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_SHIM_TILE_HPP
```

13.367 include/triSYCL/vendor/Xilinx/acap/aie/shim_tile.hpp File Reference

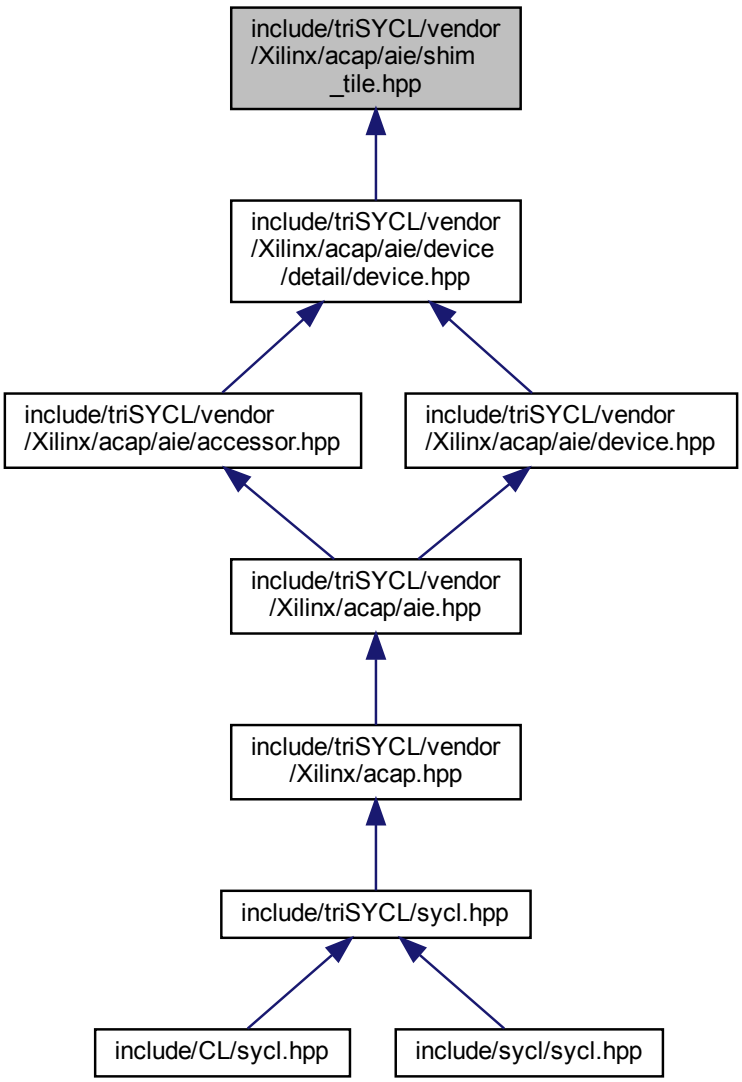
Model of a shim tile between the AI array and the rest of the circuit.

```
#include "axi_stream_switch.hpp"
```

Include dependency graph for shim_tile.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >

  The AI Engine shim tile infrastructure. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.367.1 Detailed Description

Model of a shim tile between the AI array and the rest of the circuit.

Based on Math Engine (ME) Architecture Specification Revision v2.01 March 2019, section "6 ME Shim Architecture".

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file shim_tile.hpp.

13.368 shim_tile.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_SHIM_TILE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_ARRAY_SHIM_TILE_HPP
00003
00004 /** \file
00005
00006 Model of a shim tile between the AI array and the rest of the circuit
00007
00008 Based on Math Engine (ME) Architecture Specification Revision
00009 v2.01 March 2019, section "6 ME Shim Architecture".
00010
00011 Ronan dot Keryell at Xilinx dot com
00012
00013 This file is distributed under the University of Illinois Open Source
00014 License.  See LICENSE.TXT for details.
00015 */
00016
00017 #include "axi_stream_switch.hpp"
00018
00019 namespace trisycl::vendor::xilinx::acap::aie {
00020
00021 /// \ingroup aie
00022 /// @{
00023
00024 /** The AI Engine shim tile infrastructure
00025
00026 \param AIE is the type representing the full CGRA with the
00027 programs and memory contents
00028
00029 \todo Assume only AIE Shim PL tiles for now
00030 */
00031 template <typename AIEDevice>
00032 class shim_tile {
00033   using geo = typename AIEDevice::geo;
00034   using axi_ss_geo = typename geo::shim_axi_stream_switch;
00035   using mpl = typename axi_ss_geo::master_port_layout;
00036   using spl = typename axi_ss_geo::slave_port_layout;
00037   using axi_ss_t = axi_stream_switch<axi_ss_geo>;
00038
00039 /// Keep the horizontal coordinate for debug purpose
00040   int x_coordinate;
00041
00042 /// \todo Remove?
00043 /// Keep the vertical coordinate for debug purpose
00044   int y_coordinate = -1;
00045
00046 /// The AXI stream switch of the core tile
00047   axi_ss_t axi_ss;
00048
00049 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00050 /// Keep track of the fiber executor
00051   ::trisycl::detail::fiber_pool *fe;
00052
00053 /// To shepherd the working fiber
00054   ::trisycl::detail::fiber_pool::future<void> future_work;
00055 #else
00056 /// Keep track of the std::thread execution in this tile
00057   std::future<void> future_work;
00058 #endif
00059
```

```
00060 public:
00061
00062 /// Construct the shim tile infrastructure
00063   shim_tile() {
00064     // Connect the core receivers to its AXI stream switch
00065     for (auto p :  axi_ss_geo::m_south_range)
00066       output(p) = std::make_shared<port_receiver<axi_ss_t»
00067         (axi_ss, "BLI_shim_receiver");
00068   }
00069
00070
00071 /** Start the shim tile infrastructure associated to the AIE device
00072
00073 \param[in] x is the horizontal coordinate for this tile
00074
00075 \param[in] fiber_executor is the executor used to run
00076 infrastructure details
00077 */
00078   void start(int x, ::trisycl::detail::fiber_pool &fiber_executor) {
00079     x_coordinate = x;
00080 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00081     fe = &fiber_executor;
00082 #endif
00083     axi_ss.start(x, -1, fiber_executor);
00084   }
00085
00086
00087 /// \todo Factorize some code between core tile and shim tile
00088 /** Get the input router port of the AXI stream switch
00089
00090 \param p is the slave_port_layout for the stream
00091 */
00092   auto& input(spl p) {
00093     // No index validation required because of type safety
00094     return axi_ss.input(p);
00095   }
00096
00097
00098 /** Get the output router port of the AXI stream switch
00099
00100 \param p is the master_port_layout for the stream
00101 */
00102   auto& output(mpl p) {
00103     // No index validation required because of type safety
00104     return axi_ss.output(p);
00105   }
00106
00107
00108 /** Map the input BLI id/port to the shim AXI stream switch port
00109
00110 \param[in] port is the BLI id/port to use
00111 */
00112   static auto translate_input_port(int port) {
00113     return axi_ss_t::translate_port(port, spl::south_0, spl::south_last,
00114                                     "The BLI input port is out of range");
00115   }
00116
00117
00118 /** Map the input BLI id/port to the shim AXI stream switch port
00119
00120 \param[in] port is the BLI id/port to use
00121 */
00122   static auto translate_output_port(int port) {
00123     return axi_ss_t::translate_port(port, mpl::south_0, mpl::south_last,
00124                                     "The BLI output port is out of range");
00125   }
00126
00127 public:
00128
00129 /** Get the BLI input connection from the shim tile
00130
00131 \param[in] port is the BLI id/port to use
00132 */
00133   auto& bli_in_connection(int port) {
00134     // The input is actually the output of the switch
00135     return axi_ss.out_connection(translate_output_port(port));
00136   }
00137
00138
00139 /** Get the BLI output connection to the shim tile
00140
00141 \param[in] port is the BLI id/port to use
00142 */
00143   auto& bli_out_connection(int port) {
00144     // The output is actually the input of the switch
00145     return axi_ss.in_connection(translate_input_port(port));
00146   }
```

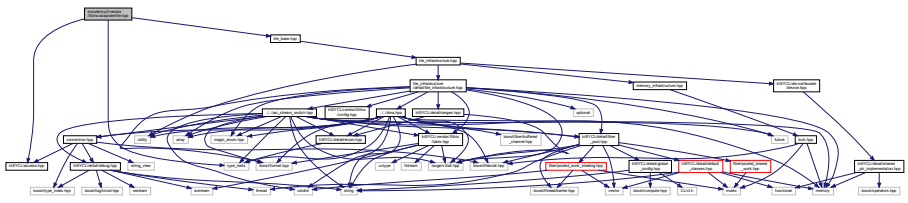
```
00147
00148
00149 /** Get the BLI (Boundary Logic Interface) input port from the PL
00150 (Programmable Logic in the FPGA) to read data from
00151
00152 \param[in] port is the BLI id/port to use
00153
00154 */
00155   auto& bli_in(int port) {
00156     return *bli_in_connection(port);
00157   }
00158
00159
00160 /** Get the BLI (Boundary Logic Interface) output port to the PL
00161 (Programmable Logic in the FPGA) to write data to
00162
00163 \param[in] port is the BLI id/port to use
00164 */
00165   auto& bli_out(int port) {
00166     return *bli_out_connection(port);
00167   }
00168
00169
00170 /// Configure a connection of the shim tile AXI stream switch
00171   void connect(typename geo::shim_axi_stream_switch::slave_port_layout sp,
00172                typename geo::shim_axi_stream_switch::master_port_layout mp) {
00173     axi_ss.connect(sp, mp);
00174   }
00175
00176
00177 };
00178
00179 /// @} End the aie Doxygen group
00180
00181 }
00182
00183 /*
00184 # Some Emacs stuff:
00185 ### Local Variables:
00186 ### ispell-local-dictionary:  "american"
00187 ### eval:  (flyspell-prog-mode)
00188 ### End:
00189 */
00190
00191 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_SHIM_TILE_HPP
```

13.369 include/sycl/vendor/Xilinx/acap/aie/tile.hpp File Reference

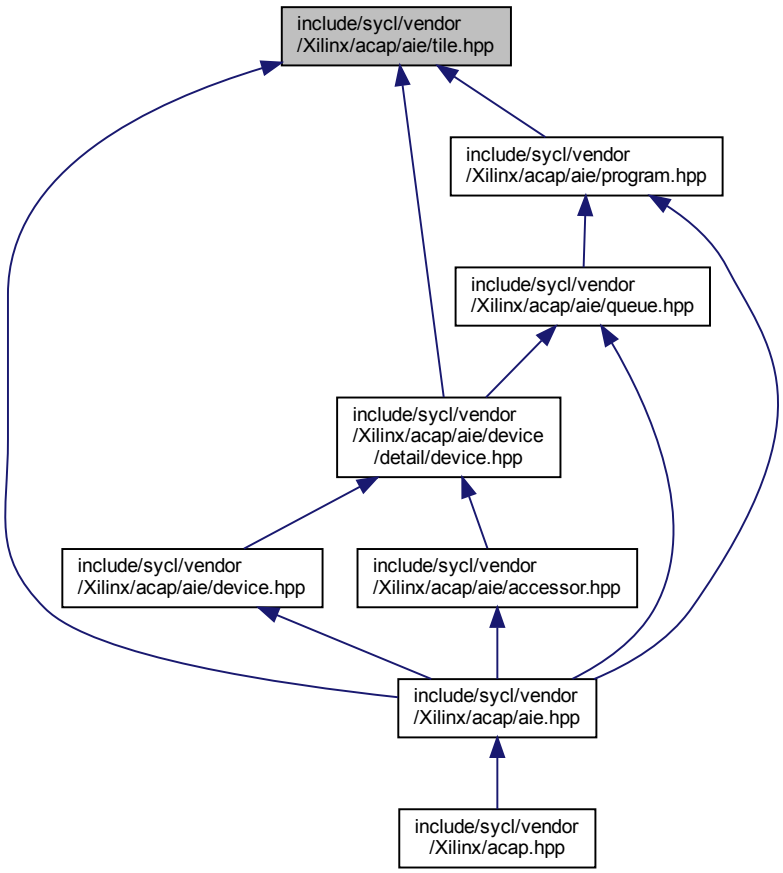
The basic AI Engine heterogeneous tile that dependent of x & y coordinates but also from the collective program run on it.

```
#include <type_traits>
#include "triSYCL/access.hpp"
#include "tile_base.hpp"
```

Include dependency graph for tile.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >

  The AI Engine tile infrastructure defining the program of a tile. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.369.1 Detailed Description

The basic AI Engine heterogeneous tile that dependent of x & y coordinates but also from the collective program run on it.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file tile.hpp.

13.370 tile.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine heterogeneous tile that dependent of x & y
00007 coordinates but also from the collective program run on it.
00008
00009 Ronan dot Keryell at Xilinx dot com
00010
00011 This file is distributed under the University of Illinois Open Source
00012 License.  See LICENSE.TXT for details.
00013 */
00014
00015 #include <type_traits>
00016
00017 #include "triSYCL/access.hpp"
00018 #include "tile_base.hpp"
00019
00020 namespace trisycl::vendor::xilinx::acap::aie {
00021
00022 /// \ingroup aie
00023 /// @{
00024
00025 /** The AI Engine tile infrastructure defining the program of a tile
00026
00027 This is the type you need to inherit from to define the program of
00028 a CGRA tile.
00029
00030 \param AIE_Program is the type representing the full CGRA with the
00031 programs and memory contents
00032
00033 \param X is the horizontal coordinate of the memory module
00034
00035 \param Y is the vertical coordinate of the memory module
00036
00037 \todo (Andrew Gozillon) FYI, until Intel fix the way they handle
00038 base classes in the SYCL compiler frontend this tile_base can
00039 never have non-default constructed data (on the device side
00040 implementation, the host should be fine to have whatever).  As the
00041 hack I have in place just now only outlines a call to the base
00042 classes default constructor, it might be feasible to work around
00043 that though by calling a base class "fill" function in the tile
00044 constructor or tweaking the way the run function of a tile is
00045 outlined so that it outlines it as a copy of the full object
00046 rather than components of the object and reconstructing it inside
00047 the kernel (although you'll fall into the standard layout trap as
00048 you'll be passing a full object to the kernel invocation and it'd
00049 take some reworking of the Tile's main generation).
00050
00051 A proper fix for this is probably quite difficult as you'd need to
00052 take into account the base classes dependencies among-st other
00053 things like the constructor body not being available in the same
00054 TU, some discussion here:  https://github.com/intel/llvm/issues/488
00055 */
00056 template <typename AIE_Program, int X, int Y>
00057 struct tile :  tile_base<AIE_Program> {
00058 /** The horizontal tile coordinates in the CGRA grid (starting at 0
00059 and increasing towards the East) */
00060   static auto constexpr x = X;
00061 /** The vertical tile coordinates in the CGRA grid (starting at
00062 increasing towards the North) */
```

```
00063   static auto constexpr y = Y;
00064
00065 /// The geography of the CGRA
00066   using geo = typename AIE_Program::geo;
00067
00068 /// Shortcut to the tile base class
00069   using tb = tile_base<AIE_Program>;
00070
00071
00072 /** Return the coordinate of the tile in the given dimension
00073
00074 \param Dim is 0 for the horizontal (x) dimension and 1 for the
00075 vertical (y) dimension
00076 */
00077   template <int Dim>
00078   static auto constexpr get_id() {
00079     static_assert(0 <= Dim && Dim <= 1,
00080                   "The dimension has to be between 0 and 1");
00081     if constexpr (Dim == 0)
00082       return x;
00083     else
00084       return y;
00085   }
00086
00087
00088 /// Return the linearized coordinate of the tile
00089   static auto constexpr linear_id() {
00090     return geo::linear_id(x, y);
00091   }
00092
00093 /** \todo Need something more modern than Boost.Operator to
00094 implement trisycl::id
00095
00096 Look for example at https://github.com/taocpp/operators/issues/9
00097 static auto constexpr get_id() {
00098 return trisycl::id<2> { x, y };
00099 }
00100 */
00101
00102
00103 /// Test if the tile is in the Western column
00104   static bool constexpr is_west_column() {
00105     return geo::is_west_column(x);
00106   }
00107
00108
00109 /// Test if the tile is in the Eastern column
00110   static bool constexpr is_east_column() {
00111     return geo::is_east_column(x);
00112   }
00113
00114
00115 /// Test if the tile is in the Southern row
00116   static bool constexpr is_south_row() {
00117     return geo::is_south_row(y);
00118   }
00119
00120
00121 /// Test if the tile is in the Northern row
00122   static bool constexpr is_north_row() {
00123     return geo::is_north_row(y);
00124   }
00125
00126
00127 /// Test if the tile has a Western neighbor
00128   static bool constexpr is_west_valid() {
00129     return !geo::is_west_column(x);
00130   }
00131
00132
00133 /// Test if the tile has a Eastern neighbor
00134   static bool constexpr is_east_valid() {
00135     return !geo::is_east_column(x);
00136   }
00137
00138
00139 /// Test if the tile has a Southern neighbor
00140   static bool constexpr is_south_valid() {
00141     return !geo::is_south_row(y);
00142   }
00143
00144
00145 /// Test if the tile has a South West neighbor
00146   static bool constexpr is_south_west_valid() {
00147     return is_south_valid() && is_west_valid();
00148   }
00149
```

```
00150
00151 /// Test if the tile has a South East neighbor
00152   static bool constexpr is_south_east_valid() {
00153     return is_south_valid() && is_east_valid();
00154   }
00155
00156
00157 /// Test if the tile has a Northern neighbor
00158   static bool constexpr is_north_valid() {
00159     return !geo::is_north_row(y);
00160   }
00161
00162
00163 /// Test if the tile has a North East neighbor
00164   static bool constexpr is_north_east_valid() {
00165     return is_north_valid() && is_east_valid();
00166   }
00167
00168
00169 /// Test if the tile has a North West neighbor
00170   static bool constexpr is_north_west_valid() {
00171     return is_north_valid() && is_west_valid();
00172   }
00173
00174
00175 /** Test if a memory module exists and is connected to this tile
00176
00177 \param[in] dx is the horizontal displacement (-1,0,+1)
00178
00179 \param[in] dy is the vertical displacement (-1,0,+1)
00180 */
00181   static bool constexpr is_memory_module(int dx, int dy) {
00182     return geo::is_memory_module(x, y, dx, dy);
00183   }
00184
00185
00186 /// Test if a memory module exists on the West of this tile
00187   static bool constexpr is_memory_module_west() {
00188     return is_memory_module(-1, 0);
00189   }
00190
00191
00192 /// Test if a memory module exists on the East of this tile
00193   static bool constexpr is_memory_module_east() {
00194     return is_memory_module(1, 0);
00195   }
00196
00197
00198 /// Test if a memory module exists on the South this tile
00199   static bool constexpr is_memory_module_south() {
00200     return is_memory_module(0, -1);
00201   }
00202
00203
00204 /// Test if a memory module exists on the North this tile
00205   static bool constexpr is_memory_module_north() {
00206     return is_memory_module(0, 1);
00207   }
00208
00209
00210 /** Compute the linear id of a memory module attached to this tile
00211
00212 \param[in] dx is the horizontal displacement (-1,0,+1)
00213
00214 \param[in] dy is the vertical displacement (-1,0,+1)
00215 */
00216   static auto constexpr memory_module_linear_id(int dx, int dy) {
00217     return geo::  memory_module_linear_id(x, y, dx, dy);
00218   }
00219
00220
00221 /// Get the memory module on the West if it does exist
00222    auto &mem_west() {
00223      static_assert(is_memory_module_west(), "There is no memory module"
00224                    " on the West of this tile in the Western column and"
00225                    " on an even row");
00226      return tb::program->template
00227        memory_module<memory_module_linear_id(-1, 0)>();
00228   }
00229
00230
00231 /// Get the memory module on the East if it does exist
00232   auto &mem_east() {
00233     static_assert(is_memory_module_east(), "There is no memory module"
00234                   " on the East of this tile in the Eastern column and"
00235                   " on an odd row");
00236     return tb::program->template
```

```
00237       memory_module<memory_module_linear_id(1, 0)>();
00238   }
00239
00240
00241 /// Get the memory module on the South if it does exist
00242   auto &mem_south() {
00243     static_assert(is_memory_module_south(), "There is no memory module"
00244                   " below the Southern tile row");
00245     return tb::program->template
00246       memory_module<memory_module_linear_id(0, -1)>();
00247   }
00248
00249
00250 /// Get the memory module on the North if it does exist
00251   auto &mem_north() {
00252     static_assert(is_memory_module_north(), "There is no memory module"
00253                   " above the Northern tile row");
00254     return tb::program->template
00255       memory_module<memory_module_linear_id(0, 1)>();
00256   }
00257
00258
00259 /// The memory module native to the tile
00260   auto &mem() {
00261     if constexpr (y & 1)
00262       return mem_west();
00263     else
00264       return mem_east();
00265   }
00266
00267
00268 /** Get the memory module relative to the tile
00269
00270 \param[in] Dx is the horizontal offset relative to the current core tile
00271
00272 \param[in] Dy is the vertical offset relative to the current core tile
00273
00274 \return the memory tile object
00275
00276 Note that since a core tile has a 4-neighbor connectivity, at
00277 least one of the offset needs to be 0 and the other to be either
00278 1 or -1.
00279 */
00280   template <int Dx, int Dy>
00281   auto &mem() {
00282     static_assert(geo::is_valid_memory_module_offset(Dx, Dy),
00283                   "Note that since a core tile has a 4-neighbor connectivity,"
00284                   " one of the offset needs to be 0 and the other to be either"
00285                   " 1 or -1");
00286     if constexpr (Dx == -1)
00287       return mem_west();
00288     else if constexpr (Dx == 1)
00289       return mem_east();
00290     else if constexpr (Dy == -1)
00291       return mem_south();
00292     else
00293       return mem_north();
00294   }
00295
00296
00297 /// The type of the memory module native to the tile
00298   using mem_t = typename AIE_Program::template tileable_memory<x, y>;
00299
00300
00301 /// Test if this tile owns the start of the cascade_stream
00302   static bool constexpr is_cascade_start() {
00303     return geo::is_cascade_start(x, y);
00304   }
00305
00306
00307 /// Test if this tile owns the end of the cascade_stream
00308   static bool constexpr is_cascade_end() {
00309     return geo::is_cascade_end(x, y);
00310   }
00311
00312
00313 /** Compute the linearized coordinate of a tile along the cascade
00314 stream */
00315   static auto constexpr cascade_linear_id() {
00316     return geo::cascade_linear_id(x, y);
00317   }
00318
00319
00320 /** Get a read accessor to the cascade stream input
00321
00322 \param T is the data type used to read from the cascade
00323 stream pipe
```

```
00324
00325 \param Target is the access mode to the pipe.  It is blocking
00326 by default
00327 */
00328   template <typename T>
00329   auto get_cascade_stream_in() {
00330     static_assert(!is_cascade_start(), "You cannot access to the cascade stream"
00331                   " input on the tile that starts the stream");
00332     return tb::cascade().template get_cascade_stream_in<T>(x, y);
00333   }
00334
00335
00336 /** Get a write accessor to the cascade stream output
00337
00338 \param T is the data type used to write to the cascade
00339 stream pipe
00340
00341 \param Target is the access mode to the pipe.  It is blocking
00342 by default
00343 */
00344   template <typename T>
00345   auto get_cascade_stream_out() {
00346     static_assert(!is_cascade_end(), "You cannot access to the cascade stream"
00347                   " output on the tile that starts the stream");
00348     return tb::cascade().template get_cascade_stream_out<T>(x, y);
00349   }
00350
00351
00352 /** An horizontal barrier using a lock
00353
00354 Implement a barrier across the tiles a line.
00355
00356 \param[in] lock is the ME lock to use.  The lock 14 is used by
00357 default
00358 */
00359   void horizontal_barrier(int lock = 14) {
00360     if constexpr (y & 1) {
00361       // Propagate a token from West to East and back
00362       if constexpr (!is_west_column()) {
00363         // Wait for the Western neighbour to be ready
00364         mem().lock(lock).acquire_with_value(true);
00365       }
00366       if constexpr (is_memory_module_east()) {
00367         mem_east().lock(lock).acquire_with_value(false);
00368         // Unleash the Eastern neighbour
00369         mem_east().lock(lock).release_with_value(true);
00370         // Wait for the Eastern neighbour to acknowledge
00371         mem_east().lock(lock).acquire_with_value(false);
00372        }
00373       if constexpr (!is_west_column()) {
00374         // Acknowledge to the Western neighbour
00375         mem().lock(lock).release_with_value(false);
00376       }
00377     } else {
00378       // Propagate a token from East to West and back
00379       if constexpr (!is_east_column()) {
00380         // Wait for the Eastern neighbour to be ready
00381         mem().lock(lock).acquire_with_value(true);
00382       }
00383       if constexpr (is_memory_module_west()) {
00384         mem_west().lock(lock).acquire_with_value(false);
00385         // Unleash the Western neighbour
00386         mem_west().lock(lock).release_with_value(true);
00387         // Wait for the Western neighbour to acknowledge
00388         mem_west().lock(lock).acquire_with_value(false);
00389        }
00390       if constexpr (!is_east_column()) {
00391         // Acknowledge to the Eastern neighbour
00392         mem().lock(lock).release_with_value(false);
00393       }
00394     }
00395   }
00396
00397
00398 /** A vertical barrier using a lock
00399
00400 Implement a barrier across the tiles a line.
00401
00402 \param[in] lock is the ME lock to use.  The lock 15 is used by
00403 default
00404 */
00405   void vertical_barrier(int lock = 15) {
00406     // Propagate a token from South to North and back
00407     if constexpr (!is_south_row()) {
00408       // Wait for the Southern neighbour to be ready
00409       mem().lock(lock).acquire_with_value(true);
00410     }
```

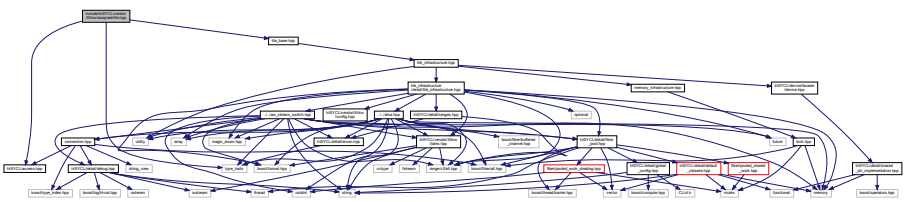
```
00411     if constexpr (is_memory_module_north()) {
00412       mem_north().lock(lock).acquire_with_value(false);
00413       // Unleash the Northern neighbour
00414       mem_north().lock(lock).release_with_value(true);
00415       // Wait for the Northern neighbour to acknowledge
00416       mem_north().lock(lock).acquire_with_value(false);
00417     }
00418     if constexpr (!is_south_row()) {
00419       // Acknowledge to the Southern neighbour
00420       mem().lock(lock).release_with_value(false);
00421     }
00422   }
00423
00424
00425 /** Full barrier using the 2 locks by default
00426
00427 Implement a barrier across the full program by using \c
00428 horizontal_barrier() and \c vertical_barrier().
00429 */
00430   void barrier() {
00431     horizontal_barrier();
00432     vertical_barrier();
00433   }
00434
00435
00436 /** Get access on a receiver DMA
00437
00438 \param[in] port specifies which DMA to access, starting at 0 */
00439   auto rx_dma(int port) {
00440     return dma_dsl { *this, this->ti.rx_dma(port) };
00441   }
00442
00443
00444 /** Get access on a transmit DMA
00445
00446 \param[in] port specifies which DMA to access, starting at 0 */
00447   auto tx_dma(int port) {
00448     return dma_dsl { *this, this->ti.tx_dma(port) };
00449   }
00450
00451
00452 };
00453
00454 /// @} End the aie Doxygen group
00455
00456 }
00457
00458 /*
00459 # Some Emacs stuff:
00460 ### Local Variables:
00461 ### ispell-local-dictionary:  "american"
00462 ### eval:  (flyspell-prog-mode)
00463 ### End:
00464 */
00465
00466 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_HPP
```

13.371 include/triSYCL/vendor/Xilinx/acap/aie/tile.hpp File Reference

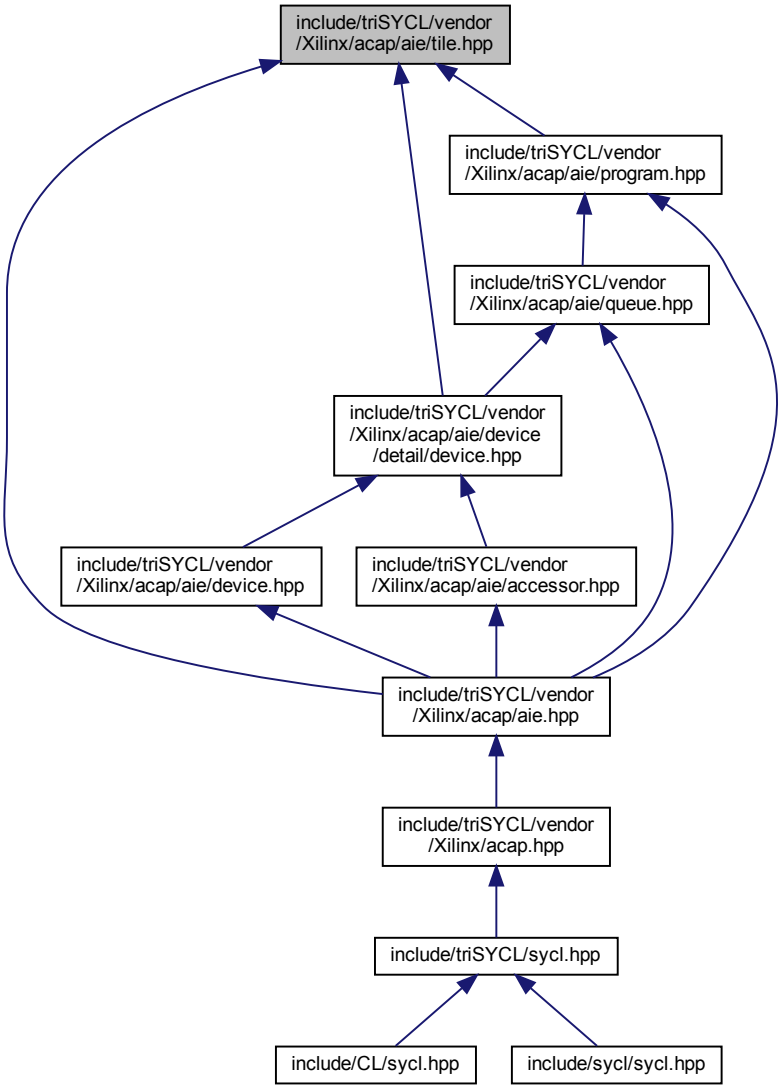
The basic AI Engine heterogeneous tile that dependent of x & y coordinates but also from the collective program run on it.

```
#include <type_traits>
#include "triSYCL/access.hpp"
#include "tile_base.hpp"
```

Include dependency graph for tile.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >

  The AI Engine tile infrastructure defining the program of a tile. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.371.1 Detailed Description

The basic AI Engine heterogeneous tile that dependent of x & y coordinates but also from the collective program run on it.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file tile.hpp.

13.372 tile.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine heterogeneous tile that dependent of x & y
00007 coordinates but also from the collective program run on it.
00008
00009 Ronan dot Keryell at Xilinx dot com
00010
00011 This file is distributed under the University of Illinois Open Source
00012 License.  See LICENSE.TXT for details.
00013 */
00014
00015 #include <type_traits>
00016
00017 #include "triSYCL/access.hpp"
00018 #include "tile_base.hpp"
00019
00020 namespace trisycl::vendor::xilinx::acap::aie {
00021
00022 /// \ingroup aie
00023 /// @{
00024
00025 /** The AI Engine tile infrastructure defining the program of a tile
00026
00027 This is the type you need to inherit from to define the program of
00028 a CGRA tile.
00029
00030 \param AIE_Program is the type representing the full CGRA with the
00031 programs and memory contents
00032
00033 \param X is the horizontal coordinate of the memory module
00034
00035 \param Y is the vertical coordinate of the memory module
00036
00037 \todo (Andrew Gozillon) FYI, until Intel fix the way they handle
00038 base classes in the SYCL compiler frontend this tile_base can
00039 never have non-default constructed data (on the device side
00040 implementation, the host should be fine to have whatever).  As the
00041 hack I have in place just now only outlines a call to the base
00042 classes default constructor, it might be feasible to work around
00043 that though by calling a base class "fill" function in the tile
00044 constructor or tweaking the way the run function of a tile is
00045 outlined so that it outlines it as a copy of the full object
00046 rather than components of the object and reconstructing it inside
00047 the kernel (although you'll fall into the standard layout trap as
00048 you'll be passing a full object to the kernel invocation and it'd
00049 take some reworking of the Tile's main generation).
00050
00051 A proper fix for this is probably quite difficult as you'd need to
00052 take into account the base classes dependencies among-st other
00053 things like the constructor body not being available in the same
00054 TU, some discussion here:  https://github.com/intel/llvm/issues/488
00055 */
00056 template <typename AIE_Program, int X, int Y>
00057 struct tile :  tile_base<AIE_Program> {
00058 /** The horizontal tile coordinates in the CGRA grid (starting at 0
00059 and increasing towards the East) */
00060   static auto constexpr x = X;
00061 /** The vertical tile coordinates in the CGRA grid (starting at
00062 increasing towards the North) */
```

```
00063   static auto constexpr y = Y;
00064
00065 /// The geography of the CGRA
00066   using geo = typename AIE_Program::geo;
00067
00068 /// Shortcut to the tile base class
00069   using tb = tile_base<AIE_Program>;
00070
00071
00072 /** Return the coordinate of the tile in the given dimension
00073
00074 \param Dim is 0 for the horizontal (x) dimension and 1 for the
00075 vertical (y) dimension
00076 */
00077   template <int Dim>
00078   static auto constexpr get_id() {
00079     static_assert(0 <= Dim && Dim <= 1,
00080                   "The dimension has to be between 0 and 1");
00081     if constexpr (Dim == 0)
00082       return x;
00083     else
00084       return y;
00085   }
00086
00087
00088 /// Return the linearized coordinate of the tile
00089   static auto constexpr linear_id() {
00090     return geo::linear_id(x, y);
00091   }
00092
00093 /** \todo Need something more modern than Boost.Operator to
00094 implement trisycl::id
00095
00096 Look for example at https://github.com/taocpp/operators/issues/9
00097 static auto constexpr get_id() {
00098 return trisycl::id<2> { x, y };
00099 }
00100 */
00101
00102
00103 /// Test if the tile is in the Western column
00104   static bool constexpr is_west_column() {
00105     return geo::is_west_column(x);
00106   }
00107
00108
00109 /// Test if the tile is in the Eastern column
00110   static bool constexpr is_east_column() {
00111     return geo::is_east_column(x);
00112   }
00113
00114
00115 /// Test if the tile is in the Southern row
00116   static bool constexpr is_south_row() {
00117     return geo::is_south_row(y);
00118   }
00119
00120
00121 /// Test if the tile is in the Northern row
00122   static bool constexpr is_north_row() {
00123     return geo::is_north_row(y);
00124   }
00125
00126
00127 /// Test if the tile has a Western neighbor
00128   static bool constexpr is_west_valid() {
00129     return !geo::is_west_column(x);
00130   }
00131
00132
00133 /// Test if the tile has a Eastern neighbor
00134   static bool constexpr is_east_valid() {
00135     return !geo::is_east_column(x);
00136   }
00137
00138
00139 /// Test if the tile has a Southern neighbor
00140   static bool constexpr is_south_valid() {
00141     return !geo::is_south_row(y);
00142   }
00143
00144
00145 /// Test if the tile has a South West neighbor
00146   static bool constexpr is_south_west_valid() {
00147     return is_south_valid() && is_west_valid();
00148   }
00149
```

```
00150
00151 /// Test if the tile has a South East neighbor
00152   static bool constexpr is_south_east_valid() {
00153     return is_south_valid() && is_east_valid();
00154   }
00155
00156
00157 /// Test if the tile has a Northern neighbor
00158   static bool constexpr is_north_valid() {
00159     return !geo::is_north_row(y);
00160   }
00161
00162
00163 /// Test if the tile has a North East neighbor
00164   static bool constexpr is_north_east_valid() {
00165     return is_north_valid() && is_east_valid();
00166   }
00167
00168
00169 /// Test if the tile has a North West neighbor
00170   static bool constexpr is_north_west_valid() {
00171     return is_north_valid() && is_west_valid();
00172   }
00173
00174
00175 /** Test if a memory module exists and is connected to this tile
00176
00177 \param[in] dx is the horizontal displacement (-1,0,+1)
00178
00179 \param[in] dy is the vertical displacement (-1,0,+1)
00180 */
00181   static bool constexpr is_memory_module(int dx, int dy) {
00182     return geo::is_memory_module(x, y, dx, dy);
00183   }
00184
00185
00186 /// Test if a memory module exists on the West of this tile
00187   static bool constexpr is_memory_module_west() {
00188     return is_memory_module(-1, 0);
00189   }
00190
00191
00192 /// Test if a memory module exists on the East of this tile
00193   static bool constexpr is_memory_module_east() {
00194     return is_memory_module(1, 0);
00195   }
00196
00197
00198 /// Test if a memory module exists on the South this tile
00199   static bool constexpr is_memory_module_south() {
00200     return is_memory_module(0, -1);
00201   }
00202
00203
00204 /// Test if a memory module exists on the North this tile
00205   static bool constexpr is_memory_module_north() {
00206     return is_memory_module(0, 1);
00207   }
00208
00209
00210 /** Compute the linear id of a memory module attached to this tile
00211
00212 \param[in] dx is the horizontal displacement (-1,0,+1)
00213
00214 \param[in] dy is the vertical displacement (-1,0,+1)
00215 */
00216   static auto constexpr memory_module_linear_id(int dx, int dy) {
00217     return geo::  memory_module_linear_id(x, y, dx, dy);
00218   }
00219
00220
00221 /// Get the memory module on the West if it does exist
00222    auto &mem_west() {
00223      static_assert(is_memory_module_west(), "There is no memory module"
00224                    " on the West of this tile in the Western column and"
00225                    " on an even row");
00226      return tb::program->template
00227        memory_module<memory_module_linear_id(-1, 0)>();
00228   }
00229
00230
00231 /// Get the memory module on the East if it does exist
00232   auto &mem_east() {
00233     static_assert(is_memory_module_east(), "There is no memory module"
00234                   " on the East of this tile in the Eastern column and"
00235                   " on an odd row");
00236     return tb::program->template
```

```
00237       memory_module<memory_module_linear_id(1, 0)>();
00238   }
00239
00240
00241 /// Get the memory module on the South if it does exist
00242   auto &mem_south() {
00243     static_assert(is_memory_module_south(), "There is no memory module"
00244                   " below the Southern tile row");
00245     return tb::program->template
00246       memory_module<memory_module_linear_id(0, -1)>();
00247   }
00248
00249
00250 /// Get the memory module on the North if it does exist
00251   auto &mem_north() {
00252     static_assert(is_memory_module_north(), "There is no memory module"
00253                   " above the Northern tile row");
00254     return tb::program->template
00255       memory_module<memory_module_linear_id(0, 1)>();
00256   }
00257
00258
00259 /// The memory module native to the tile
00260   auto &mem() {
00261     if constexpr (y & 1)
00262       return mem_west();
00263     else
00264       return mem_east();
00265   }
00266
00267
00268 /** Get the memory module relative to the tile
00269
00270 \param[in] Dx is the horizontal offset relative to the current core tile
00271
00272 \param[in] Dy is the vertical offset relative to the current core tile
00273
00274 \return the memory tile object
00275
00276 Note that since a core tile has a 4-neighbor connectivity, at
00277 least one of the offset needs to be 0 and the other to be either
00278 1 or -1.
00279 */
00280   template <int Dx, int Dy>
00281   auto &mem() {
00282     static_assert(geo::is_valid_memory_module_offset(Dx, Dy),
00283                   "Note that since a core tile has a 4-neighbor connectivity,"
00284                   " one of the offset needs to be 0 and the other to be either"
00285                   " 1 or -1");
00286     if constexpr (Dx == -1)
00287       return mem_west();
00288     else if constexpr (Dx == 1)
00289       return mem_east();
00290     else if constexpr (Dy == -1)
00291       return mem_south();
00292     else
00293       return mem_north();
00294   }
00295
00296
00297 /// The type of the memory module native to the tile
00298   using mem_t = typename AIE_Program::template tileable_memory<x, y>;
00299
00300
00301 /// Test if this tile owns the start of the cascade_stream
00302   static bool constexpr is_cascade_start() {
00303     return geo::is_cascade_start(x, y);
00304   }
00305
00306
00307 /// Test if this tile owns the end of the cascade_stream
00308   static bool constexpr is_cascade_end() {
00309     return geo::is_cascade_end(x, y);
00310   }
00311
00312
00313 /** Compute the linearized coordinate of a tile along the cascade
00314 stream */
00315   static auto constexpr cascade_linear_id() {
00316     return geo::cascade_linear_id(x, y);
00317   }
00318
00319
00320 /** Get a read accessor to the cascade stream input
00321
00322 \param T is the data type used to read from the cascade
00323 stream pipe
```

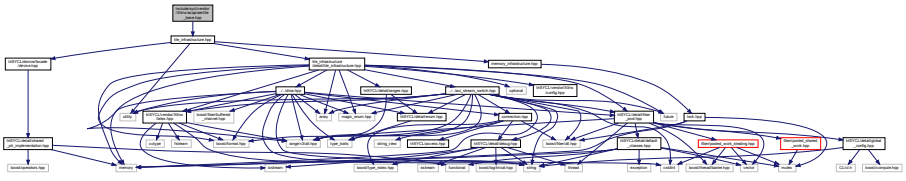
```
00324
00325 \param Target is the access mode to the pipe.  It is blocking
00326 by default
00327 */
00328   template <typename T>
00329   auto get_cascade_stream_in() {
00330     static_assert(!is_cascade_start(), "You cannot access to the cascade stream"
00331                   " input on the tile that starts the stream");
00332     return tb::cascade().template get_cascade_stream_in<T>(x, y);
00333   }
00334
00335
00336 /** Get a write accessor to the cascade stream output
00337
00338 \param T is the data type used to write to the cascade
00339 stream pipe
00340
00341 \param Target is the access mode to the pipe.  It is blocking
00342 by default
00343 */
00344   template <typename T>
00345   auto get_cascade_stream_out() {
00346     static_assert(!is_cascade_end(), "You cannot access to the cascade stream"
00347                   " output on the tile that starts the stream");
00348     return tb::cascade().template get_cascade_stream_out<T>(x, y);
00349   }
00350
00351
00352 /** An horizontal barrier using a lock
00353
00354 Implement a barrier across the tiles a line.
00355
00356 \param[in] lock is the ME lock to use.  The lock 14 is used by
00357 default
00358 */
00359   void horizontal_barrier(int lock = 14) {
00360     if constexpr (y & 1) {
00361       // Propagate a token from West to East and back
00362       if constexpr (!is_west_column()) {
00363         // Wait for the Western neighbour to be ready
00364         mem().lock(lock).acquire_with_value(true);
00365       }
00366       if constexpr (is_memory_module_east()) {
00367         mem_east().lock(lock).acquire_with_value(false);
00368         // Unleash the Eastern neighbour
00369         mem_east().lock(lock).release_with_value(true);
00370         // Wait for the Eastern neighbour to acknowledge
00371         mem_east().lock(lock).acquire_with_value(false);
00372        }
00373       if constexpr (!is_west_column()) {
00374         // Acknowledge to the Western neighbour
00375         mem().lock(lock).release_with_value(false);
00376       }
00377     } else {
00378       // Propagate a token from East to West and back
00379       if constexpr (!is_east_column()) {
00380         // Wait for the Eastern neighbour to be ready
00381         mem().lock(lock).acquire_with_value(true);
00382       }
00383       if constexpr (is_memory_module_west()) {
00384         mem_west().lock(lock).acquire_with_value(false);
00385         // Unleash the Western neighbour
00386         mem_west().lock(lock).release_with_value(true);
00387         // Wait for the Western neighbour to acknowledge
00388         mem_west().lock(lock).acquire_with_value(false);
00389        }
00390       if constexpr (!is_east_column()) {
00391         // Acknowledge to the Eastern neighbour
00392         mem().lock(lock).release_with_value(false);
00393       }
00394     }
00395   }
00396
00397
00398 /** A vertical barrier using a lock
00399
00400 Implement a barrier across the tiles a line.
00401
00402 \param[in] lock is the ME lock to use.  The lock 15 is used by
00403 default
00404 */
00405   void vertical_barrier(int lock = 15) {
00406     // Propagate a token from South to North and back
00407     if constexpr (!is_south_row()) {
00408       // Wait for the Southern neighbour to be ready
00409       mem().lock(lock).acquire_with_value(true);
00410     }
```

```
00411     if constexpr (is_memory_module_north()) {
00412       mem_north().lock(lock).acquire_with_value(false);
00413       // Unleash the Northern neighbour
00414       mem_north().lock(lock).release_with_value(true);
00415       // Wait for the Northern neighbour to acknowledge
00416       mem_north().lock(lock).acquire_with_value(false);
00417     }
00418     if constexpr (!is_south_row()) {
00419       // Acknowledge to the Southern neighbour
00420       mem().lock(lock).release_with_value(false);
00421     }
00422   }
00423
00424
00425 /** Full barrier using the 2 locks by default
00426
00427 Implement a barrier across the full program by using \c
00428 horizontal_barrier() and \c vertical_barrier().
00429 */
00430   void barrier() {
00431     horizontal_barrier();
00432     vertical_barrier();
00433   }
00434
00435
00436 /** Get access on a receiver DMA
00437
00438 \param[in] port specifies which DMA to access, starting at 0 */
00439   auto rx_dma(int port) {
00440     return dma_dsl { *this, this->ti.rx_dma(port) };
00441   }
00442
00443
00444 /** Get access on a transmit DMA
00445
00446 \param[in] port specifies which DMA to access, starting at 0 */
00447   auto tx_dma(int port) {
00448     return dma_dsl { *this, this->ti.tx_dma(port) };
00449   }
00450
00451
00452 };
00453
00454 /// @} End the aie Doxygen group
00455
00456 }
00457
00458 /*
00459 # Some Emacs stuff:
00460 ### Local Variables:
00461 ### ispell-local-dictionary:  "american"
00462 ### eval:  (flyspell-prog-mode)
00463 ### End:
00464 */
00465
00466 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_HPP
```

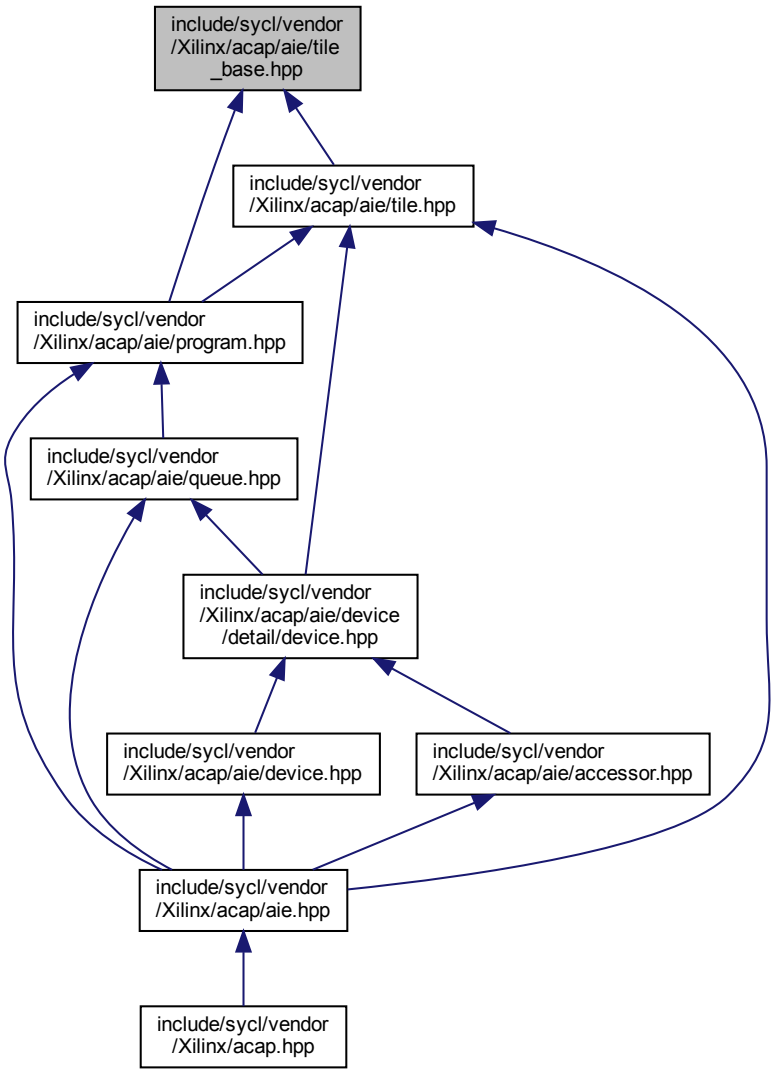
13.373 include/sycl/vendor/Xilinx/acap/aie/tile_base.hpp File Reference

The basic AI Engine homogeneous tile, with common content to all the tiles (i.e.

`#include "tile_infrastructure.hpp"`
Include dependency graph for tile_base.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >

  The AI Engine tile infrastructure common to all the tiles. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.373.1 Detailed Description

The basic AI Engine homogeneous tile, with common content to all the tiles (i.e.

independent of x & y coordinates) but depending on a collective program.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file tile_base.hpp.

13.374 tile_base.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_BASE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_BASE_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine homogeneous tile, with common content to all
00007 the tiles (i.e.  independent of x & y coordinates) but depending on
00008 a collective program.
00009
00010 Ronan dot Keryell at Xilinx dot com
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include "tile_infrastructure.hpp"
00017
00018 namespace trisycl::vendor::xilinx::acap::aie {
00019
00020 /// \ingroup aie
00021 /// @{
00022
00023 /** The AI Engine tile infrastructure common to all the tiles
00024
00025 This allows some type erasure while accessing the common
00026 tile infrastructure.
00027
00028 \param AIE_Program is the type representing the full CGRA program
00029 with the tile programs and memory contents
00030 */
00031 template <typename AIE_Program>
00032 class tile_base {
00033
00034   using geo = typename AIE_Program::device::geo;
00035
00036 protected:
00037
00038 /// Keep a reference to the AIE_Program with the full tile and memory view
00039 /// \todo can probably be removed
00040   AIE_Program *program;
00041
00042 /// Keep a reference to the tile_infrastructure hardware features
00043   tile_infrastructure<geo> ti;
00044
00045 public:
00046
00047 /** Provide a run member function that does nothing so it is
00048 possible to write a minimum AI Engine program that does nothing.
00049
00050 Note that even if this function is not virtual, in the common
00051 case a programmer implements it to specify the program executed
00052 by a tile
00053 */
00054   void run() {
00055   }
00056
00057
00058 /// Submit a callable on this tile
00059   template <typename Work>
00060   void single_task(Work &&f) {
00061     ti.single_task(std::forward<Work>(f));
```

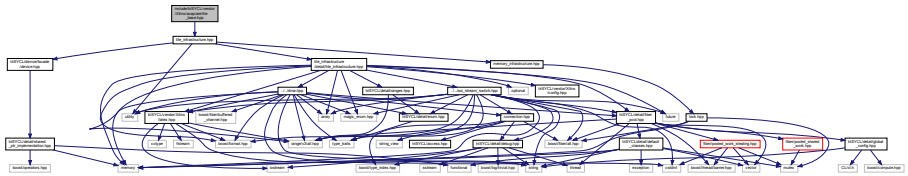
```
00062   }
00063
00064
00065 /// Wait for the execution of the callable on this tile
00066   void wait() {
00067     ti.wait();
00068   }
00069
00070
00071 /// Access the cascade connections
00072   auto &cascade() {
00073     // \todo the cascade should be part of the tile infrastructure instead
00074     return program->cascade();
00075   }
00076
00077
00078 /** Get the user input connection from the AXI stream switch
00079
00080 \param[in] port is the port to use
00081 */
00082   auto &in_connection(int port) {
00083     return ti.in_connection(port);
00084   }
00085
00086
00087 /** Get the user output connection to the AXI stream switch
00088
00089 \param[in] port is port to use
00090 */
00091   auto &out_connection(int port) {
00092     return ti.out_connection(port);
00093   }
00094
00095
00096 /// Get the user input port from the AXI stream switch
00097   auto &in(int port) {
00098     return ti.in(port);
00099   }
00100
00101
00102 /// Get the user output port to the AXI stream switch
00103   auto &out(int port) {
00104     return ti.out(port);
00105   }
00106
00107
00108 /// Store a way to access to the program
00109   void set_program(AIE_Program &p) {
00110     program = &p;
00111   }
00112
00113
00114 /// Store a way to access to hardware infrastructure of the tile
00115   void set_tile_infrastructure(tile_infrastructure<geo> t) {
00116     ti = t;
00117   }
00118
00119 };
00120
00121 /// @} End the aie Doxygen group
00122
00123 }
00124
00125 /*
00126 # Some Emacs stuff:
00127 ### Local Variables:
00128 ### ispell-local-dictionary:  "american"
00129 ### eval:  (flyspell-prog-mode)
00130 ### End:
00131 */
00132
00133 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_BASE_HPP
```

13.375 include/triSYCL/vendor/Xilinx/acap/aie/tile_base.hpp File Reference

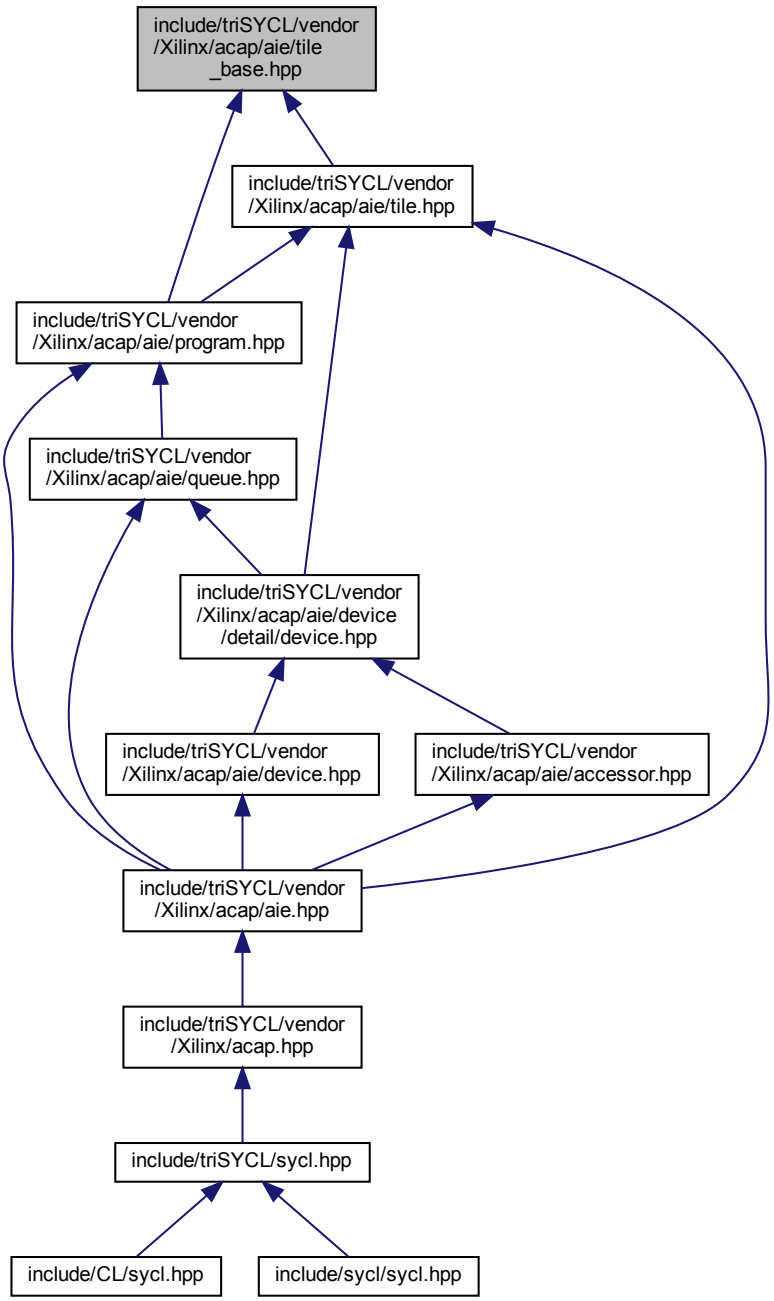
The basic AI Engine homogeneous tile, with common content to all the tiles (i.e.

```
#include "tile_infrastructure.hpp"
```

Include dependency graph for tile_base.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >

  The AI Engine tile infrastructure common to all the tiles. More...

Namespaces

- namespace trisycl

The vector type to be used as SYCL vector.

- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.375.1 Detailed Description

The basic AI Engine homogeneous tile, with common content to all the tiles (i.e.

independent of x & y coordinates) but depending on a collective program.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file tile_base.hpp.

13.376 tile_base.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_BASE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_BASE_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine homogeneous tile, with common content to all
00007 the tiles (i.e.  independent of x & y coordinates) but depending on
00008 a collective program.
00009
00010 Ronan dot Keryell at Xilinx dot com
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include "tile_infrastructure.hpp"
00017
00018 namespace trisycl::vendor::xilinx::acap::aie {
00019
00020 /// \ingroup aie
00021 /// @{
00022
00023 /** The AI Engine tile infrastructure common to all the tiles
00024
00025 This allows some type erasure while accessing the common
00026 tile infrastructure.
00027
00028 \param AIE_Program is the type representing the full CGRA program
00029 with the tile programs and memory contents
00030 */
00031 template <typename AIE_Program>
00032 class tile_base {
00033
00034   using geo = typename AIE_Program::device::geo;
00035
00036 protected:
00037
00038 /// Keep a reference to the AIE_Program with the full tile and memory view
00039 /// \todo can probably be removed
00040   AIE_Program *program;
00041
00042 /// Keep a reference to the tile_infrastructure hardware features
00043   tile_infrastructure<geo> ti;
00044
00045 public:
00046
00047 /** Provide a run member function that does nothing so it is
00048 possible to write a minimum AI Engine program that does nothing.
00049
```

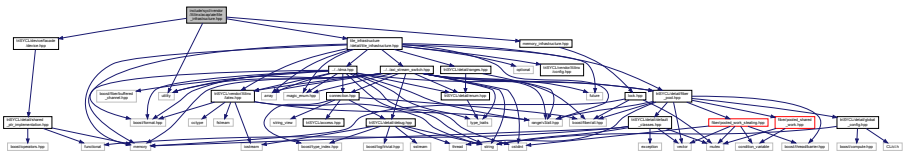
```
00050 Note that even if this function is not virtual, in the common
00051 case a programmer implements it to specify the program executed
00052 by a tile
00053 */
00054   void run() {
00055   }
00056
00057
00058 /// Submit a callable on this tile
00059   template <typename Work>
00060   void single_task(Work &&f) {
00061     ti.single_task(std::forward<Work>(f));
00062   }
00063
00064
00065 /// Wait for the execution of the callable on this tile
00066   void wait() {
00067     ti.wait();
00068   }
00069
00070
00071 /// Access the cascade connections
00072   auto &cascade() {
00073     // \todo the cascade should be part of the tile infrastructure instead
00074     return program->cascade();
00075   }
00076
00077
00078 /** Get the user input connection from the AXI stream switch
00079
00080 \param[in] port is the port to use
00081 */
00082   auto &in_connection(int port) {
00083     return ti.in_connection(port);
00084   }
00085
00086
00087 /** Get the user output connection to the AXI stream switch
00088
00089 \param[in] port is port to use
00090 */
00091   auto &out_connection(int port) {
00092     return ti.out_connection(port);
00093   }
00094
00095
00096 /// Get the user input port from the AXI stream switch
00097   auto &in(int port) {
00098     return ti.in(port);
00099   }
00100
00101
00102 /// Get the user output port to the AXI stream switch
00103   auto &out(int port) {
00104     return ti.out(port);
00105   }
00106
00107
00108 /// Store a way to access to the program
00109   void set_program(AIE_Program &p) {
00110     program = &p;
00111   }
00112
00113
00114 /// Store a way to access to hardware infrastructure of the tile
00115   void set_tile_infrastructure(tile_infrastructure<geo> t) {
00116     ti = t;
00117   }
00118
00119 };
00120
00121 /// @} End the aie Doxygen group
00122
00123 }
00124
00125 /*
00126 # Some Emacs stuff:
00127 ### Local Variables:
00128 ### ispell-local-dictionary:  "american"
00129 ### eval:  (flyspell-prog-mode)
00130 ### End:
00131 */
00132
00133 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_BASE_HPP
```

13.377 include/sycl/vendor/Xilinx/acap/aie/tile_infrastructure.hpp File Reference

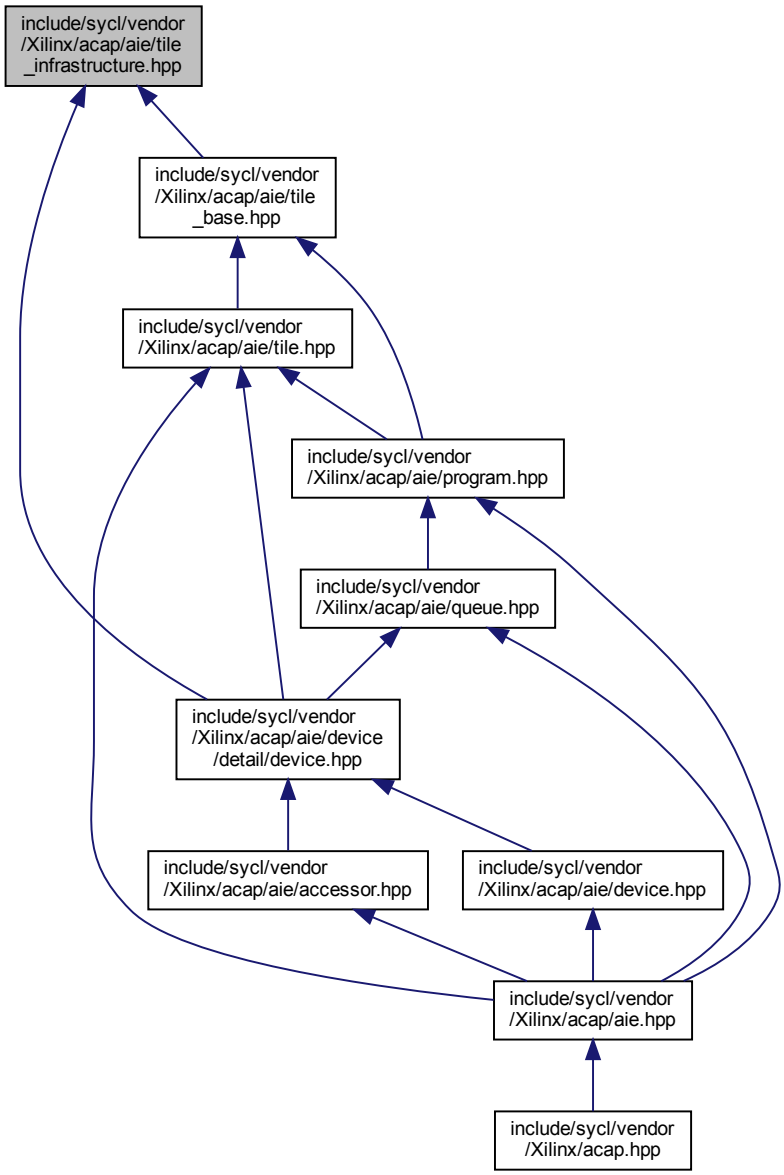
The basic AI Engine homogeneous tile, with the common infrastructure to all the tiles, i.e.

```
#include <utility>
#include "memory_infrastructure.hpp"
#include "tile_infrastructure/detail/tile_infrastructure.hpp"
#include "triSYCL/device/facade/device.hpp"
```

Include dependency graph for tile_infrastructure.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >

  The AI Engine tile infrastructure common to all the tiles. More...
- struct trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.377.1 Detailed Description

The basic AI Engine homogeneous tile, with the common infrastructure to all the tiles, i.e.

independent of x & y coordinates, but also from the tile program itself.

This tile can be seen as the SYCL subdevice of the raw CGRA SYCL device to run elemental functions.

This is owned by the device, so for example the AXI stream switch configuration and packet can survive to some program changes.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file tile_infrastructure.hpp.

13.378 tile_infrastructure.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_INFRASTRUCTURE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_INFRASTRUCTURE_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine homogeneous tile, with the common
00007 infrastructure to all the tiles, i.e.  independent of x & y
00008 coordinates, but also from the tile program itself.
00009
00010 This tile can be seen as the SYCL subdevice of the raw CGRA SYCL
00011 device to run elemental functions.
00012
00013 This is owned by the device, so for example the AXI stream switch
00014 configuration and packet can survive to some program changes.
00015
00016 Ronan dot Keryell at Xilinx dot com
00017
00018 This file is distributed under the University of Illinois Open Source
00019 License.  See LICENSE.TXT for details.
00020 */
00021
00022 #include <utility>
00023
00024 #include "memory_infrastructure.hpp"
00025 #include "tile_infrastructure/detail/tile_infrastructure.hpp"
00026 #include "triSYCL/device/facade/device.hpp"
00027
00028 namespace trisycl::vendor::xilinx::acap::aie {
00029
00030 /// \ingroup aie
00031 /// @{
00032
00033 /** The AI Engine tile infrastructure common to all the tiles
00034
00035 This allows some type erasure while accessing the common
00036 tile infrastructure.
00037
00038 \param Geography is the \c geography type representing the full
00039 layout of the architecture
00040 */
00041 template <typename Geography>
00042 class tile_infrastructure
00043     :  public facade::device<tile_infrastructure<Geography>,
```

```
00044                             detail::tile_infrastructure<Geography> {
00045 /// The type encapsulating the implementation
00046   using dti = detail::tile_infrastructure<Geography>;
00047
00048  public:
00049 /// The façade used to implement part of the user-facing type
00050   using facade_t = facade::device<tile_infrastructure<Geography>, dti>;
00051
00052   using geo = Geography;
00053   using axi_ss_geo = typename dti::axi_ss_geo;
00054   using mpl = typename dti::mpl;
00055   using spl = typename dti::spl;
00056   using axi_ss_t = typename dti::axi_ss_t;
00057
00058 /// Make the implementation member directly accessible in this class
00059   using facade_t::implementation;
00060
00061 /** Start the tile infrastructure associated to the AIE device
00062
00063 \param[in] x is the horizontal coordinate for this tile
00064
00065 \param[in] y is the vertical coordinate for this tile
00066
00067 \param[in] dev is the aie::detail::device used to control
00068 hardware when using real hardware and provide some debug
00069 information from inside the tile_infrastructure.  Use auto
00070 concept here to avoid explicit type causing circular dependency
00071
00072 \param[in] fiber_executor is the executor used to run
00073 infrastructure details
00074 */
00075   tile_infrastructure(int x, int y, auto& dev,
00076                       ::trisycl::detail::fiber_pool& fiber_executor)
00077       :  facade_t { std::make_shared<dti>(x, y, dev, fiber_executor) } {}
00078   tile_infrastructure() = default;
00079
00080  // \todo Implement a real tile handle instead of using x(), y(),
00081   // etc.  from here
00082
00083 /// Get the horizontal coordinate
00084   int x() { return implementation->x(); }
00085
00086 /// Get the vertical coordinate
00087   int y() { return implementation->y(); }
00088
00089 /// Get the horizontal number of tiles
00090   static constexpr int x_size() { return geo::x_size; }
00091
00092 /// Get the vertical number of tiles
00093   static constexpr int y_size() { return geo::y_size; }
00094
00095 /// Access to the common infrastructure part of a tile memory
00096   auto& mem() {
00097     return implementation->mem();
00098   }
00099
00100   struct cgh_t {
00101     tile_infrastructure ti;
00102     cgh_t(tile_infrastructure t) :  ti { t } {}
00103
00104   template <typename Invocable>
00105     void single_task(Invocable&& t) {
00106       ti.single_task([=] { t(); });
00107     }
00108   };
00109
00110 /// Submit a command group to the tile
00111   template <typename Invocable>
00112   void submit(Invocable&& f)const {
00113     cgh_t cgh { *this };
00114     std::forward<Invocable>(f)(cgh);
00115   }
00116
00117 /** Get the user input connection from the AXI stream switch
00118
00119 \param[in] port is the port to use
00120 */
00121   auto& in_connection(int port) { return implementation->in_connection(port); }
00122
00123 /** Get the user output connection to the AXI stream switch
00124
00125 \param[in] port is port to use
00126 */
00127   auto& out_connection(int port) {
00128     return implementation->out_connection(port);
00129   }
00130
```

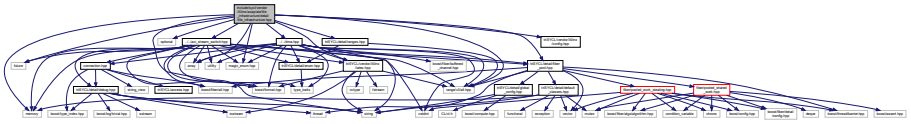
```
00131 /** Get the user input port from the AXI stream switch
00132
00133 \param[in] port is the port to use
00134 */
00135   auto& in(int port) { return implementation->in(port); }
00136
00137 /** Get the user output port to the AXI stream switch
00138
00139 \param[in] port is the port to use
00140 */
00141   auto& out(int port) { return implementation->out(port); }
00142
00143 /** Get access to a receiver DMA
00144
00145 \param[in] id specifies which DMA to access */
00146   auto& rx_dma(int id) { return implementation->rx_dma(id); }
00147
00148 /** Get access to a transmit DMA
00149
00150 \param[in] id specifies which DMA to access */
00151   auto& tx_dma(int id) { return implementation->tx_dma(id); }
00152
00153 /** Get the input router port of the AXI stream switch
00154
00155 \param p is the slave_port_layout for the stream
00156 */
00157   auto& input(spl p) { return implementation->input(p); }
00158
00159 /** Get the output router port of the AXI stream switch
00160
00161 \param p is the master_port_layout for the stream
00162 */
00163   auto& output(mpl p) { return implementation->output(p); }
00164
00165 /// Launch a callable on this tile
00166   template <typename Work> auto& single_task(Work&& f) {
00167     /* Add an immediate lambda call to avoid a warning about capturing
00168 this when non using it */
00169     auto kernel = [&] {
00170       if constexpr (requires { f(); })
00171         /* If the invocable is not interested by the handler, do not
00172 provide it.  Use a copy of the callable to make the
00173 separation from host to device memory very clear.  */
00174         return [work = f] () mutable { return work(); };
00175       else
00176 /** Recycle the tile_infrastructure shared_ptr as a fake tile
00177 handle which is copyable.
00178
00179 \todo In a device implementation we should have a real
00180 tile_handler type making sense on the device instead of
00181 just *this */
00182         return [this, work = f] () mutable { return work(*this); };
00183     }();
00184
00185     implementation->single_task(kernel);
00186     // To allow chaining commands
00187     return *this;
00188   }
00189
00190 /// Wait for the execution of the callable on this tile
00191   auto& wait() {
00192     implementation->wait();
00193     // To allow chaining commands
00194     return *this;
00195   }
00196
00197 /// Configure a connection of the core tile AXI stream switch
00198   auto& connect(typename geo::core_axi_stream_switch::slave_port_layout sp,
00199                 typename geo::core_axi_stream_switch::master_port_layout mp) {
00200     implementation->connect(sp, mp);
00201     // To allow chaining commands
00202     return *this;
00203   }
00204
00205 /// Compute the size of the graphics representation of the tile
00206   static vec<int, 2> display_size() { return dti::display_size(); }
00207
00208 /// Display the tile to a LaTeX context
00209   void display(latex::context& c)const { implementation->display(c); }
00210 };
00211
00212 /// @} End the aie Doxygen group
00213
00214 } // namespace trisycl::vendor::xilinx::acap::aie
00215
00216 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_INFRASTRUCTURE_HPP
```

13.379 include/sycl/vendor/Xilinx/acap/aie/tile_infrastructure/detail/tile_infrastructure.hpp File Reference

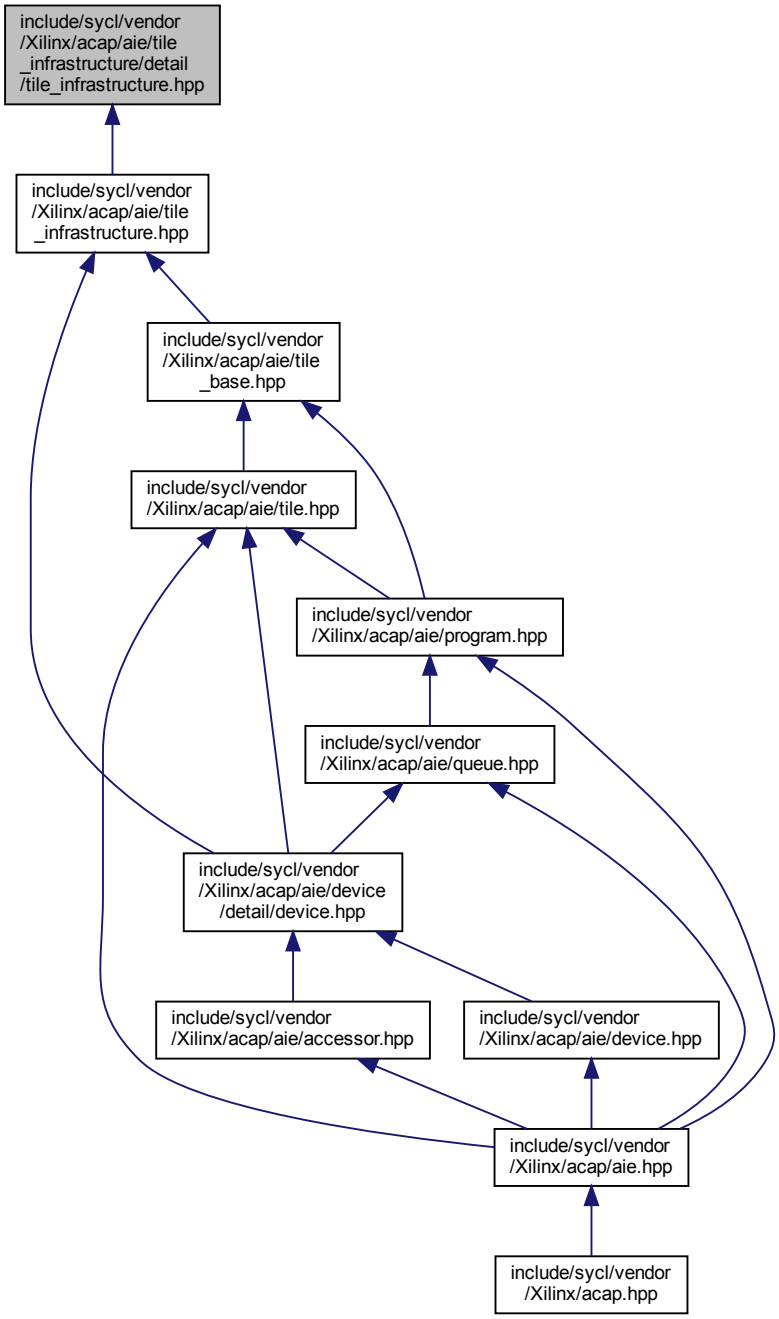
The basic AI Engine homogeneous tile, with the common infrastructure to all the tiles, i.e.

```
#include <array>
#include <future>
#include <memory>
#include <optional>
#include "magic_enum.hpp"
#include <boost/format.hpp>
#include <range/v3/all.hpp>
#include "../../axi_stream_switch.hpp"
#include "../../dma.hpp"
#include "triSYCL/detail/fiber_pool.hpp"
#include "triSYCL/detail/ranges.hpp"
#include "triSYCL/vendor/Xilinx/config.hpp"
#include "triSYCL/vendor/Xilinx/latex.hpp"
```

Include dependency graph for tile_infrastructure.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >

  The AI Engine tile infrastructure common to all the tiles. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::detail

13.379.1 Detailed Description

The basic AI Engine homogeneous tile, with the common infrastructure to all the tiles, i.e.

independent of x & y coordinates, but also from the tile program itself.

This tile can be seen as the raw CGRA subdevice to run elemental functions.

This is owned by the device, so for example the AXI stream switch configuration and packet can survive to some program changes.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file tile_infrastructure.hpp.

13.380 tile_infrastructure.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_INFRASTRUCTURE_DETAIL_TILE_INFRASTRUCTURE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_INFRASTRUCTURE_DETAIL_TILE_INFRASTRUCTURE_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine homogeneous tile, with the common
00007 infrastructure to all the tiles, i.e.  independent of x & y
00008 coordinates, but also from the tile program itself.
00009
00010 This tile can be seen as the raw CGRA subdevice to run elemental
00011 functions.
00012
00013 This is owned by the device, so for example the AXI stream switch
00014 configuration and packet can survive to some program changes.
00015
00016 Ronan dot Keryell at Xilinx dot com
00017
00018 This file is distributed under the University of Illinois Open Source
00019 License.  See LICENSE.TXT for details.
00020 */
00021
00022 #include <array>
00023 #include <future>
00024 #include <memory>
00025 #include <optional>
00026
00027 #include "magic_enum.hpp"
00028 #include <boost/format.hpp>
00029 #include <range/v3/all.hpp>
00030
00031 #include "../../axi_stream_switch.hpp"
00032 #include "../../dma.hpp"
00033 #include "triSYCL/detail/fiber_pool.hpp"
00034 #include "triSYCL/detail/ranges.hpp"
00035 #include "triSYCL/vendor/Xilinx/config.hpp"
```

```
00036 #include "triSYCL/vendor/Xilinx/latex.hpp"
00037
00038 namespace trisycl::vendor::xilinx::acap::aie::detail {
00039
00040 /// \ingroup aie
00041 /// @{
00042
00043 /** The AI Engine tile infrastructure common to all the tiles
00044
00045 This allows some type erasure while accessing the common
00046 tile infrastructure.
00047
00048 \param AIE is the type representing the full CGRA with the
00049 programs and memory contents
00050 */
00051 template <typename Geography> class tile_infrastructure {
00052
00053  public:
00054   using geo = Geography;
00055   using axi_ss_geo = typename geo::core_axi_stream_switch;
00056   using mpl = typename axi_ss_geo::master_port_layout;
00057   using spl = typename axi_ss_geo::slave_port_layout;
00058   using axi_ss_t = axi_stream_switch<axi_ss_geo>;
00059
00060  private:
00061 /// Keep the horizontal coordinate
00062   int x_coordinate;
00063
00064 /// Keep the vertical coordinate
00065   int y_coordinate;
00066
00067 /** Keep track of the aie::detail::device for hardware resource
00068 control in device mode or for debugging purpose for better
00069 messages.
00070
00071 Use void* for now to avoid cyclic header dependencies for now
00072 instead of the aie::detail::device */
00073   void* dev [[maybe_unused]];
00074
00075 /// The AXI stream switch of the core tile
00076   axi_ss_t axi_ss;
00077
00078 /** Keep track of all the infrastructure tile memories
00079 of this device */
00080   aie::memory_infrastructure mi;
00081
00082 /** Sending DMAs
00083
00084 Use std::optional to postpone initialization */
00085   std::array<std::optional<sending_dma<axi_ss_t>, axi_ss_geo::s_dma_size>
00086       tx_dmas;
00087
00088 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00089 /// Keep track of the fiber executor
00090   ::trisycl::detail::fiber_pool* fe;
00091
00092 /// To shepherd the working fiber
00093   ::trisycl::detail::fiber_pool::future<void> future_work;
00094 #else
00095 /// Keep track of the std::thread execution in this tile
00096   std::future<void> future_work;
00097 #endif
00098
00099 /** Map the user input port number to the AXI stream switch port
00100
00101 \param[in] port is the user port to use
00102 */
00103   static auto translate_input_port(int port) {
00104     return axi_ss_t::translate_port(port, spl::me_0, spl::me_last,
00105                                     "The core input port is out of range");
00106   }
00107
00108 /** Map the user output port number to the AXI stream switch port
00109
00110 \param[in] port is the user port to use
00111 */
00112   static auto translate_output_port(int port) {
00113     return axi_ss_t::translate_port(port, mpl::me_0, mpl::me_last,
00114                                     "The core output port is out of range");
00115   }
00116
00117  public:
00118 /** Start the tile infrastructure associated to the AIE device tile
00119
00120 \param[in] x is the horizontal coordinate for this tile
00121
00122 \param[in] y is the vertical coordinate for this tile
```

```
00123
00124 \param[in] dev is the aie::detail::device used to control
00125 hardware when using real hardware and provide some debug
00126 information from inside the tile_infrastructure.  Use auto
00127 concept here to avoid explicit type causing circular dependency
00128
00129 \param[in] fiber_executor is the executor used to run
00130 infrastructure details
00131 */
00132   tile_infrastructure(int x, int y, auto& dev,
00133                       ::trisycl::detail::fiber_pool& fiber_executor)
00134       :  x_coordinate { x }
00135       , y_coordinate { y }
00136       , dev { &dev }
00137       , mi { dev }
00138 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00139       , fe { &fiber_executor }
00140 #endif
00141   {
00142     // Connect the core receivers to its AXI stream switch
00143     for (auto p :  views::enum_type(mpl::me_0, mpl::me_last))
00144       output(p) =
00145           std::make_shared<port_receiver<axi_ss_t»(axi_ss, "core_receiver");
00146     axi_ss.start(x, y, fiber_executor);
00147     /* Create the core tile receiver DMAs and make them directly the
00148 switch output ports */
00149     for (auto p :  axi_ss_geo::m_dma_range)
00150       output(p) =
00151           std::make_shared<receiving_dma<axi_ss_t»(axi_ss, fiber_executor);
00152     /* Create the core tile sender DMAs and connect them internally to
00153 their switch input ports */
00154     for (const auto& [d, p] :
00155          ranges::views::zip(tx_dmas, axi_ss_geo::s_dma_range))
00156       d.emplace(fiber_executor, input(p));
00157   }
00158
00159 /// Get the horizontal coordinate
00160   int x() { return x_coordinate; }
00161
00162 /// Get the vertical coordinate
00163   int y() { return y_coordinate; }
00164
00165 /// Access to the common infrastructure part of a tile memory
00166   auto& mem() {
00167     return mi;
00168   }
00169
00170 /** Get the user input connection from the AXI stream switch
00171
00172 \param[in] port is the port to use
00173 */
00174   auto& in_connection(int port) {
00175     /* The input port for the core is actually the corresponding
00176 output on the switch */
00177     return axi_ss.out_connection(translate_output_port(port));
00178   }
00179
00180 /** Get the user output connection to the AXI stream switch
00181
00182 \param[in] port is port to use
00183 */
00184   auto& out_connection(int port) {
00185     /* The output port for the core is actually the corresponding
00186 input on the switch */
00187     return axi_ss.in_connection(translate_input_port(port));
00188   }
00189
00190 /** Get the user input port from the AXI stream switch
00191
00192 \param[in] port is the port to use
00193 */
00194   auto& in(int port) {
00195     TRISYCL_DUMP_T("in(" « port « ") on tile(" « x_coordinate « ',’
00196                    « y_coordinate « ')’);
00197     return *in_connection(port);
00198   }
00199
00200 /** Get the user output port to the AXI stream switch
00201
00202 \param[in] port is the port to use
00203 */
00204   auto& out(int port) {
00205     TRISYCL_DUMP_T("out(" « port « ") on tile(" « x_coordinate « ',’
00206                    « y_coordinate « ')’);
00207     return *out_connection(port);
00208   }
00209
```

```
00210 /** Get access to a receiver DMA
00211
00212 \param[in] id specifies which DMA to access */
00213   auto& rx_dma(int id) {
00214 /** The output of the switch is actually a receiving DMA, so we
00215 can view it as a DMA */
00216     return static_cast<receiving_dma<axi_ss_t>&>(*output(
00217         axi_ss_t::translate_port(id, mpl::dma_0, mpl::dma_last,
00218                                  "The receiver DMA port is out of range")));
00219   }
00220
00221 /** Get access to a transmit DMA
00222
00223 \param[in] id specifies which DMA to access */
00224   auto& tx_dma(int id) { return *tx_dmas.at(id); }
00225
00226 /** Get the input router port of the AXI stream switch
00227
00228 \param p is the slave_port_layout for the stream
00229 */
00230   auto& input(spl p) {
00231     // No index validation required because of type safety
00232     return axi_ss.input(p);
00233   }
00234
00235 /** Get the output router port of the AXI stream switch
00236
00237 \param p is the master_port_layout for the stream
00238 */
00239   auto& output(mpl p) {
00240     // No index validation required because of type safety
00241     return axi_ss.output(p);
00242   }
00243
00244 /// Launch an invocable on this tile
00245   template <typename Work> void single_task(Work&& f) {
00246     if (future_work.valid())
00247       throw std::logic_error("Something is already running on this tile!");
00248     // Launch the tile program immediately on a new executor engine
00249 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00250     future_work = fe->submit(std::forward<Work>(f));
00251 #else
00252     future_work = std::async(std::launch::async, std::forward<Work>(f));
00253 #endif
00254   }
00255
00256 /// Wait for the execution of the callable on this tile
00257   void wait() { if (future_work.valid())
00258       future_work.get();
00259   }
00260
00261 /// Configure a connection of the core tile AXI stream switch
00262   void connect(typename geo::core_axi_stream_switch::slave_port_layout sp,
00263                typename geo::core_axi_stream_switch::master_port_layout mp) {
00264     axi_ss.connect(sp, mp);
00265   }
00266
00267 /// Compute the size of the graphics representation of the processor
00268   static vec<int, 2> display_core_size() {
00269     // This is the minimum rectangle fitting all the processor outputs & inputs
00270     return { 1 + ranges::distance(axi_ss_geo::m_me_range),
00271              1 + ranges::distance(axi_ss_geo::s_me_range) };
00272   }
00273
00274 /// Compute the size of the graphics representation of the tile
00275   static vec<int, 2> display_size() {
00276     // Just the sum of the size of its content
00277     return display_core_size() + axi_ss_t::display_size();
00278   }
00279
00280 /// Display the tile to a LaTeX context
00281   void display(latex::context& c)const {
00282     auto get_tikz_coordinate = [&](auto x, auto y) {
00283       auto const [x_size, y_size] = display_size();
00284       return (boost::format { "(%1%,%2%)" }
00285               // Scale real LaTeX coordinate to fit
00286               % c.scale(x_coordinate * x_size + x) %
00287               c.scale(y_coordinate * y_size + y))
00288           .str();
00289     };
00290     c.add((boost::format { "  \\begin{scope}[name prefix = TileX%1%Y%2%]" } %
00291            x_coordinate % y_coordinate)
00292               .str());
00293     axi_ss.display(c, display_core_size(), get_tikz_coordinate);
00294
00295     // Connect the core receivers to its AXI stream switch
00296     for (auto [i, p] :  axi_ss_geo::m_me_range | ranges::views::enumerate) {
```

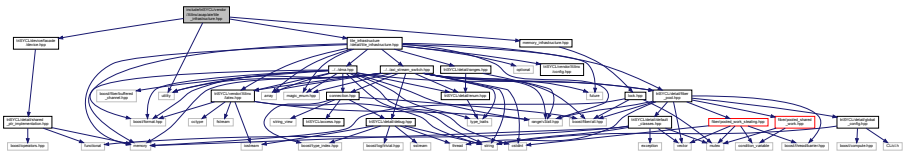
```
00297       c.add(
00298           (boost::format { R"(
00299 \coordinate(CoreIn%1%) at %2%;
00300 \node[rotate=90,anchor=east](CoreIn%1%Label) at %2% {in(%1%)};
00301 \draw (node cs:name=MMe%1%)
00302 -| (node cs:name=CoreIn%1%);)" } %
00303 i %
00304 get_tikz_coordinate(i, ranges::distance(axi_ss_geo::m_me_range) + 1))
00305 .str());
00306 };
00307 // Connect the core senders to its AXI stream switch
00308     for (auto [i, p] :  axi_ss_geo::s_me_range | ranges::views::enumerate) {
00309       c.add(
00310           (boost::format { R"(
00311 \coordinate(CoreOut%1%) at %2%;
00312 \node[anchor=east](CoreOut%1%Label) at %2%  {out(%1%)};
00313 \draw (node cs:name=CoreOut%1%)
00314 -| (node cs:name=SMe%1%);)" } %
00315 i %
00316 get_tikz_coordinate(ranges::distance(axi_ss_geo::s_me_range), i + 1))
00317 .str());
00318 };
00319 c.add((boost::format { R"(
00320 \node[black] () at %1% {\texttt{tile<%2%,%3%>}};
00321 \begin{scope}[on background layer]
00322 \node [fill=orange!30, fit={(node cs:name=CoreIn0Label)
00323 (node cs:name=CoreOut0Label)}]
00324 (Core) {};
00325 \end{scope}
00326 \end{scope}
00327
00328 )" } % get_tikz_coordinate(1, 0) %
00329 x_coordinate % y_coordinate)
00330 .str());
00331 }
00332 };
00333
00334 /// @} End the aie Doxygen group
00335
00336 } // namespace trisycl::vendor::xilinx::acap::aie::detail
00337
00338 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_INFRASTRUCTURE_DETAIL_TILE_INFRASTRUCTURE_HPP
```

13.381 include/triSYCL/vendor/Xilinx/acap/aie/tile_infrastructure.hpp File Reference

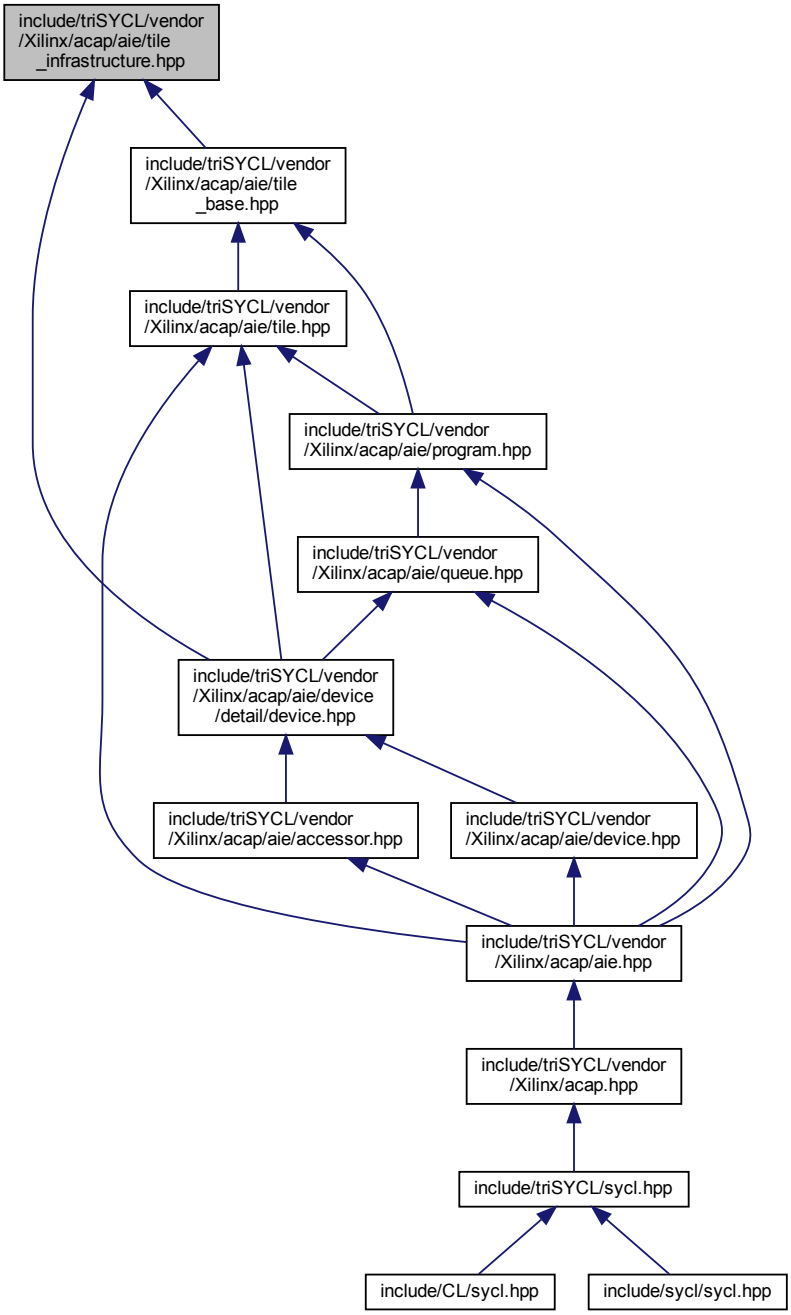
The basic AI Engine homogeneous tile, with the common infrastructure to all the tiles, i.e.

```
#include <utility>
#include "memory_infrastructure.hpp"
#include "tile_infrastructure/detail/tile_infrastructure.hpp"
#include "triSYCL/device/facade/device.hpp"
```

Include dependency graph for tile_infrastructure.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >

  The AI Engine tile infrastructure common to all the tiles. More...
- struct trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie

13.381.1 Detailed Description

The basic AI Engine homogeneous tile, with the common infrastructure to all the tiles, i.e.

independent of x & y coordinates, but also from the tile program itself.

This tile can be seen as the SYCL subdevice of the raw CGRA SYCL device to run elemental functions.

This is owned by the device, so for example the AXI stream switch configuration and packet can survive to some program changes.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file tile_infrastructure.hpp.

13.382 tile_infrastructure.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_INFRASTRUCTURE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_INFRASTRUCTURE_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine homogeneous tile, with the common
00007 infrastructure to all the tiles, i.e.  independent of x & y
00008 coordinates, but also from the tile program itself.
00009
00010 This tile can be seen as the SYCL subdevice of the raw CGRA SYCL
00011 device to run elemental functions.
00012
00013 This is owned by the device, so for example the AXI stream switch
00014 configuration and packet can survive to some program changes.
00015
00016 Ronan dot Keryell at Xilinx dot com
00017
00018 This file is distributed under the University of Illinois Open Source
00019 License.  See LICENSE.TXT for details.
00020 */
00021
00022 #include <utility>
00023
00024 #include "memory_infrastructure.hpp"
00025 #include "tile_infrastructure/detail/tile_infrastructure.hpp"
00026 #include "triSYCL/device/facade/device.hpp"
00027
00028 namespace trisycl::vendor::xilinx::acap::aie {
00029
00030 /// \ingroup aie
00031 /// @{
00032
00033 /** The AI Engine tile infrastructure common to all the tiles
00034
00035 This allows some type erasure while accessing the common
00036 tile infrastructure.
00037
```

```
00038 \param Geography is the \c geography type representing the full
00039 layout of the architecture
00040 */
00041 template <typename Geography>
00042 class tile_infrastructure
00043     :  public facade::device<tile_infrastructure<Geography>,
00044                              detail::tile_infrastructure<Geography>> {
00045 /// The type encapsulating the implementation
00046   using dti = detail::tile_infrastructure<Geography>;
00047
00048  public:
00049 /// The façade used to implement part of the user-facing type
00050   using facade_t = facade::device<tile_infrastructure<Geography>, dti>;
00051
00052   using geo = Geography;
00053   using axi_ss_geo = typename dti::axi_ss_geo;
00054   using mpl = typename dti::mpl;
00055   using spl = typename dti::spl;
00056   using axi_ss_t = typename dti::axi_ss_t;
00057
00058 /// Make the implementation member directly accessible in this class
00059   using facade_t::implementation;
00060
00061 /** Start the tile infrastructure associated to the AIE device
00062
00063 \param[in] x is the horizontal coordinate for this tile
00064
00065 \param[in] y is the vertical coordinate for this tile
00066
00067 \param[in] dev is the aie::detail::device used to control
00068 hardware when using real hardware and provide some debug
00069 information from inside the tile_infrastructure.  Use auto
00070 concept here to avoid explicit type causing circular dependency
00071
00072 \param[in] fiber_executor is the executor used to run
00073 infrastructure details
00074 */
00075   tile_infrastructure(int x, int y, auto& dev,
00076                       ::trisycl::detail::fiber_pool& fiber_executor)
00077       :  facade_t { std::make_shared<dti>(x, y, dev, fiber_executor) } {}
00078   tile_infrastructure() = default;
00079
00080  // \todo Implement a real tile handle instead of using x(), y(),
00081   // etc.  from here
00082
00083 /// Get the horizontal coordinate
00084   int x() { return implementation->x(); }
00085
00086 /// Get the vertical coordinate
00087   int y() { return implementation->y(); }
00088
00089 /// Get the horizontal number of tiles
00090   static constexpr int x_size() { return geo::x_size; }
00091
00092 /// Get the vertical number of tiles
00093   static constexpr int y_size() { return geo::y_size; }
00094
00095 /// Access to the common infrastructure part of a tile memory
00096   auto& mem() {
00097     return implementation->mem();
00098   }
00099
00100   struct cgh_t {
00101     tile_infrastructure ti;
00102     cgh_t(tile_infrastructure t) :  ti { t } {}
00103
00104   template <typename Invocable>
00105     void single_task(Invocable&& t) {
00106       ti.single_task([=] { t(); });
00107     }
00108   };
00109
00110 /// Submit a command group to the tile
00111   template <typename Invocable>
00112   void submit(Invocable&& f)const {
00113     cgh_t cgh { *this };
00114     std::forward<Invocable>(f)(cgh);
00115   }
00116
00117 /** Get the user input connection from the AXI stream switch
00118
00119 \param[in] port is the port to use
00120 */
00121   auto& in_connection(int port) { return implementation->in_connection(port); }
00122
00123 /** Get the user output connection to the AXI stream switch
00124
```

```
00125 \param[in] port is port to use
00126 */
00127   auto& out_connection(int port) {
00128     return implementation->out_connection(port);
00129   }
00130
00131 /** Get the user input port from the AXI stream switch
00132
00133 \param[in] port is the port to use
00134 */
00135   auto& in(int port) { return implementation->in(port); }
00136
00137 /** Get the user output port to the AXI stream switch
00138
00139 \param[in] port is the port to use
00140 */
00141   auto& out(int port) { return implementation->out(port); }
00142
00143 /** Get access to a receiver DMA
00144
00145 \param[in] id specifies which DMA to access */
00146   auto& rx_dma(int id) { return implementation->rx_dma(id); }
00147
00148 /** Get access to a transmit DMA
00149
00150 \param[in] id specifies which DMA to access */
00151   auto& tx_dma(int id) { return implementation->tx_dma(id); }
00152
00153 /** Get the input router port of the AXI stream switch
00154
00155 \param p is the slave_port_layout for the stream
00156 */
00157   auto& input(spl p) { return implementation->input(p); }
00158
00159 /** Get the output router port of the AXI stream switch
00160
00161 \param p is the master_port_layout for the stream
00162 */
00163   auto& output(mpl p) { return implementation->output(p); }
00164
00165 /// Launch a callable on this tile
00166   template <typename Work> auto& single_task(Work&& f) {
00167     /* Add an immediate lambda call to avoid a warning about capturing
00168 this when non using it */
00169     auto kernel = [&] {
00170       if constexpr (requires { f(); })
00171         /* If the invocable is not interested by the handler, do not
00172 provide it.  Use a copy of the callable to make the
00173 separation from host to device memory very clear.  */
00174         return [work = f] () mutable { return work(); };
00175       else
00176 /** Recycle the tile_infrastructure shared_ptr as a fake tile
00177 handle which is copyable.
00178
00179 \todo In a device implementation we should have a real
00180 tile_handler type making sense on the device instead of
00181 just *this */
00182         return [this, work = f] () mutable { return work(*this); };
00183     }();
00184
00185     implementation->single_task(kernel);
00186     // To allow chaining commands
00187     return *this;
00188   }
00189
00190 /// Wait for the execution of the callable on this tile
00191   auto& wait() {
00192     implementation->wait();
00193     // To allow chaining commands
00194     return *this;
00195   }
00196
00197 /// Configure a connection of the core tile AXI stream switch
00198   auto& connect(typename geo::core_axi_stream_switch::slave_port_layout sp,
00199                 typename geo::core_axi_stream_switch::master_port_layout mp) {
00200     implementation->connect(sp, mp);
00201     // To allow chaining commands
00202     return *this;
00203   }
00204
00205 /// Compute the size of the graphics representation of the tile
00206   static vec<int, 2> display_size() { return dti::display_size(); }
00207
00208 /// Display the tile to a LaTeX context
00209   void display(latex::context& c)const { implementation->display(c); }
00210 };
00211
```

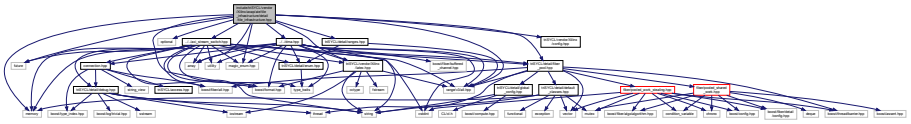
```
00212 /// @} End the aie Doxygen group
00213
00214 } // namespace trisycl::vendor::xilinx::acap::aie
00215
00216 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_INFRASTRUCTURE_HPP
```

13.383 include/triSYCL/vendor/Xilinx/acap/aie/tile_↩ infrastructure/detail/tile_infrastructure.hpp File Reference

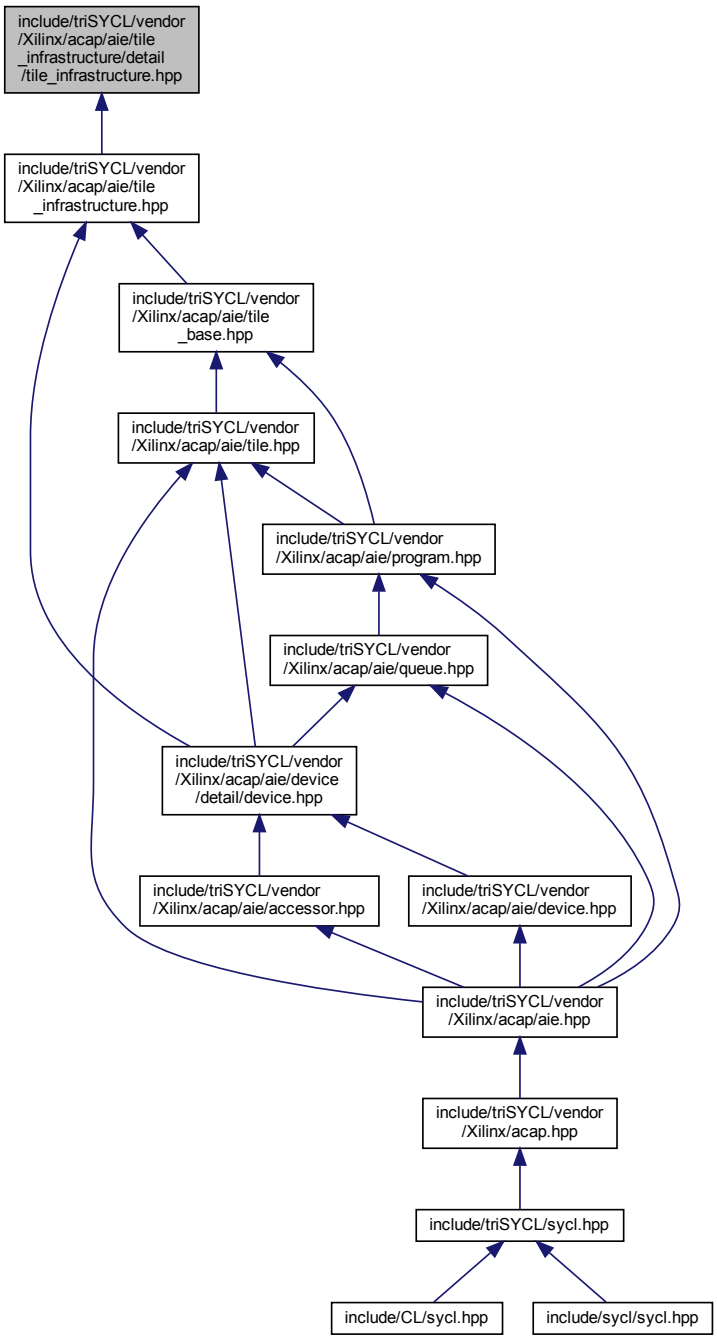
The basic AI Engine homogeneous tile, with the common infrastructure to all the tiles, i.e.

```
#include <array>
#include <future>
#include <memory>
#include <optional>
#include "magic_enum.hpp"
#include <boost/format.hpp>
#include <range/v3/all.hpp>
#include "../../axi_stream_switch.hpp"
#include "../../dma.hpp"
#include "triSYCL/detail/fiber_pool.hpp"
#include "triSYCL/detail/ranges.hpp"
#include "triSYCL/vendor/Xilinx/config.hpp"
#include "triSYCL/vendor/Xilinx/latex.hpp"
```

Include dependency graph for tile_infrastructure.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >

  The AI Engine tile infrastructure common to all the tiles. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::aie
- namespace trisycl::vendor::xilinx::acap::aie::detail

13.383.1 Detailed Description

The basic AI Engine homogeneous tile, with the common infrastructure to all the tiles, i.e.

independent of x & y coordinates, but also from the tile program itself.

This tile can be seen as the raw CGRA subdevice to run elemental functions.

This is owned by the device, so for example the AXI stream switch configuration and packet can survive to some program changes.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file tile_infrastructure.hpp.

13.384 tile_infrastructure.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_INFRASTRUCTURE_DETAIL_TILE_INFRASTRUCTURE_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_INFRASTRUCTURE_DETAIL_TILE_INFRASTRUCTURE_HPP
00003
00004 /** \file
00005
00006 The basic AI Engine homogeneous tile, with the common
00007 infrastructure to all the tiles, i.e.  independent of x & y
00008 coordinates, but also from the tile program itself.
00009
00010 This tile can be seen as the raw CGRA subdevice to run elemental
00011 functions.
00012
00013 This is owned by the device, so for example the AXI stream switch
00014 configuration and packet can survive to some program changes.
00015
00016 Ronan dot Keryell at Xilinx dot com
00017
00018 This file is distributed under the University of Illinois Open Source
00019 License.  See LICENSE.TXT for details.
00020 */
00021
00022 #include <array>
00023 #include <future>
00024 #include <memory>
00025 #include <optional>
00026
00027 #include "magic_enum.hpp"
00028 #include <boost/format.hpp>
00029 #include <range/v3/all.hpp>
00030
00031 #include "../../axi_stream_switch.hpp"
00032 #include "../../dma.hpp"
00033 #include "triSYCL/detail/fiber_pool.hpp"
00034 #include "triSYCL/detail/ranges.hpp"
00035 #include "triSYCL/vendor/Xilinx/config.hpp"
```

```
00036 #include "triSYCL/vendor/Xilinx/latex.hpp"
00037
00038 namespace trisycl::vendor::xilinx::acap::aie::detail {
00039
00040 /// \ingroup aie
00041 /// @{
00042
00043 /** The AI Engine tile infrastructure common to all the tiles
00044
00045 This allows some type erasure while accessing the common
00046 tile infrastructure.
00047
00048 \param AIE is the type representing the full CGRA with the
00049 programs and memory contents
00050 */
00051 template <typename Geography> class tile_infrastructure {
00052
00053  public:
00054   using geo = Geography;
00055   using axi_ss_geo = typename geo::core_axi_stream_switch;
00056   using mpl = typename axi_ss_geo::master_port_layout;
00057   using spl = typename axi_ss_geo::slave_port_layout;
00058   using axi_ss_t = axi_stream_switch<axi_ss_geo>;
00059
00060  private:
00061 /// Keep the horizontal coordinate
00062   int x_coordinate;
00063
00064 /// Keep the vertical coordinate
00065   int y_coordinate;
00066
00067 /** Keep track of the aie::detail::device for hardware resource
00068 control in device mode or for debugging purpose for better
00069 messages.
00070
00071 Use void* for now to avoid cyclic header dependencies for now
00072 instead of the aie::detail::device */
00073   void* dev [[maybe_unused]];
00074
00075 /// The AXI stream switch of the core tile
00076   axi_ss_t axi_ss;
00077
00078 /** Keep track of all the infrastructure tile memories
00079 of this device */
00080   aie::memory_infrastructure mi;
00081
00082 /** Sending DMAs
00083
00084 Use std::optional to postpone initialization */
00085   std::array<std::optional<sending_dma<axi_ss_t>, axi_ss_geo::s_dma_size>
00086       tx_dmas;
00087
00088 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00089 /// Keep track of the fiber executor
00090   ::trisycl::detail::fiber_pool* fe;
00091
00092 /// To shepherd the working fiber
00093   ::trisycl::detail::fiber_pool::future<void> future_work;
00094 #else
00095 /// Keep track of the std::thread execution in this tile
00096   std::future<void> future_work;
00097 #endif
00098
00099 /** Map the user input port number to the AXI stream switch port
00100
00101 \param[in] port is the user port to use
00102 */
00103   static auto translate_input_port(int port) {
00104     return axi_ss_t::translate_port(port, spl::me_0, spl::me_last,
00105                                     "The core input port is out of range");
00106   }
00107
00108 /** Map the user output port number to the AXI stream switch port
00109
00110 \param[in] port is the user port to use
00111 */
00112   static auto translate_output_port(int port) {
00113     return axi_ss_t::translate_port(port, mpl::me_0, mpl::me_last,
00114                                     "The core output port is out of range");
00115   }
00116
00117  public:
00118 /** Start the tile infrastructure associated to the AIE device tile
00119
00120 \param[in] x is the horizontal coordinate for this tile
00121
00122 \param[in] y is the vertical coordinate for this tile
```

```
00123
00124 \param[in] dev is the aie::detail::device used to control
00125 hardware when using real hardware and provide some debug
00126 information from inside the tile_infrastructure.  Use auto
00127 concept here to avoid explicit type causing circular dependency
00128
00129 \param[in] fiber_executor is the executor used to run
00130 infrastructure details
00131 */
00132   tile_infrastructure(int x, int y, auto& dev,
00133                       ::trisycl::detail::fiber_pool& fiber_executor)
00134       :  x_coordinate { x }
00135       , y_coordinate { y }
00136       , dev { &dev }
00137       , mi { dev }
00138 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00139       , fe { &fiber_executor }
00140 #endif
00141   {
00142     // Connect the core receivers to its AXI stream switch
00143     for (auto p :  views::enum_type(mpl::me_0, mpl::me_last))
00144       output(p) =
00145           std::make_shared<port_receiver<axi_ss_t»(axi_ss, "core_receiver");
00146     axi_ss.start(x, y, fiber_executor);
00147     /* Create the core tile receiver DMAs and make them directly the
00148 switch output ports */
00149     for (auto p :  axi_ss_geo::m_dma_range)
00150       output(p) =
00151           std::make_shared<receiving_dma<axi_ss_t»(axi_ss, fiber_executor);
00152     /* Create the core tile sender DMAs and connect them internally to
00153 their switch input ports */
00154     for (const auto& [d, p] :
00155          ranges::views::zip(tx_dmas, axi_ss_geo::s_dma_range))
00156       d.emplace(fiber_executor, input(p));
00157   }
00158
00159 /// Get the horizontal coordinate
00160   int x() { return x_coordinate; }
00161
00162 /// Get the vertical coordinate
00163   int y() { return y_coordinate; }
00164
00165 /// Access to the common infrastructure part of a tile memory
00166   auto& mem() {
00167     return mi;
00168   }
00169
00170 /** Get the user input connection from the AXI stream switch
00171
00172 \param[in] port is the port to use
00173 */
00174   auto& in_connection(int port) {
00175     /* The input port for the core is actually the corresponding
00176 output on the switch */
00177     return axi_ss.out_connection(translate_output_port(port));
00178   }
00179
00180 /** Get the user output connection to the AXI stream switch
00181
00182 \param[in] port is port to use
00183 */
00184   auto& out_connection(int port) {
00185     /* The output port for the core is actually the corresponding
00186 input on the switch */
00187     return axi_ss.in_connection(translate_input_port(port));
00188   }
00189
00190 /** Get the user input port from the AXI stream switch
00191
00192 \param[in] port is the port to use
00193 */
00194   auto& in(int port) {
00195     TRISYCL_DUMP_T("in(" « port « ") on tile(" « x_coordinate « ',’
00196                    « y_coordinate « ')’);
00197     return *in_connection(port);
00198   }
00199
00200 /** Get the user output port to the AXI stream switch
00201
00202 \param[in] port is the port to use
00203 */
00204   auto& out(int port) {
00205     TRISYCL_DUMP_T("out(" « port « ") on tile(" « x_coordinate « ',’
00206                    « y_coordinate « ')’);
00207     return *out_connection(port);
00208   }
00209
```

```
00210 /** Get access to a receiver DMA
00211
00212 \param[in] id specifies which DMA to access */
00213   auto& rx_dma(int id) {
00214 /** The output of the switch is actually a receiving DMA, so we
00215 can view it as a DMA */
00216     return static_cast<receiving_dma<axi_ss_t>&>(*output(
00217         axi_ss_t::translate_port(id, mpl::dma_0, mpl::dma_last,
00218                                  "The receiver DMA port is out of range")));
00219   }
00220
00221 /** Get access to a transmit DMA
00222
00223 \param[in] id specifies which DMA to access */
00224   auto& tx_dma(int id) { return *tx_dmas.at(id); }
00225
00226 /** Get the input router port of the AXI stream switch
00227
00228 \param p is the slave_port_layout for the stream
00229 */
00230   auto& input(spl p) {
00231     // No index validation required because of type safety
00232     return axi_ss.input(p);
00233   }
00234
00235 /** Get the output router port of the AXI stream switch
00236
00237 \param p is the master_port_layout for the stream
00238 */
00239   auto& output(mpl p) {
00240     // No index validation required because of type safety
00241     return axi_ss.output(p);
00242   }
00243
00244 /// Launch an invocable on this tile
00245   template <typename Work> void single_task(Work&& f) {
00246     if (future_work.valid())
00247       throw std::logic_error("Something is already running on this tile!");
00248     // Launch the tile program immediately on a new executor engine
00249 #if TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00250     future_work = fe->submit(std::forward<Work>(f));
00251 #else
00252     future_work = std::async(std::launch::async, std::forward<Work>(f));
00253 #endif
00254   }
00255
00256 /// Wait for the execution of the callable on this tile
00257   void wait() { if (future_work.valid())
00258       future_work.get();
00259   }
00260
00261 /// Configure a connection of the core tile AXI stream switch
00262   void connect(typename geo::core_axi_stream_switch::slave_port_layout sp,
00263                typename geo::core_axi_stream_switch::master_port_layout mp) {
00264     axi_ss.connect(sp, mp);
00265   }
00266
00267 /// Compute the size of the graphics representation of the processor
00268   static vec<int, 2> display_core_size() {
00269     // This is the minimum rectangle fitting all the processor outputs & inputs
00270     return { 1 + ranges::distance(axi_ss_geo::m_me_range),
00271              1 + ranges::distance(axi_ss_geo::s_me_range) };
00272   }
00273
00274 /// Compute the size of the graphics representation of the tile
00275   static vec<int, 2> display_size() {
00276     // Just the sum of the size of its content
00277     return display_core_size() + axi_ss_t::display_size();
00278   }
00279
00280 /// Display the tile to a LaTeX context
00281   void display(latex::context& c)const {
00282     auto get_tikz_coordinate = [&](auto x, auto y) {
00283       auto const [x_size, y_size] = display_size();
00284       return (boost::format { "(%1%,%2%)" }
00285               // Scale real LaTeX coordinate to fit
00286               % c.scale(x_coordinate * x_size + x) %
00287               c.scale(y_coordinate * y_size + y))
00288           .str();
00289     };
00290     c.add((boost::format { "  \\begin{scope}[name prefix = TileX%1%Y%2%]" } %
00291            x_coordinate % y_coordinate)
00292               .str());
00293     axi_ss.display(c, display_core_size(), get_tikz_coordinate);
00294
00295     // Connect the core receivers to its AXI stream switch
00296     for (auto [i, p] :  axi_ss_geo::m_me_range | ranges::views::enumerate) {
```

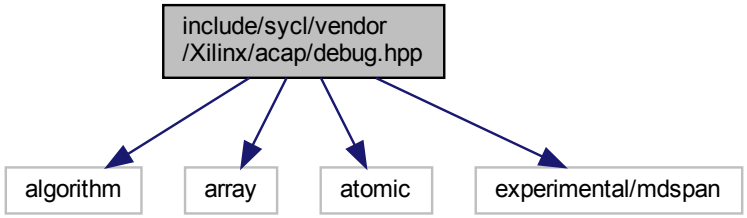
```
00297       c.add(
00298           (boost::format { R"(
00299 \coordinate(CoreIn%1%) at %2%;
00300 \node[rotate=90,anchor=east](CoreIn%1%Label) at %2% {in(%1%)};
00301 \draw (node cs:name=MMe%1%)
00302 -| (node cs:name=CoreIn%1%);)" } %
00303 i %
00304 get_tikz_coordinate(i, ranges::distance(axi_ss_geo::m_me_range) + 1))
00305 .str());
00306 };
00307 // Connect the core senders to its AXI stream switch
00308     for (auto [i, p] :  axi_ss_geo::s_me_range | ranges::views::enumerate) {
00309       c.add(
00310           (boost::format { R"(
00311 \coordinate(CoreOut%1%) at %2%;
00312 \node[anchor=east](CoreOut%1%Label) at %2%  {out(%1%)};
00313 \draw (node cs:name=CoreOut%1%)
00314 -| (node cs:name=SMe%1%);)" } %
00315 i %
00316 get_tikz_coordinate(ranges::distance(axi_ss_geo::s_me_range), i + 1))
00317 .str());
00318 };
00319 c.add((boost::format { R"(
00320 \node[black] () at %1% {\texttt{tile<%2%,%3%>}};
00321 \begin{scope}[on background layer]
00322 \node [fill=orange!30, fit={(node cs:name=CoreIn0Label)
00323 (node cs:name=CoreOut0Label)}]
00324 (Core) {};
00325 \end{scope}
00326 \end{scope}
00327
00328 )" } % get_tikz_coordinate(1, 0) %
00329 x_coordinate % y_coordinate)
00330 .str());
00331 }
00332 };
00333
00334 /// @} End the aie Doxygen group
00335
00336 } // namespace trisycl::vendor::xilinx::acap::aie::detail
00337
00338 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_AIE_TILE_INFRASTRUCTURE_DETAIL_TILE_INFRASTRUCTURE_HPP
```

13.385 include/sycl/vendor/Xilinx/acap/debug.hpp File Reference

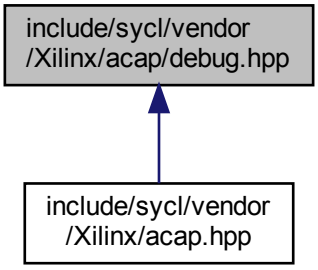
Some debugging utilities for ACAP.

```
#include <algorithm>
#include <array>
#include <atomic>
#include <experimental/mdspan>
```

Include dependency graph for debug.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >

  Verify that an SPMD program has a bulk-synchronous behaviour across all the tiles of the CGRA. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::debug

13.385.1 Detailed Description

Some debugging utilities for ACAP.

Definition in file debug.hpp.

13.386 debug.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_DEBUG_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_DEBUG_HPP
00003
00004 /** \file
00005
00006 Some debugging utilities for ACAP
00007 */
00008
00009 #include <algorithm>
00010 #include <array>
00011 #include <atomic>
00012
00013 #include <experimental/mdspan>
```

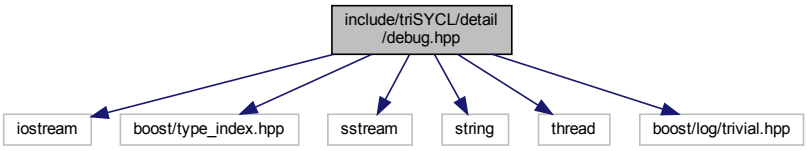
```
00014
00015 /** \defgroup acap Extensions for Xilinx ACAP Versal
00016
00017 Some classes and functions to support devices including CPU, FPGA
00018 & AI Engine CGRA
00019 @{
00020 */
00021
00022 /** \defgroup acap_debug Debugging support for ACAP
00023
00024 Some classes and functions to help debuging ACAP applications,
00025 such as verifying BSP forward progress
00026 @{
00027 */
00028
00029 namespace trisycl::vendor::xilinx::acap::debug {
00030
00031 /// @}
00032 /// \ingroup acap_debug
00033 /// @{
00034
00035 /** Verify that an SPMD program has a bulk-synchronous behaviour
00036 across all the tiles of the CGRA.
00037
00038 If there are some discrepancies, display the tile id which is too
00039 late or too early.
00040 */
00041 template <typename Geography> class bsp_checker {
00042 /// A vector clock to compare differerent tiles together
00043   std::array<std::atomic<int>, Geography::size> vector_clock_alloc;
00044   std::mdspan<std::atomic<int>,
00045               std::extents<std::size_t, Geography::x_size, Geography::y_size»
00046       vector_clock { &vector_clock_alloc[0] };
00047
00048  public:
00049 /** Check that none of the tiles executing this are separated by
00050 more than one execution step each other.
00051
00052 This functions has to be called in a BSP synchronous way by all
00053 the tiles.
00054 */
00055   void check(int x, int y) {
00056     ++vector_clock[x, y];
00057     /* While the algorithm execution is not atomic, it is enough to
00058 find some asynchronous bugs */
00059     auto [min_element, max_element] = std::minmax_element(
00060         vector_clock_alloc.cbegin(), vector_clock_alloc.cend());
00061
00062     if (*max_element - *min_element > 1) {
00063       TRISYCL_DUMP_T(std::dec « "compute(" « x « ',' « y
00064                               « ") vector clock min = " « *min_element
00065                               « ", max = " « *max_element);
00066       std::terminate();
00067     }
00068   }
00069 };
00070
00071 } // namespace trisycl::vendor::xilinx::acap::debug
00072
00073 /// @} End the acap_debug Doxygen group
00074
00075 /// @} End the acap Doxygen group
00076
00077 /*
00078 # Some Emacs stuff:
00079 ### Local Variables:
00080 ### ispell-local-dictionary:  "american"
00081 ### eval:  (flyspell-prog-mode)
00082 ### End:
00083 */
00084
00085 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_DEBUG_HPP
```

13.387 include/triSYCL/detail/debug.hpp File Reference

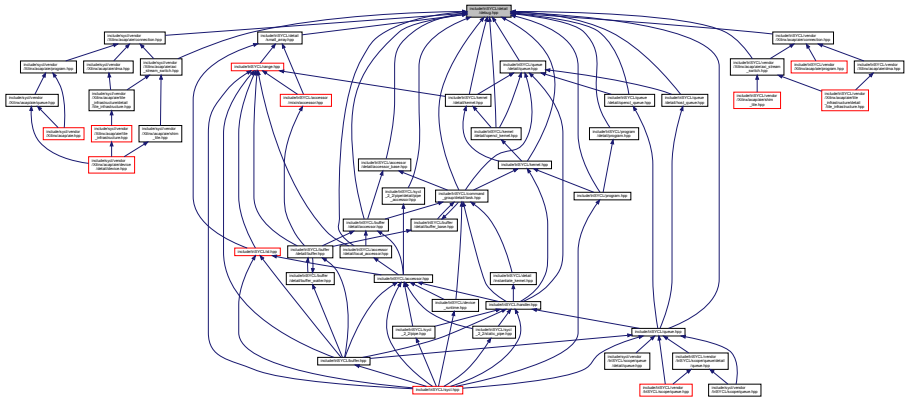
```
#include <iostream>
#include <boost/type_index.hpp>
#include <sstream>
#include <string>
```

```
#include <thread>
#include <boost/log/trivial.hpp>
```

Include dependency graph for debug.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::detail::debug< T >

  Class used to trace the construction, copy-construction, move-construction and destruction of classes that inherit from it. More...
- struct trisycl::detail::display_vector< T >

  Class used to display a vector-like type of classes that inherit from it. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::detail

  Some global triSYCL configuration.

Macros

- #define TRISYCL_INTERNAL_DUMP(expression)

  Dump a debug message in a formatted way.
- #define TRISYCL_DUMP(expression) TRISYCL_INTERNAL_DUMP(expression)
- #define TRISYCL_DUMP_T(expression)

  Same as TRISYCL_DUMP() but with thread id first.

Functions

- template<typename KernelName , typename Functor >
  auto trisycl::detail::trace_kernel (Functor f)

  Wrap a kernel functor in some tracing messages to have start/stop information when TRISYCL_TRACE_KERNEL macro is defined.

13.387.1 Macro Definition Documentation

13.387.1.1 TRISYCL_DUMP

```
#define TRISYCL_DUMP(
              expression ) TRISYCL_INTERNAL_DUMP(expression)
```

Definition at line 44 of file debug.hpp.

13.387.1.2 TRISYCL_DUMP_T

```
#define TRISYCL_DUMP_T(
              expression )
```

Value:

```
  TRISYCL_DUMP("Thread " « std::hex                                    \
               « std::this_thread::get_id() « ":  " « expression)
```

Same as TRISYCL_DUMP() but with thread id first.

Definition at line 47 of file debug.hpp.

13.387.1.3 TRISYCL_INTERNAL_DUMP

```
#define TRISYCL_INTERNAL_DUMP(
              expression )
```

Value:

```
    do {       \
    std::ostringstream s;                            \
    s « expression;                                  \
    BOOST_LOG_TRIVIAL(debug) « s.str();              \
  } while(0)
```

Dump a debug message in a formatted way.

Use an intermediate ostringstream because there are issues with BOOST_LOG_TRIVIAL to display C strings

Definition at line 36 of file debug.hpp.

13.388 debug.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_DETAIL_DEBUG_HPP
00002 #define TRISYCL_SYCL_DETAIL_DEBUG_HPP
00003
00004 /** \file Track constructor/destructor invocations and trace kernel execution
00005
00006 Define the TRISYCL_DEBUG CPP flag to have an output.
00007
00008 To use it in some class C, make C inherit from debug<C>.
00009
00010 Ronan at Keryell point FR
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include <iostream>
00017
00018 #include <boost/type_index.hpp>
00019
00020 // Only when the common debug or trace infrastructure is required
00021 #if defined(TRISYCL_DEBUG) || defined(TRISYCL_TRACE_KERNEL)
00022 #include <sstream>
00023 #include <string>
00024 #include <thread>
00025
00026 #include <boost/log/trivial.hpp>
00027
00028 // To be able to construct string literals like "blah"s
00029 using namespace std::string_literals;
00030
00031 /** Dump a debug message in a formatted way.
00032
00033 Use an intermediate ostringstream because there are issues with
00034 BOOST_LOG_TRIVIAL to display C strings
00035 */
00036 #define TRISYCL_INTERNAL_DUMP(expression) do {       \
00037 std::ostringstream s;                                \
00038 s « expression;                                      \
00039 BOOST_LOG_TRIVIAL(debug) « s.str();                  \
00040 } while(0)
00041 #endif
00042
00043 #ifdef TRISYCL_DEBUG
00044 #define TRISYCL_DUMP(expression) TRISYCL_INTERNAL_DUMP(expression)
00045
00046 /// Same as TRISYCL_DUMP() but with thread id first
00047 #define TRISYCL_DUMP_T(expression)                                    \
00048 TRISYCL_DUMP("Thread " « std::hex                                     \
00049              « std::this_thread::get_id() « ":  " « expression)
00050 #else
00051 #define TRISYCL_DUMP(expression) do { } while(0)
00052 #define TRISYCL_DUMP_T(expression) do { } while(0)
00053 #endif
00054
00055 namespace trisycl::detail {
00056
00057 /** \addtogroup debug_trace Debugging and tracing support
00058 @{
00059 */
00060
00061 /** Class used to trace the construction, copy-construction,
00062 move-construction and destruction of classes that inherit from it
00063
00064 Also trace the copy and move assignments.
00065
00066 \param T is the real type name to be used in the debug output.
00067 */
00068 template <typename T>
00069 struct debug {
00070 /// Get the pretty name of T itself
00071   static auto constexpr type_pretty_name() {
00072     return boost::typeindex::type_id<T>().pretty_name();
00073   }
00074
00075 #ifdef TRISYCL_DEBUG_STRUCTORS
00076
00077 /// Trace the construction
00078   debug() {
00079     TRISYCL_DUMP("Constructor of " « type_pretty_name() « " "
00080                  « static_cast<void*>(this));
00081   }
00082
```

```
00083 #if 0
00084 /// Since Clang++-11 this is no longer working, because it should not have
00085 /// never been working!  But is seems to work with G++
00086
00087 /// Trace the copy construction
00088   debug(debug const& old) {
00089     TRISYCL_DUMP("Copy of " « type_pretty_name() « " into "
00090                  « static_cast<void*>(this) « " from "
00091                  « static_cast<const void*>(&old));
00092   }
00093
00094
00095 /// Trace the copy assignment
00096   debug& operator=(const debug& rhs) {
00097     TRISYCL_DUMP("Copy assignment of " « type_pretty_name()
00098                  « " into " « static_cast<void*>(this)
00099                  « " from " « static_cast<const void*>(&rhs));
00100     return *this;
00101   }
00102
00103
00104 /// Trace the move construction
00105   debug(debug&& old) {
00106     TRISYCL_DUMP("Move of " « type_pretty_name() « " into "
00107                  « static_cast<void*>(this) « " from "
00108                  « static_cast<void*>(&old));
00109   }
00110
00111
00112 /// Trace the move assignment
00113   debug& operator=(debug&& rhs) {
00114     TRISYCL_DUMP("Move assignment of " « type_pretty_name()
00115                  « " into " « static_cast<void*>(this)
00116                  « " from " « static_cast<void*>(&rhs));
00117     return *this;
00118   }
00119 #endif
00120
00121 /// Trace the destruction
00122   ~debug() {
00123     TRISYCL_DUMP("~ Destructor of " « type_pretty_name() « " "
00124                  « static_cast<void*>(this));
00125   }
00126 #endif
00127 };
00128
00129
00130 /** Wrap a kernel functor in some tracing messages to have start/stop
00131 information when TRISYCL_TRACE_KERNEL macro is defined */
00132 template <typename KernelName, typename Functor>
00133 auto trace_kernel(Functor f) {
00134 #ifdef TRISYCL_TRACE_KERNEL
00135   // Inject tracing message around the kernel
00136   return [=] () mutable {
00137     /* Since the class KernelName may just be declared and not really
00138 defined, just use it through a class pointer to have
00139 typeid().name() not complaining */
00140     TRISYCL_INTERNAL_DUMP(
00141       "Kernel started "
00142       « boost::typeindex::type_id<KernelName *>().pretty_name());
00143     f();
00144     TRISYCL_INTERNAL_DUMP(
00145       "Kernel stopped "
00146       « boost::typeindex::type_id<KernelName *>().pretty_name());
00147   };
00148 #else
00149   // Identity by default
00150   return f;
00151 #endif
00152 }
00153
00154
00155 /** Class used to display a vector-like type of classes that inherit from
00156 it
00157
00158 \param T is the real type name to be used in the debug output.
00159
00160 Calling the display() method dump the values on std::cout
00161 */
00162 template <typename T>
00163 struct display_vector {
00164
00165 /// To debug and test
00166   void display()const {
00167 #ifdef TRISYCL_DEBUG
00168     std::cout « boost::typeindex::type_id<T>().pretty_name() « ":";
00169 #endif
```

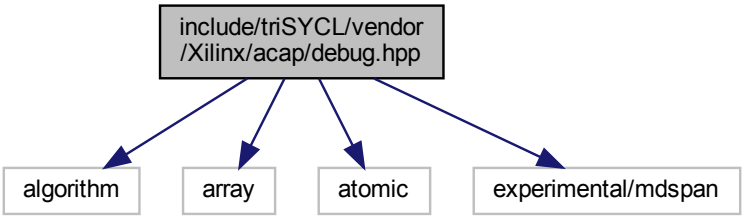
```
00170     // Get a pointer to the real object
00171     for (auto e :  static_cast<const T&>(*this))
00172       std::cout « " " « e;
00173     std::cout « std::endl;
00174   }
00175
00176 };
00177
00178 /// @} End the debug_trace Doxygen group
00179
00180 }
00181
00182 /*
00183 # Some Emacs stuff:
00184 ### Local Variables:
00185 ### ispell-local-dictionary:  "american"
00186 ### eval:  (flyspell-prog-mode)
00187 ### End:
00188 */
00189
00190 #endif // TRISYCL_SYCL_DETAIL_DEBUG_HPP
```

13.389 include/triSYCL/vendor/Xilinx/acap/debug.hpp File Reference

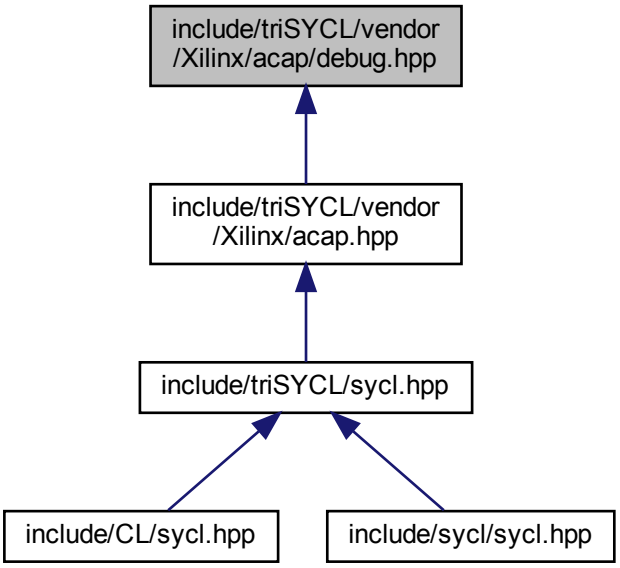
Some debugging utilities for ACAP.

```
#include <algorithm>
#include <array>
#include <atomic>
#include <experimental/mdspan>
```

Include dependency graph for debug.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >

  Verify that an SPMD program has a bulk-synchronous behaviour across all the tiles of the CGRA. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::acap
- namespace trisycl::vendor::xilinx::acap::debug

13.389.1 Detailed Description

Some debugging utilities for ACAP.

Definition in file debug.hpp.

13.390 debug.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_ACAP_DEBUG_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_ACAP_DEBUG_HPP
00003
00004 /** \file
00005
00006 Some debugging utilities for ACAP
00007 */
00008
00009 #include <algorithm>
00010 #include <array>
00011 #include <atomic>
00012
00013 #include <experimental/mdspan>
00014
00015 /** \defgroup acap Extensions for Xilinx ACAP Versal
00016
00017 Some classes and functions to support devices including CPU, FPGA
00018 & AI Engine CGRA
00019 @{
00020 */
00021
00022 /** \defgroup acap_debug Debugging support for ACAP
00023
00024 Some classes and functions to help debuging ACAP applications,
00025 such as verifying BSP forward progress
00026 @{
00027 */
00028
00029 namespace trisycl::vendor::xilinx::acap::debug {
00030
00031 /// @}
00032 /// \ingroup acap_debug
00033 /// @{
00034
00035 /** Verify that an SPMD program has a bulk-synchronous behaviour
00036 across all the tiles of the CGRA.
00037
00038 If there are some discrepancies, display the tile id which is too
00039 late or too early.
00040 */
00041 template <typename Geography> class bsp_checker {
00042 /// A vector clock to compare differerent tiles together
00043   std::array<std::atomic<int>, Geography::size> vector_clock_alloc;
00044   std::mdspan<std::atomic<int>,
00045               std::extents<std::size_t, Geography::x_size, Geography::y_size»
00046       vector_clock { &vector_clock_alloc[0] };
00047
00048  public:
00049 /** Check that none of the tiles executing this are separated by
00050 more than one execution step each other.
00051
00052 This functions has to be called in a BSP synchronous way by all
00053 the tiles.
00054 */
00055   void check(int x, int y) {
00056     ++vector_clock[x, y];
00057     /* While the algorithm execution is not atomic, it is enough to
00058 find some asynchronous bugs */
00059     auto [min_element, max_element] = std::minmax_element(
00060         vector_clock_alloc.cbegin(), vector_clock_alloc.cend());
00061
00062     if (*max_element - *min_element > 1) {
00063       TRISYCL_DUMP_T(std::dec « "compute(" « x « ',' « y
00064                              « ") vector clock min = " « *min_element
00065                              « ", max = " « *max_element);
00066       std::terminate();
00067     }
00068   }
00069 };
00070
00071 } // namespace trisycl::vendor::xilinx::acap::debug
00072
00073 /// @} End the acap_debug Doxygen group
00074
00075 /// @} End the acap Doxygen group
00076
00077 /*
00078 # Some Emacs stuff:
00079 ### Local Variables:
00080 ### ispell-local-dictionary:  "american"
00081 ### eval:  (flyspell-prog-mode)
00082 ### End:
```

```
00083 */
00084
00085 #endif // TRISYCL_SYCL_VENDOR_XILINX_ACAP_DEBUG_HPP
```

13.391 include/sycl/vendor/Xilinx/acap/tests/processor_array.cpp File Reference

Functions

- int main ()

13.391.1 Function Documentation

13.391.1.1 main()

```
int main ( )
```

Definition at line 5 of file processor_array.cpp.

References i, p, x, and y.

13.392 processor_array.cpp

Go to the documentation of this file.

```
00001 /* Experimenting with processor arrays
00002
00003 2017-09-18--22-Khronos_F2F_Chicago-Xilinx/2017-09-19--20-Xilinx-SYCL-Next.pdf
00004 */
00005 int main() {
00006   /* This test has to be evaluated at compile time and only 1 branch
00007 appears in the binary */
00008   if constexpr (i_am_the_last_PE_of_the_pipeline()) {
00009     receive_from_left_neighbour();
00010     deal_with_IO();
00011   else {
00012     receive_from_left_neighbour();
00013     process_data();
00014     send_to_right_neighbour();
00015   }
00016
00017   // All the following code is constexpr
00018   auto d = sycl::device::get_static_devices();
00019   auto fpga = sycl::device::get_static_device<acme::FPGA>();
00020   auto p = sycl::pipe::get_static_pipe<float, 3>();
00021   auto dsp = sycl::device::get_static_device<acme::DSP>();
00022   auto dsp_line = dsp.get_sub_device<as_1D>();
00023   auto first_dsp = dsp_line[0];
00024   auto dsp_pipes = <make>(dsp_line[0]..dsp_line[d::size - 2]) ////
00025
00026   queue { fpga }.submit([&] (auto h) {
00027       h.single_task([=] {
00028           auto to_dsp_0 =
00029             p.get_access<sycl::access::mode::write,
00030                          sycl::access::target::blocking_pipe>(cgh);
00031           for /* ever */ (;;) //< This is the only non static code here!
00032             to_dsp_0 « acme::read_radio_sample_from_ADC();
00033         });
00034       });
00035
```

```
00036   hana::for_each(dsp_line, [&] (auto d) {
00037       queue { d }.submit([&] (auto h) {
00038           h.single_task([=] {
00039                 auto from_fpga =
00040                   p.get_access<sycl::access::mode::read,
00041                                sycl::access::target::blocking_pipe>(cgh);
00042                 auto from_fpga =
00043                   p.get_access<sycl::access::mode::read,
00044                                sycl::access::target::blocking_pipe>(cgh);
00045                 for /* ever */ (;;) {
00046                   if constexpr (d::order == 0) {
00047                     auto data = from_fpga.read();
00048                     to_next_neighbour_dsp « pre_process(data);
00049                   }
00050                   else if constexpr (d::order == d::size - 1) {
00051                     output « process<d>(input.read());
00052                   }
00053                   « acme::read_radio_sample_from_ADC();
00054           }
00055         });
00056       });
00057     });
00058 }
00059
00060
00061
00062 device d { my_accelerator_selector {} };
00063 // Rely on C++20 array_ref for multidimensional objects
00064 auto processor_array = d.get_device_array_ref();
00065 auto storage_array = d.get_storage_array_ref();
00066
00067 for (auto x :  processor_array.get_id(0))
00068   for (auto y :  processor_array.get_id(1))
00069     queue { processor_array[x][y] }.submit([] (auto h) {
00070         auto left = storage_array[x - 1][y].get_access(h, x > 0);
00071         auto right = storage_array[x][y].get_access(h, x < N);
00072
00073         h.single_task([] {
00074
00075             });
00076         });
00077
00078
00079 bool constexpr is_red(int x, int y) {
00080   return (x + y);
00081 };
00082
00083 struct my_me :  me_array<my_me> {
00084   struct red {
00085     float v;
00086   };
00087
00088   struct black {
00089     int i;
00090   };
00091
00092   // Local memory(x,y) shared by 4 neighbours
00093   template <int x, int y>
00094   struct local_mem
00095     :  me_array::local_mem<x, y, local_mem>
00096     , std::conditional<is_red(x,y), red, black> {
00097     int use_count = 0;
00098   };
00099
00100   // Local program of PE(x,y)
00101   template <int x, int y>
00102   struct tile :  me_array::tile<x, y, tile> {
00103
00104     void hello() {
00105       std::cout « "Hello, I am the PE " « x « ',' « y « std::endl;
00106       if constexpr (is_shim()) {
00107         std::cout « "  and I am a shim PE!" « std::endl;
00108       }
00109       else if constexpr (is_noc()) {
00110         std::cout « "  and I am a NoC controller!" « std::endl;
00111       }
00112     }
00113
00114     void compute() {
00115       if constexpr (!(is_shim() || is_noc())) {
00116         for(;;) {
00117           if constexpr (is_red(x,y)) {
00118             std::lock_guard l { local_mem<x + 1>::lock(1) };
00119             v += 1.5;
00120           }
00121           else {
00122             std::lock_guard l { local_mem<x + 1, y>::lock(2) };
```

```
00123             i++;
00124           }
00125         }
00126       }
00127     }
00128   }
00129 }
```

13.393 include/triSYCL/vendor/Xilinx/acap/tests/processor_array.cpp File Reference

Functions

- int main ()

13.393.1 Function Documentation

13.393.1.1 main()

```
int main ( )
```

Definition at line 5 of file processor_array.cpp.

References i, p, x, and y.

13.394 processor_array.cpp

Go to the documentation of this file.

```
00001 /* Experimenting with processor arrays
00002
00003 2017-09-18--22-Khronos_F2F_Chicago-Xilinx/2017-09-19--20-Xilinx-SYCL-Next.pdf
00004 */
00005 int main() {
00006   /* This test has to be evaluated at compile time and only 1 branch
00007 appears in the binary */
00008   if constexpr (i_am_the_last_PE_of_the_pipeline()) {
00009     receive_from_left_neighbour();
00010     deal_with_IO();
00011   else {
00012     receive_from_left_neighbour();
00013     process_data();
00014     send_to_right_neighbour();
00015   }
00016
00017   // All the following code is constexpr
00018   auto d = sycl::device::get_static_devices();
00019   auto fpga = sycl::device::get_static_device<acme::FPGA>();
00020   auto p = sycl::pipe::get_static_pipe<float, 3>();
00021   auto dsp = sycl::device::get_static_device<acme::DSP>();
00022   auto dsp_line = dsp.get_sub_device<as_1D>();
00023   auto first_dsp = dsp_line[0];
00024   auto dsp_pipes = <make>(dsp_line[0]..dsp_line[d::size - 2]) ////
00025
00026   queue { fpga }.submit([&] (auto h) {
00027       h.single_task([=] {
00028           auto to_dsp_0 =
00029             p.get_access<sycl::access::mode::write,
00030                          sycl::access::target::blocking_pipe>(cgh);
00031           for /* ever */ (;;) //< This is the only non static code here!
```

```
00032             to_dsp_0 « acme::read_radio_sample_from_ADC();
00033         });
00034       });
00035
00036   hana::for_each(dsp_line, [&] (auto d) {
00037       queue { d }.submit([&] (auto h) {
00038           h.single_task([=] {
00039                 auto from_fpga =
00040                   p.get_access<sycl::access::mode::read,
00041                                sycl::access::target::blocking_pipe>(cgh);
00042                 auto from_fpga =
00043                   p.get_access<sycl::access::mode::read,
00044                                sycl::access::target::blocking_pipe>(cgh);
00045                 for /* ever */ (;;) {
00046                   if constexpr (d::order == 0) {
00047                     auto data = from_fpga.read();
00048                     to_next_neighbour_dsp « pre_process(data);
00049                   }
00050                   else if constexpr (d::order == d::size - 1) {
00051                     output « process<d>(input.read());
00052                   }
00053                   « acme::read_radio_sample_from_ADC();
00054           }
00055         });
00056       });
00057     });
00058 }
00059
00060
00061
00062 device d { my_accelerator_selector {} };
00063 // Rely on C++20 array_ref for multidimensional objects
00064 auto processor_array = d.get_device_array_ref();
00065 auto storage_array = d.get_storage_array_ref();
00066
00067 for (auto x :  processor_array.get_id(0))
00068   for (auto y :  processor_array.get_id(1))
00069     queue { processor_array[x][y] }.submit([] (auto h) {
00070         auto left = storage_array[x - 1][y].get_access(h, x > 0);
00071         auto right = storage_array[x][y].get_access(h, x < N);
00072
00073         h.single_task([] {
00074
00075             });
00076         });
00077
00078
00079 bool constexpr is_red(int x, int y) {
00080   return (x + y);
00081 };
00082
00083 struct my_me :  me_array<my_me> {
00084   struct red {
00085     float v;
00086   };
00087
00088   struct black {
00089     int i;
00090   };
00091
00092   // Local memory(x,y) shared by 4 neighbours
00093   template <int x, int y>
00094   struct local_mem
00095     :  me_array::local_mem<x, y, local_mem>
00096     , std::conditional<is_red(x,y), red, black> {
00097     int use_count = 0;
00098   };
00099
00100   // Local program of PE(x,y)
00101   template <int x, int y>
00102   struct tile :  me_array::tile<x, y, tile> {
00103
00104     void hello() {
00105       std::cout « "Hello, I am the PE " « x « ',' « y « std::endl;
00106       if constexpr (is_shim()) {
00107         std::cout « "  and I am a shim PE!" « std::endl;
00108       }
00109       else if constexpr (is_noc()) {
00110         std::cout « "  and I am a NoC controller!" « std::endl;
00111       }
00112     }
00113
00114     void compute() {
00115       if constexpr (!(is_shim() || is_noc())) {
00116         for(;;) {
00117           if constexpr (is_red(x,y)) {
00118             std::lock_guard l { local_mem<x + 1>::lock(1) };
```

```
00119             v += 1.5;
00120           }
00121           else {
00122             std::lock_guard l { local_mem<x + 1, y>::lock(2) };
00123             i++;
00124           }
00125         }
00126       }
00127     }
00128   }
00129 }
```

13.395 include/sycl/vendor/Xilinx/config.hpp File Reference

Control the configuration of some implementation of Xilinx-related features.

Macros

- #define TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS 8

  Define the number of threads used in the fiber executor by default if undefined in the compiler option.
- #define TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER 0

  Do not use a fiber to run the tile core program by default if undefined in the compiler option.

13.395.1 Detailed Description

Control the configuration of some implementation of Xilinx-related features.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file config.hpp.

13.395.2 Macro Definition Documentation

13.395.2.1 TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS

```
#define TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS 8
```

Define the number of threads used in the fiber executor by default if undefined in the compiler option.

Definition at line 18 of file config.hpp.

13.395.2.2 TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER

```
#define TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER 0
```

Do not use a fiber to run the tile core program by default if undefined in the compiler option.

Definition at line 24 of file config.hpp.

13.396 config.hpp

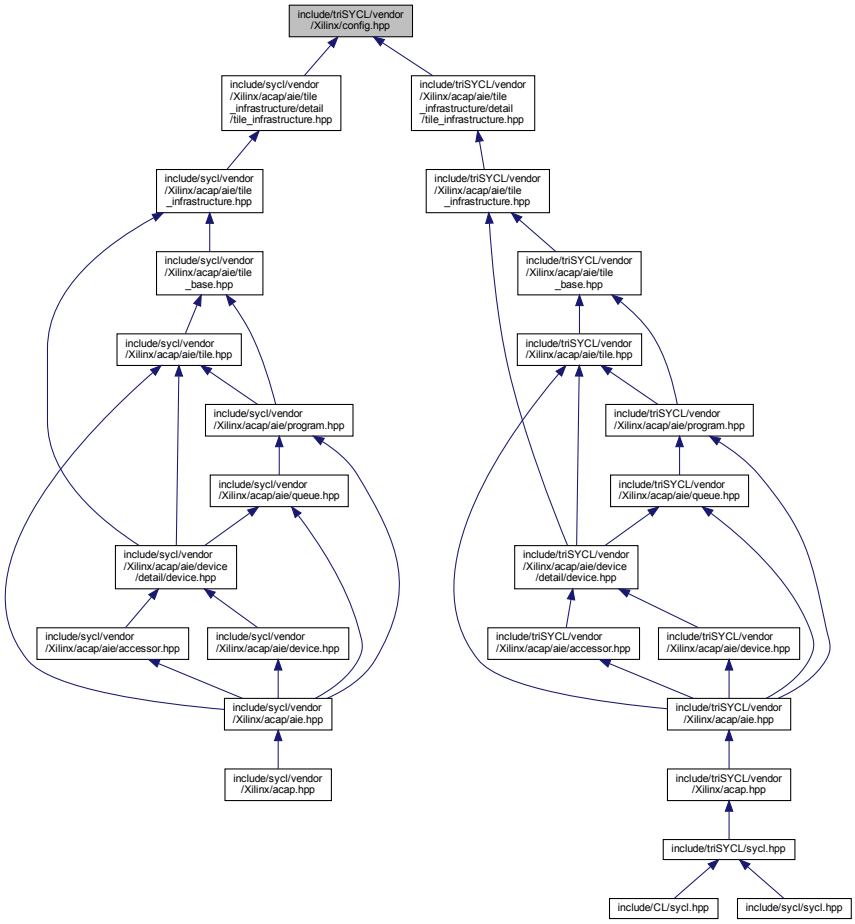
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_CONFIG_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_CONFIG_HPP
00003
00004 /** \file
00005
00006 Control the configuration of some implementation of Xilinx-related
00007 features.
00008
00009 Ronan dot Keryell at Xilinx dot com
00010
00011 This file is distributed under the University of Illinois Open Source
00012 License.  See LICENSE.TXT for details.
00013 */
00014
00015 #ifndef TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS
00016 /// Define the number of threads used in the fiber executor by default
00017 /// if undefined in the compiler option
00018 #define TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS 8
00019 #endif
00020
00021 #ifndef TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00022 /// Do not use a fiber to run the tile core program by default if undefined
00023 /// in the compiler option
00024 #define TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER 0
00025 #endif
00026
00027 /*
00028 # Some Emacs stuff:
00029 ### Local Variables:
00030 ### ispell-local-dictionary:  "american"
00031 ### eval:  (flyspell-prog-mode)
00032 ### End:
00033 */
00034
00035 #endif // TRISYCL_SYCL_VENDOR_XILINX_CONFIG_HPP
```

13.397 include/triSYCL/vendor/Xilinx/config.hpp File Reference

Control the configuration of some implementation of Xilinx-related features.

This graph shows which files directly or indirectly include this file:



Macros

- #define TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS 8

  Define the number of threads used in the fiber executor by default if undefined in the compiler option.
- #define TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER 0

  Do not use a fiber to run the tile core program by default if undefined in the compiler option.

13.397.1 Detailed Description

Control the configuration of some implementation of Xilinx-related features.

Ronan dot Keryell at Xilinx dot com

This file is distributed under the University of Illinois Open Source License. See LICENSE.TXT for details.

Definition in file config.hpp.

13.397.2 Macro Definition Documentation

13.397.2.1 TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS

```
#define TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS 8
```

Define the number of threads used in the fiber executor by default if undefined in the compiler option.

Definition at line 18 of file config.hpp.

13.397.2.2 TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER

```
#define TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER 0
```

Do not use a fiber to run the tile core program by default if undefined in the compiler option.

Definition at line 24 of file config.hpp.

13.398 config.hpp

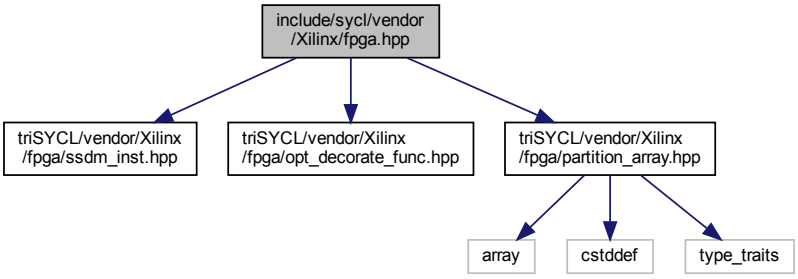
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_CONFIG_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_CONFIG_HPP
00003
00004 /** \file
00005
00006 Control the configuration of some implementation of Xilinx-related
00007 features.
00008
00009 Ronan dot Keryell at Xilinx dot com
00010
00011 This file is distributed under the University of Illinois Open Source
00012 License.  See LICENSE.TXT for details.
00013 */
00014
00015 #ifndef TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS
00016 /// Define the number of threads used in the fiber executor by default
00017 /// if undefined in the compiler option
00018 #define TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS 8
00019 #endif
00020
00021 #ifndef TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
00022 /// Do not use a fiber to run the tile core program by default if undefined
00023 /// in the compiler option
00024 #define TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER 0
00025 #endif
00026
00027 /*
00028 # Some Emacs stuff:
00029 ### Local Variables:
00030 ### ispell-local-dictionary:  "american"
00031 ### eval:  (flyspell-prog-mode)
00032 ### End:
00033 */
00034
00035 #endif // TRISYCL_SYCL_VENDOR_XILINX_CONFIG_HPP
```

13.399 include/sycl/vendor/Xilinx/fpga.hpp File Reference

```
#include "triSYCL/vendor/Xilinx/fpga/ssdm_inst.hpp"
#include "triSYCL/vendor/Xilinx/fpga/opt_decorate_func.hpp"
#include "triSYCL/vendor/Xilinx/fpga/partition_array.hpp"
```

Include dependency graph for fpga.hpp:



13.400 fpga.hpp

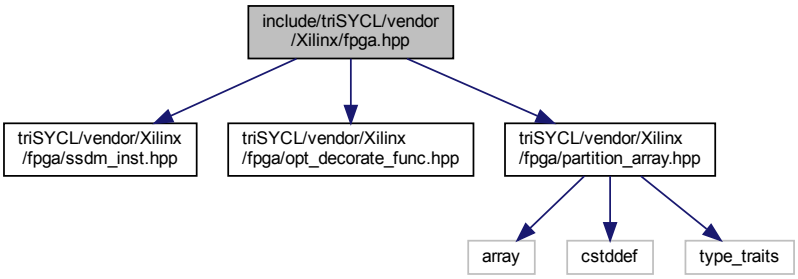
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_FPGA_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_FPGA_HPP
00003
00004 /** \file Some extensions for Xilinx FPGA
00005
00006 Ronan Keryell @ Xilinx
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 // Some include files for Xilinx-specific features.
00013 // SSDM functions needs to be declared before their implementation
00014 #include "triSYCL/vendor/Xilinx/fpga/ssdm_inst.hpp"
00015 #include "triSYCL/vendor/Xilinx/fpga/opt_decorate_func.hpp"
00016 #include "triSYCL/vendor/Xilinx/fpga/partition_array.hpp"
00017
00018 /*
00019 # Some Emacs stuff:
00020 ### Local Variables:
00021 ### ispell-local-dictionary:  "american"
00022 ### eval:  (flyspell-prog-mode)
00023 ### End:
00024 */
00025
00026 #endif // TRISYCL_SYCL_VENDOR_XILINX_FPGA_HPP
```

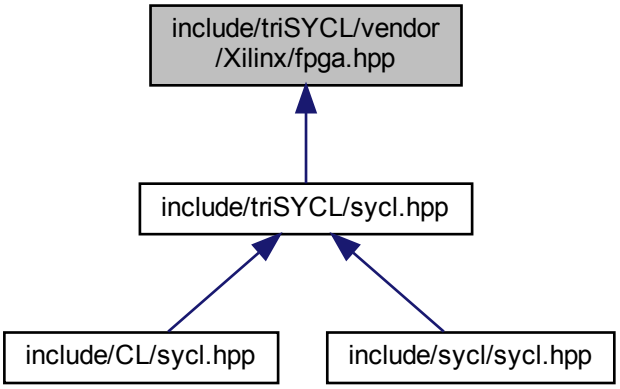
13.401 include/triSYCL/vendor/Xilinx/fpga.hpp File Reference

```
#include "triSYCL/vendor/Xilinx/fpga/ssdm_inst.hpp"
#include "triSYCL/vendor/Xilinx/fpga/opt_decorate_func.hpp"
```

`#include "triSYCL/vendor/Xilinx/fpga/partition_array.hpp"`
Include dependency graph for fpga.hpp:



This graph shows which files directly or indirectly include this file:



13.402 fpga.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_FPGA_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_FPGA_HPP
00003
00004 /** \file Some extensions for Xilinx FPGA
00005
00006 Ronan Keryell @ Xilinx
00007
00008 This file is distributed under the University of Illinois Open Source
00009 License.  See LICENSE.TXT for details.
00010 */
00011
00012 // Some include files for Xilinx-specific features.
00013 // SSDM functions needs to be declared before their implementation
00014 #include "triSYCL/vendor/Xilinx/fpga/ssdm_inst.hpp"
00015 #include "triSYCL/vendor/Xilinx/fpga/opt_decorate_func.hpp"
00016 #include "triSYCL/vendor/Xilinx/fpga/partition_array.hpp"
00017
```

```
00018 /*
00019 # Some Emacs stuff:
00020 ### Local Variables:
00021 ### ispell-local-dictionary:  "american"
00022 ### eval:  (flyspell-prog-mode)
00023 ### End:
00024 */
00025
00026 #endif // TRISYCL_SYCL_VENDOR_XILINX_FPGA_HPP
```

13.403 include/sycl/vendor/Xilinx/fpga/opt_decorate_func.hpp File Reference

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx

Variables

- auto trisycl::vendor::xilinx::dataflow

  Apply dataflow execution on functions or loops.
- auto trisycl::vendor::xilinx::pipeline

  Execute loops in a pipelined manner.

13.404 opt_decorate_func.hpp

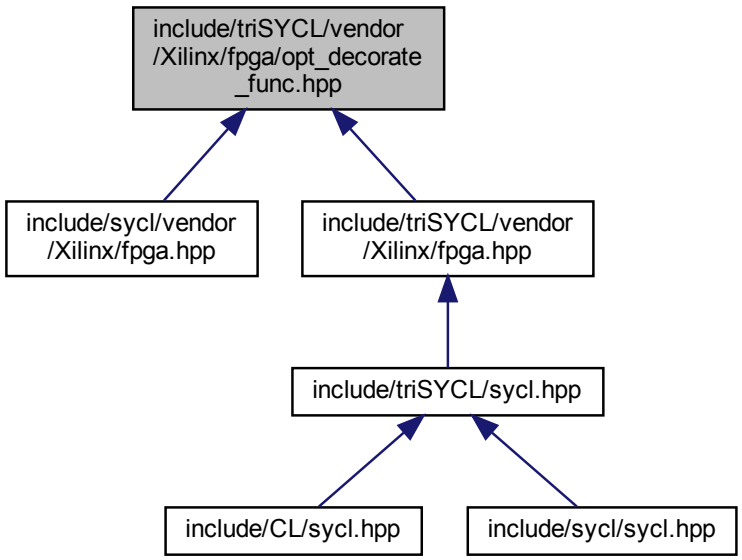
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_FPGA_OPT_DECORATE_FUNC_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_FPGA_OPT_DECORATE_FUNC_HPP
00003
00004 /** \file This file includes some decorating functions for
00005 some optimizations supported by Xilinx tools.
00006
00007 This file is distributed under the University of Illinois Open Source
00008 License.  See LICENSE.TXT for details.
00009 */
00010
00011 /** \addtogroup Xilinx Xilinx vendor extensions
00012 @{
00013 */
00014
00015 namespace trisycl::vendor::xilinx {
00016
00017 /** Apply dataflow execution on functions or loops
00018
00019 With this mode, Xilinx tools analyze the dataflow dependencies
00020 between sequential functions or loops and create channels (based
00021 on ping-pong RAMs or FIFOs) that allow consumer functions or loops
00022 to start operation before the producer functions or loops have
00023 completed.
00024
00025 This allows functions or loops to operate in parallel, which
00026 decreases latency and improves the throughput.
00027
00028 \param[in] f is a function that functions or loops in f will be executed
00029 in a dataflow manner.
00030 */
00031 auto inline dataflow = [] (auto functor) noexcept {
00032   /* SSDM instruction is inserted before the argument functor to guide xocc to
00033 do dataflow.  */
00034   _ssdm_op_SpecDataflowPipeline(-1, "");
```

```
00035   functor();
00036 };
00037
00038
00039 /** Execute loops in a pipelined manner
00040
00041 A loop with pipeline mode processes a new input every clock
00042 cycle.  This allows the operations of different iterations of the
00043 loop to be executed in a concurrent manner to reduce latency.
00044
00045 \param[in] f is a function with an innermost loop to be executed in a
00046 pipeline way.
00047 */
00048 auto inline pipeline = [] (auto functor) noexcept {
00049   /* SSDM instruction is inserted before the argument functor to guide xocc to
00050 do pipeline.  */
00051   _ssdm_op_SpecPipeline(1, 1, 0, 0, "");
00052   functor();
00053 };
00054
00055 }
00056
00057 /// @} End the Xilinx Doxygen group
00058
00059 /*
00060 # Some Emacs stuff:
00061 ### Local Variables:
00062 ### ispell-local-dictionary:  "american"
00063 ### eval:  (flyspell-prog-mode)
00064 ### End:
00065 */
00066
00067 #endif // TRISYCL_SYCL_VENDOR_XILINX_FPGA_OPT_DECORATE_FUNC_HPP
```

13.405 include/triSYCL/vendor/Xilinx/fpga/opt_decorate_func.hpp File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx

13.406 opt_decorate_func.hpp

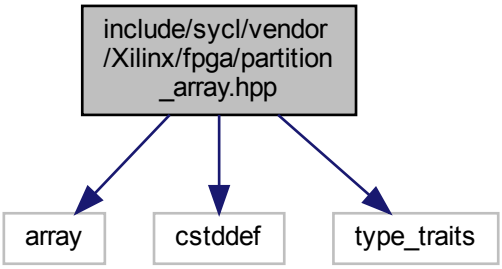
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_FPGA_OPT_DECORATE_FUNC_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_FPGA_OPT_DECORATE_FUNC_HPP
00003
00004 /** \file This file includes some decorating functions for
00005 some optimizations supported by Xilinx tools.
00006
00007 This file is distributed under the University of Illinois Open Source
00008 License.  See LICENSE.TXT for details.
00009 */
00010
00011 /** \addtogroup Xilinx Xilinx vendor extensions
00012 @{
00013 */
00014
00015 namespace trisycl::vendor::xilinx {
00016
00017 /** Apply dataflow execution on functions or loops
00018
00019 With this mode, Xilinx tools analyze the dataflow dependencies
00020 between sequential functions or loops and create channels (based
00021 on ping-pong RAMs or FIFOs) that allow consumer functions or loops
00022 to start operation before the producer functions or loops have
00023 completed.
00024
00025 This allows functions or loops to operate in parallel, which
00026 decreases latency and improves the throughput.
00027
00028 \param[in] f is a function that functions or loops in f will be executed
00029 in a dataflow manner.
00030 */
00031 auto inline dataflow = [] (auto functor) noexcept {
00032   /* SSDM instruction is inserted before the argument functor to guide xocc to
00033 do dataflow.  */
00034   _ssdm_op_SpecDataflowPipeline(-1, "");
00035   functor();
00036 };
00037
00038
00039 /** Execute loops in a pipelined manner
00040
00041 A loop with pipeline mode processes a new input every clock
00042 cycle.  This allows the operations of different iterations of the
00043 loop to be executed in a concurrent manner to reduce latency.
00044
00045 \param[in] f is a function with an innermost loop to be executed in a
00046 pipeline way.
00047 */
00048 auto inline pipeline = [] (auto functor) noexcept {
00049   /* SSDM instruction is inserted before the argument functor to guide xocc to
00050 do pipeline.  */
00051   _ssdm_op_SpecPipeline(1, 1, 0, 0, "");
00052   functor();
00053 };
00054
00055 }
00056
00057 /// @} End the Xilinx Doxygen group
00058
00059 /*
00060 # Some Emacs stuff:
00061 ### Local Variables:
00062 ### ispell-local-dictionary:  "american"
00063 ### eval:  (flyspell-prog-mode)
00064 ### End:
00065 */
00066
00067 #endif // TRISYCL_SYCL_VENDOR_XILINX_FPGA_OPT_DECORATE_FUNC_HPP
```

13.407 include/sycl/vendor/Xilinx/fpga/partition_array.hpp File Reference

```
#include <array>
#include <cstddef>
#include <type_traits>
```

Include dependency graph for partition_array.hpp:



Classes

- struct trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim >

  Represent a cyclic partition.
- struct trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim >

  Represent a block partition.
- struct trisycl::vendor::xilinx::partition::complete< PDim >

  Represent a complete partition.
- struct trisycl::vendor::xilinx::partition::none

  Represent a none partition.
- struct trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >

  Define an array class with partition feature. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::partition

  Kind of array partition.

Enumerations

- enum class trisycl::vendor::xilinx::partition::type { trisycl::vendor::xilinx::partition::cyclic , trisycl::vendor::xilinx::partition::block , trisycl::vendor::xilinx::partition::complete , trisycl::vendor::xilinx::partition::none , trisycl::vendor::xilinx::partition::cyclic , trisycl::vendor::xilinx::partition::block , trisycl::vendor::xilinx::partition::complete , trisycl::vendor::xilinx::partition::none }

  Three real partition types: cyclic, block, complete.

13.408 partition_array.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_FPGA_PARTITION_ARRAY_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_FPGA_PARTITION_ARRAY_HPP
00003
00004 /** \file This is a class expressing arrays that can be partitioned.
00005
00006 \todo While Xilinx xocc supports multidimensional C arrays, the
00007 current implementation only support 1-dim array partition.
00008
00009 \todo Extend this with multidimensional C++ arrays, such as with future
00010 mdspan C++20 syntax.
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include <array>
00017 #include <cstddef>
00018 #include <type_traits>
00019
00020 /** \addtogroup Xilinx Xilinx vendor extensions
00021 @{
00022 */
00023
00024 namespace trisycl::vendor::xilinx {
00025
00026 /** Kind of array partition
00027
00028 To be used when defining or declaring a vendor-supported partition
00029 array in kernel.
00030 */
00031 namespace partition {
00032 /** Three real partition types:  cyclic, block, complete.
00033
00034 none represents non partitioned standard array.
00035 */
00036   enum class type {
00037     cyclic,
00038     block,
00039     complete,
00040     none
00041   };
00042
00043
00044 /** Represent a cyclic partition.
00045
00046 The single array would be partitioned into several small physical
00047 memories in this case.  These small physical memories can be
00048 accessed simultaneously which drive the performance.  Each
00049 element in the array would be partitioned to each memory in order
00050 and cyclically.
00051
00052 That is if we have a 4-element array which contains 4 integers
00053 0, 1, 2, and 3.  If we set factor to 2, and partition
00054 dimension to 1 for this cyclic partition array.  Then, the
00055 contents of this array will be distributed to 2 physical
00056 memories:  one contains 1, 3 and the other contains 2,4.
00057
00058 \param PhyMemNum is the number of physical memories that user wants to
00059 have.
00060
00061 \param PDim is the dimension that user wants to apply cyclic partition on.
00062 If PDim is 0, all dimensions will be partitioned with cyclic order.
00063
00064 \todo Deal with multi-dimension array.  Now, since we can only deal with
00065 1-dim, PDim is set to 1 by default.
00066 */
00067   template <std::size_t PhyMemNum = 1, std::size_t PDim = 1>
00068   struct cyclic {
00069     static constexpr auto physical_mem_num = PhyMemNum;
00070     static constexpr auto partition_dim = PDim;
00071     static constexpr auto partition_type = type::cyclic;
00072   };
00073
00074
00075 /** Represent a block partition.
00076
00077 The single array would be partitioned into several small
00078 physical memories and can be accessed simultaneously, too.
00079 However, the first physical memory will be filled up first, then
00080 the next.
00081
00082 That is if we have a 4-element array which contains 4 integers
```

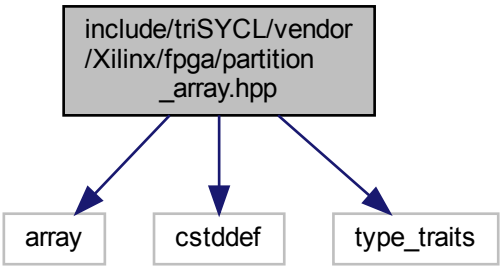
```
00083 0, 1, 2, and 3.  If we set factor to 2, and partition
00084 dimension to 1 for this cyclic partition array.  Then, the
00085 contents of this array will be distributed to 2 physical
00086 memories:  one contains 1, 2 and the other contains 3,4.
00087
00088 \param ElmInEachPhyMem is the number of elements in each physical memory
00089 that user wants to have.
00090
00091 \param PDim is the dimension that user wants to apply block partition on.
00092 If PDim is 0, all dimensions will be partitioned with block order.
00093
00094 \todo Deal with multi-dimension array.  Now, since we can only deal with
00095 1-dim, PDim is set to 1 by default.
00096 */
00097   template <std::size_t ElmInEachPhyMem = 1, std::size_t PDim = 1>
00098   struct block {
00099     static constexpr auto ele_in_each_physical_mem = ElmInEachPhyMem;
00100     static constexpr auto partition_dim = PDim;
00101     static constexpr auto partition_type = type::block;
00102   };
00103
00104
00105 /** Represent a complete partition.
00106
00107 The single array would be partitioned into individual elements.
00108 That is if we have a 4-element array with one dimension, the
00109 array is completely partitioned into distributed RAM or 4
00110 independent registers.
00111
00112 \param PDim is the dimension that user wants to apply complete partition
00113 on.  If PDim is 0, all dimensions will be completely partitioned.
00114
00115 \todo Deal with multi-dimension array.  Now, since we can only deal with
00116 1-dim, PDim is set to 1 by default.
00117 */
00118   template <std::size_t PDim = 1>
00119   struct complete {
00120     static constexpr auto partition_dim = PDim;
00121     static constexpr auto partition_type = type::complete;
00122   };
00123
00124
00125 /** Represent a none partition.
00126
00127 The single array would be the same as std::array.
00128 */
00129   struct none {
00130     static constexpr auto partition_type = type::none;
00131   };
00132 }
00133
00134
00135 /** Define an array class with partition feature.
00136
00137 Since on FPGA, users can customize the memory architecture in the system
00138 and within the CU. Array partition help us to partition single array to
00139 multiple memories that can be accessed simultaneously getting higher memory
00140 bandwidth.
00141
00142 \param ValueType is the type of element.
00143
00144 \param Size is the size of the array.
00145
00146 \param PartitionType is the array partition type:  cyclic, block, and
00147 complete.  The default type is none.
00148
00149 \todo Deal with multi-dimension array.
00150 */
00151 template <typename ValueType,
00152           std::size_t Size,
00153           typename PartitionType = partition::none>
00154 struct partition_array {
00155 /** Store the array elements.
00156
00157 Note that it means default initialization for partition_array
00158 elements, which is lazily convenient for heterogeneous
00159 computing */
00160   ValueType elems[Size];
00161 /// The number of elements of the 1-D array
00162   static constexpr auto array_size = Size;
00163 /// The kind of partitioning
00164   static constexpr auto partition_type = PartitionType::partition_type;
00165 /// Type of array elements
00166   using element_type = ValueType;
00167
00168
00169 /// Provide iterator
```

```
00170   auto begin() { return elems; }
00171   const auto begin()const { return elems; }
00172   auto end() { return elems+Size; }
00173   const auto end()const { return elems+Size; }
00174
00175
00176 /// Evaluate size
00177   constexpr auto size() const noexcept {
00178     return Size;
00179   }
00180
00181
00182 /// Construct an array
00183   partition_array() {
00184     // Add the intrinsic according expressing to the target compiler the
00185     // partitioning to use
00186     if constexpr (partition_type == partition::type::cyclic)
00187       _ssdm_SpecArrayPartition(&(*this)[0], PartitionType::partition_dim,
00188                                "CYCLIC", PartitionType::physical_mem_num, "");
00189     if constexpr (partition_type == partition::type::block)
00190       _ssdm_SpecArrayPartition(&(*this)[0], PartitionType::partition_dim,
00191                                "BLOCK", PartitionType::ele_in_each_physical_mem,
00192                                "");
00193     if constexpr (partition_type == partition::type::complete)
00194       _ssdm_SpecArrayPartition(&(*this)[0], PartitionType::partition_dim,
00195                                "COMPLETE", 0, "");
00196   }
00197
00198
00199 /// A constructor from some container
00200   template <typename SomeContainer>
00201   partition_array(const SomeContainer &src)
00202     :  partition_array { } {
00203 /// \todo Find a way to specialize this with a safer
00204 /// implementation when the size of src is at least constexpr
00205     std::copy_n(std::begin(src), Size, begin());
00206   }
00207
00208
00209 /// Construct an array from initializer_list
00210   template <typename SourceBasicType,
00211             // Only use this constructor for a real element-oriented
00212             // initializer_list we can assign, not for the case
00213             // partition_array<> a = { some_other_array }
00214             typename = std::enable_if_t<std::is_convertible<SourceBasicType,
00215                                                             ValueType>::value»
00216   constexpr partition_array(std::initializer_list<SourceBasicType> l)
00217     :  partition_array { } {
00218 /// \todo Find a way to specialize this with a safer
00219 /// implementation when the size of src is at least constexpr
00220 /// This does not work...
00221 /// static_assert(l.size() == Size);
00222     std::copy_n(std::begin(l), Size, begin());
00223   }
00224
00225
00226 /// Provide a subscript operator
00227   constexpr ValueType& operator[](std::size_t i) {
00228     return elems[i];
00229   }
00230
00231
00232   constexpr const ValueType& operator[](std::size_t i)const {
00233     return elems[i];
00234   }
00235
00236
00237 /// Return the partition type of the array
00238   constexpr auto get_partition_type()const {
00239     return partition_type;
00240   }
00241 };
00242
00243 /// @} End the Xilinx Doxygen group
00244
00245 }
00246
00247 /*
00248 # Some Emacs stuff:
00249 ### Local Variables:
00250 ### ispell-local-dictionary:  "american"
00251 ### eval:  (flyspell-prog-mode)
00252 ### End:
00253 */
00254
00255 #endif// TRISYCL_SYCL_VENDOR_XILINX_FPGA_PARTITION_ARRAY_HPP
```

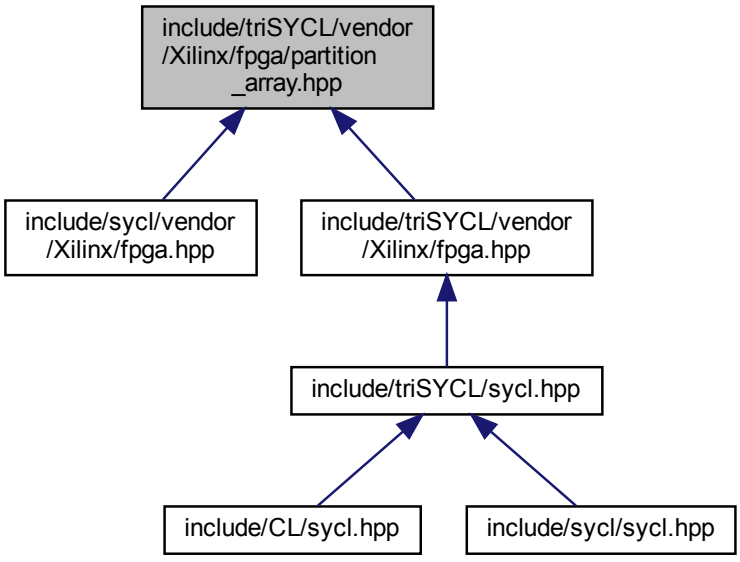
13.409 include/triSYCL/vendor/Xilinx/fpga/partition_array.hpp File Reference

```
#include <array>
#include <cstddef>
#include <type_traits>
```

Include dependency graph for partition_array.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim >

Represent a cyclic partition.

- struct trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim >

  Represent a block partition.

- struct trisycl::vendor::xilinx::partition::complete< PDim >

  Represent a complete partition.

- struct trisycl::vendor::xilinx::partition::none

  Represent a none partition.

- struct trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >

  Define an array class with partition feature. More...

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.

- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::partition

  Kind of array partition.

Enumerations

- enum class trisycl::vendor::xilinx::partition::type { trisycl::vendor::xilinx::partition::cyclic , trisycl::vendor::xilinx::partition::block , trisycl::vendor::xilinx::partition::complete , trisycl::vendor::xilinx::partition::none , trisycl::vendor::xilinx::partition::cyclic , trisycl::vendor::xilinx::partition::block , trisycl::vendor::xilinx::partition::complete , trisycl::vendor::xilinx::partition::none }

  Three real partition types: cyclic, block, complete.

13.410 partition_array.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_FPGA_PARTITION_ARRAY_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_FPGA_PARTITION_ARRAY_HPP
00003
00004 /** \file This is a class expressing arrays that can be partitioned.
00005
00006 \todo While Xilinx xocc supports multidimensional C arrays, the
00007 current implementation only support 1-dim array partition.
00008
00009 \todo Extend this with multidimensional C++ arrays, such as with future
00010 mdspan C++20 syntax.
00011
00012 This file is distributed under the University of Illinois Open Source
00013 License.  See LICENSE.TXT for details.
00014 */
00015
00016 #include <array>
00017 #include <cstddef>
00018 #include <type_traits>
00019
00020 /** \addtogroup Xilinx Xilinx vendor extensions
00021 @{
00022 */
00023
00024 namespace trisycl::vendor::xilinx {
00025
00026 /** Kind of array partition
00027
00028 To be used when defining or declaring a vendor-supported partition
00029 array in kernel.
00030 */
```

```
00031 namespace partition {
00032 /** Three real partition types:  cyclic, block, complete.
00033
00034 none represents non partitioned standard array.
00035 */
00036   enum class type {
00037     cyclic,
00038     block,
00039     complete,
00040     none
00041   };
00042
00043
00044 /** Represent a cyclic partition.
00045
00046 The single array would be partitioned into several small physical
00047 memories in this case.  These small physical memories can be
00048 accessed simultaneously which drive the performance.  Each
00049 element in the array would be partitioned to each memory in order
00050 and cyclically.
00051
00052 That is if we have a 4-element array which contains 4 integers
00053 0, 1, 2, and 3.  If we set factor to 2, and partition
00054 dimension to 1 for this cyclic partition array.  Then, the
00055 contents of this array will be distributed to 2 physical
00056 memories:  one contains 1, 3 and the other contains 2,4.
00057
00058 \param PhyMemNum is the number of physical memories that user wants to
00059 have.
00060
00061 \param PDim is the dimension that user wants to apply cyclic partition on.
00062 If PDim is 0, all dimensions will be partitioned with cyclic order.
00063
00064 \todo Deal with multi-dimension array.  Now, since we can only deal with
00065 1-dim, PDim is set to 1 by default.
00066 */
00067   template <std::size_t PhyMemNum = 1, std::size_t PDim = 1>
00068   struct cyclic {
00069     static constexpr auto physical_mem_num = PhyMemNum;
00070     static constexpr auto partition_dim = PDim;
00071     static constexpr auto partition_type = type::cyclic;
00072   };
00073
00074
00075 /** Represent a block partition.
00076
00077 The single array would be partitioned into several small
00078 physical memories and can be accessed simultaneously, too.
00079 However, the first physical memory will be filled up first, then
00080 the next.
00081
00082 That is if we have a 4-element array which contains 4 integers
00083 0, 1, 2, and 3.  If we set factor to 2, and partition
00084 dimension to 1 for this cyclic partition array.  Then, the
00085 contents of this array will be distributed to 2 physical
00086 memories:  one contains 1, 2 and the other contains 3,4.
00087
00088 \param ElmInEachPhyMem is the number of elements in each physical memory
00089 that user wants to have.
00090
00091 \param PDim is the dimension that user wants to apply block partition on.
00092 If PDim is 0, all dimensions will be partitioned with block order.
00093
00094 \todo Deal with multi-dimension array.  Now, since we can only deal with
00095 1-dim, PDim is set to 1 by default.
00096 */
00097   template <std::size_t ElmInEachPhyMem = 1, std::size_t PDim = 1>
00098   struct block {
00099     static constexpr auto ele_in_each_physical_mem = ElmInEachPhyMem;
00100     static constexpr auto partition_dim = PDim;
00101     static constexpr auto partition_type = type::block;
00102   };
00103
00104
00105 /** Represent a complete partition.
00106
00107 The single array would be partitioned into individual elements.
00108 That is if we have a 4-element array with one dimension, the
00109 array is completely partitioned into distributed RAM or 4
00110 independent registers.
00111
00112 \param PDim is the dimension that user wants to apply complete partition
00113 on.  If PDim is 0, all dimensions will be completely partitioned.
00114
00115 \todo Deal with multi-dimension array.  Now, since we can only deal with
00116 1-dim, PDim is set to 1 by default.
00117 */
```

```
00118   template <std::size_t PDim = 1>
00119   struct complete {
00120     static constexpr auto partition_dim = PDim;
00121     static constexpr auto partition_type = type::complete;
00122   };
00123
00124
00125 /** Represent a none partition.
00126
00127 The single array would be the same as std::array.
00128 */
00129   struct none {
00130     static constexpr auto partition_type = type::none;
00131   };
00132 }
00133
00134
00135 /** Define an array class with partition feature.
00136
00137 Since on FPGA, users can customize the memory architecture in the system
00138 and within the CU. Array partition help us to partition single array to
00139 multiple memories that can be accessed simultaneously getting higher memory
00140 bandwidth.
00141
00142 \param ValueType is the type of element.
00143
00144 \param Size is the size of the array.
00145
00146 \param PartitionType is the array partition type:  cyclic, block, and
00147 complete.  The default type is none.
00148
00149 \todo Deal with multi-dimension array.
00150 */
00151 template <typename ValueType,
00152           std::size_t Size,
00153           typename PartitionType = partition::none>
00154 struct partition_array {
00155 /** Store the array elements.
00156
00157 Note that it means default initialization for partition_array
00158 elements, which is lazily convenient for heterogeneous
00159 computing */
00160   ValueType elems[Size];
00161 /// The number of elements of the 1-D array
00162   static constexpr auto array_size = Size;
00163 /// The kind of partitioning
00164   static constexpr auto partition_type = PartitionType::partition_type;
00165 /// Type of array elements
00166   using element_type = ValueType;
00167
00168
00169 /// Provide iterator
00170   auto begin() { return elems; }
00171   const auto begin()const { return elems; }
00172   auto end() { return elems+Size; }
00173   const auto end()const { return elems+Size; }
00174
00175
00176 /// Evaluate size
00177   constexpr auto size() const noexcept {
00178     return Size;
00179   }
00180
00181
00182 /// Construct an array
00183   partition_array() {
00184     // Add the intrinsic according expressing to the target compiler the
00185     // partitioning to use
00186     if constexpr (partition_type == partition::type::cyclic)
00187       _ssdm_SpecArrayPartition(&(*this)[0], PartitionType::partition_dim,
00188                                "CYCLIC", PartitionType::physical_mem_num, "");
00189     if constexpr (partition_type == partition::type::block)
00190       _ssdm_SpecArrayPartition(&(*this)[0], PartitionType::partition_dim,
00191                                "BLOCK", PartitionType::ele_in_each_physical_mem,
00192                                "");
00193     if constexpr (partition_type == partition::type::complete)
00194       _ssdm_SpecArrayPartition(&(*this)[0], PartitionType::partition_dim,
00195                                "COMPLETE", 0, "");
00196   }
00197
00198
00199 /// A constructor from some container
00200   template <typename SomeContainer>
00201   partition_array(const SomeContainer &src)
00202     :  partition_array { } {
00203 /// \todo Find a way to specialize this with a safer
00204 /// implementation when the size of src is at least constexpr
```

```
00205     std::copy_n(std::begin(src), Size, begin());
00206   }
00207
00208
00209 /// Construct an array from initializer_list
00210   template <typename SourceBasicType,
00211             // Only use this constructor for a real element-oriented
00212             // initializer_list we can assign, not for the case
00213             // partition_array<> a = { some_other_array }
00214             typename = std::enable_if_t<std::is_convertible<SourceBasicType,
00215                                                             ValueType>::value»
00216   constexpr partition_array(std::initializer_list<SourceBasicType> l)
00217     :  partition_array { } {
00218 /// \todo Find a way to specialize this with a safer
00219 /// implementation when the size of src is at least constexpr
00220 /// This does not work...
00221 /// static_assert(l.size() == Size);
00222     std::copy_n(std::begin(l), Size, begin());
00223   }
00224
00225
00226 /// Provide a subscript operator
00227   constexpr ValueType& operator[](std::size_t i) {
00228     return elems[i];
00229   }
00230
00231
00232   constexpr const ValueType& operator[](std::size_t i)const {
00233     return elems[i];
00234   }
00235
00236
00237 /// Return the partition type of the array
00238   constexpr auto get_partition_type()const {
00239     return partition_type;
00240   }
00241 };
00242
00243 /// @} End the Xilinx Doxygen group
00244
00245 }
00246
00247 /*
00248 # Some Emacs stuff:
00249 ### Local Variables:
00250 ### ispell-local-dictionary:  "american"
00251 ### eval:  (flyspell-prog-mode)
00252 ### End:
00253 */
00254
00255 #endif// TRISYCL_SYCL_VENDOR_XILINX_FPGA_PARTITION_ARRAY_HPP
```

13.411 include/sycl/vendor/Xilinx/fpga/ssdm_inst.hpp File Reference

Macros

- #define _ssdm_op_SpecDataflowPipeline(...) do { } while (0)
- #define _ssdm_op_SpecPipeline(...) do { } while (0)
- #define _ssdm_SpecArrayPartition(...) do { } while (0)

13.412 ssdm_inst.hpp

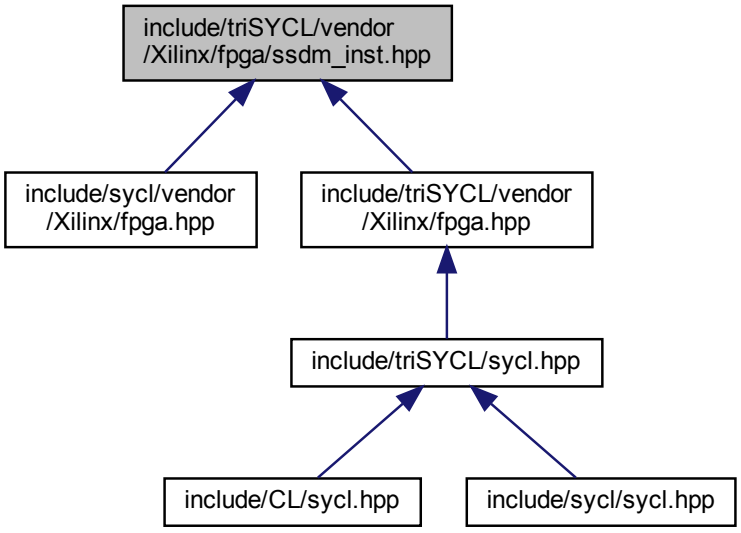
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_FPGA_SSDM_INST_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_FPGA_SSDM_INST_HPP
00003
00004 /** \file This file defines some of the SSDM intrinsics used in Xilinx
00005 tools.
00006
00007 This file is distributed under the University of Illinois Open Source
00008 License.  See LICENSE.TXT for details.
00009 */
00010
```

```
00011 /** \addtogroup Xilinx Xilinx vendor extensions
00012 @{
00013 */
00014
00015 #ifdef TRISYCL_DEVICE
00016 extern "C" {
00017 /// SSDM Intrinsics:  dataflow operation
00018   void _ssdm_op_SpecDataflowPipeline(...)
00019     __attribute__ ((nothrow, noinline, weak));
00020 /// SSDM Intrinsics:  pipeline operation
00021   void _ssdm_op_SpecPipeline(...)  __attribute__ ((nothrow, noinline, weak));
00022 /// SSDM Intrinsics:  array partition operation
00023   void _ssdm_SpecArrayPartition(...)  __attribute__ ((nothrow, noinline, weak));
00024 }
00025 #else
00026 /* If not on device, just ignore the intrinsics as defining them as
00027 empty variadic macros replaced by an empty do-while to avoid some
00028 warning when compiling (and used in an if branch */
00029 #define _ssdm_op_SpecDataflowPipeline(...)  do { } while (0)
00030 #define _ssdm_op_SpecPipeline(...)  do { } while (0)
00031 #define _ssdm_SpecArrayPartition(...)  do { } while (0)
00032 #endif
00033
00034 /// @} End the Xilinx Doxygen group
00035
00036 /*
00037 # Some Emacs stuff:
00038 ### Local Variables:
00039 ### ispell-local-dictionary:  "american"
00040 ### eval:  (flyspell-prog-mode)
00041 ### End:
00042 */
00043
00044 #endif // TRISYCL_SYCL_VENDOR_XILINX_FPGA_SSDM_INST_HPP
```

13.413 include/triSYCL/vendor/Xilinx/fpga/ssdm_inst.hpp File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define _ssdm_op_SpecDataflowPipeline(...) do { } while (0)

- #define _ssdm_op_SpecPipeline(...) do { } while (0)
- #define _ssdm_SpecArrayPartition(...) do { } while (0)

13.414 ssdm_inst.hpp

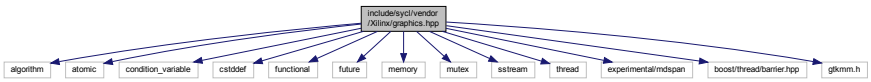
Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_FPGA_SSDM_INST_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_FPGA_SSDM_INST_HPP
00003
00004 /** \file This file defines some of the SSDM intrinsics used in Xilinx
00005 tools.
00006
00007 This file is distributed under the University of Illinois Open Source
00008 License.  See LICENSE.TXT for details.
00009 */
00010
00011 /** \addtogroup Xilinx Xilinx vendor extensions
00012 @{
00013 */
00014
00015 #ifdef TRISYCL_DEVICE
00016 extern "C" {
00017 /// SSDM Intrinsics:  dataflow operation
00018   void _ssdm_op_SpecDataflowPipeline(...)
00019     __attribute__ ((nothrow, noinline, weak));
00020 /// SSDM Intrinsics:  pipeline operation
00021   void _ssdm_op_SpecPipeline(...)  __attribute__ ((nothrow, noinline, weak));
00022 /// SSDM Intrinsics:  array partition operation
00023   void _ssdm_SpecArrayPartition(...)  __attribute__ ((nothrow, noinline, weak));
00024 }
00025 #else
00026 /* If not on device, just ignore the intrinsics as defining them as
00027 empty variadic macros replaced by an empty do-while to avoid some
00028 warning when compiling (and used in an if branch */
00029 #define _ssdm_op_SpecDataflowPipeline(...)  do { } while (0)
00030 #define _ssdm_op_SpecPipeline(...)  do { } while (0)
00031 #define _ssdm_SpecArrayPartition(...)  do { } while (0)
00032 #endif
00033
00034 /// @} End the Xilinx Doxygen group
00035
00036 /*
00037 # Some Emacs stuff:
00038 ### Local Variables:
00039 ### ispell-local-dictionary:  "american"
00040 ### eval:  (flyspell-prog-mode)
00041 ### End:
00042 */
00043
00044 #endif // TRISYCL_SYCL_VENDOR_XILINX_FPGA_SSDM_INST_HPP
```

13.415 include/sycl/vendor/Xilinx/graphics.hpp File Reference

```
#include <algorithm>
#include <atomic>
#include <condition_variable>
#include <cstddef>
#include <functional>
#include <future>
#include <memory>
#include <mutex>
#include <sstream>
#include <thread>
#include <experimental/mdspan>
#include <boost/thread/barrier.hpp>
```

`#include <gtkmm.h>`
Include dependency graph for graphics.hpp:



Classes

- struct trisycl::vendor::xilinx::graphics::frame_grid

  An application window displaying a grid of tiles.
- struct trisycl::vendor::xilinx::graphics::palette

  Color palette to project a linear data space to an RGB color space.
- struct trisycl::vendor::xilinx::graphics::image_grid

  An application window displaying a grid of tiled images.
- struct trisycl::vendor::xilinx::graphics::application

  A graphics application running in a separate thread to display images in a grid of tiles.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::graphics

Typedefs

- using trisycl::vendor::xilinx::graphics::rgb = std::array< std::uint8_t, 3 >

13.416 graphics.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_GRAPHICS_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_GRAPHICS_HPP
00003
00004 /** \file Some graphics windowing support useful for debugging
00005
00006 Based on GTK+3 with GTKMM 3 en C++ wrapper.
00007
00008 There are several graphical back-ends available with GTK.
00009
00010 An interesting one is the Broadway X11 back-end, allowing to
00011 display in a Web browser :-)
00012
00013 broadwayd :5 &
00014 xdg-open http://127.0.0.1:8085
00015 GDK_BACKEND=broadway BROADWAY_DISPLAY=:5 acap/wave_propagation
00016
00017 Ronan at Keryell point FR
00018
00019 This file is distributed under the University of Illinois Open Source
00020 License.  See LICENSE.TXT for details.
00021 */
00022
```

```
00023 #include <algorithm>
00024 #include <atomic>
00025 #include <condition_variable>
00026 #include <cstddef>
00027 #include <functional>
00028 #include <future>
00029 #include <memory>
00030 #include <mutex>
00031 #include <sstream>
00032 #include <thread>
00033
00034 #include <experimental/mdspan>
00035
00036 #include <boost/thread/barrier.hpp>
00037
00038 #include <gtkmm.h>
00039
00040 /** \defgroup graphics Graphics support for CGRA-like interaction
00041
00042 This can create a graphics application and typically update some
00043 sub-images on a frame
00044 @{
00045 */
00046
00047 namespace trisycl::vendor::xilinx::graphics {
00048
00049 // RGB 8 bit images, so 3 bytes per pixel
00050 using rgb = std::array<std::uint8_t, 3>;
00051
00052 /** An application window displaying a grid of tiles
00053
00054 Each tile is framed with the tile identifiers
00055 */
00056 struct frame_grid :  Gtk::ApplicationWindow {
00057 /// Nice to have some scroll bars around when the main window is too small
00058   Gtk::ScrolledWindow sw;
00059
00060 /// The container to represent the grid of tile images
00061   Gtk::Grid grid;
00062
00063 /** A close button in case the closing function is not provided by
00064 the window manager */
00065   Gtk::Button close_button { "Close" };
00066
00067 /** The linearized 2D vector of frames used to decorate the tile
00068 images with some frames with the tile names */
00069   std::vector<Gtk::Frame> frames;
00070
00071 /// Number of frame columns
00072   int nx;
00073
00074 /// Number of frame lines
00075   int ny;
00076
00077 /// An action to do when the window is closed
00078   std::function<void(void)> close_action;
00079
00080 /// Set to true by the closing handler
00081   std::atomic<bool> done = false;
00082
00083
00084 /** Create a grid of tiles
00085
00086 \param[in] nx is the numer of tiles horizontally
00087
00088 \param[in] ny is the numer of tiles vertically
00089 */
00090   frame_grid(int nx, int ny) :  nx { nx }, ny { ny } {
00091     set_default_size(1900, 1020);
00092
00093     add(sw);
00094     sw.add(grid);
00095     for (int y = 0; y < ny; ++y)
00096       for (int x = 0; x < nx; ++x) {
00097         std::ostringstream s;
00098         s « "Tile(" « x « ',' « y « ')';
00099         frames.emplace_back(s.str());
00100         frames.back().set_shadow_type(Gtk::SHADOW_ETCHED_OUT);
00101         // A minimal border to save space on main window
00102         frames.back().set_border_width(1);
00103         // Display the frame with the lower y South in a mathematical sense
00104         grid.attach(frames.back(), x, ny - y - 1, 1, 1);
00105       }
00106
00107     grid.add(close_button);
00108
00109     // Make the button the default widget
```

```
00110     close_button.set_can_default();
00111     close_button.grab_default();
00112
00113     // Connect the clicked signal of the close button to
00114     close_button.signal_clicked().connect([this] {
00115         // Unmap the window from the screen
00116         hide();
00117         // Call the handler if it does exist
00118         if (close_action)
00119           close_action();
00120       });
00121
00122     // Show all children of the window
00123     show_all_children();
00124   }
00125
00126
00127 /// Set a function to be called on close
00128   void set_close_action(std::function<void(void)> f) {
00129     close_action = f;
00130   }
00131
00132
00133 /// Get the frame at a given grid position
00134   auto &get_frame(int x, int y) {
00135     return frames.at(x + nx*y);
00136   }
00137
00138 };
00139
00140
00141 /** Color palette to project a linear data space to an RGB color space
00142
00143 Reimplement the palette concept by Nicolas Paris in PompC & HyperC
00144 back from the 90's...  :-)
00145 */
00146 struct palette {
00147   static constexpr int size = 256;
00148   std::array<rgb, size> color_mapping;
00149
00150   enum kind { gray, rainbow };
00151   kind k;
00152   int phase;
00153   int clip;
00154   int frequency_log2;
00155
00156 /** Create a palette
00157
00158 \param[in] k is the kind of palette
00159
00160 \param[in] phase is optional and allows the palette to have a
00161 phase shift instead of starting at 0
00162
00163 \param[in] clip is optional and specifies a value to be enhanced
00164 */
00165   palette(kind k = gray, int phase = 0, int frequency_log2 = 0, int clip = -1)
00166     :  k { k }, phase { phase } , clip { clip }, frequency_log2{frequency_log2} {
00167       update();
00168     }
00169
00170
00171   // Create a gray palette
00172   void gray_palette() {
00173     for (int i = 0; auto &e :  color_mapping) {
00174       if (i == clip)
00175         // Put a full red color for the selected value
00176         e = { 255, 0, 0 };
00177       else {
00178         int j;
00179         // There is no bidirectional shift operator in C++
00180         if (frequency_log2 < 0)
00181           // Divide the step by 2^{-fl}
00182           j = i » -frequency_log2;
00183         else
00184           // Multiply the step by 2^{fl}
00185           j = i « frequency_log2;
00186         auto v = static_cast<std::uint8_t>(j + phase);
00187         // A non-saturated "gray" of luminosity v
00188         e = { v,  v,  v };
00189       }
00190       ++i;
00191     }
00192   }
00193
00194
00195   // Create a rainbow palette
00196   void rainbow_palette() {
```

```
00197     Gdk::RGBA color;
00198     for (int i = 0; auto &e :  color_mapping) {
00199       // Put the selected value as black
00200       double luminosity = !(i == clip);
00201       int j;
00202       // There is no bidirectional shift operator in C++
00203       if (frequency_log2 < 0)
00204         // Divide the step by 2^{-fl}
00205         j = i » -frequency_log2;
00206       else
00207         // Multiply the step by 2^{fl}
00208         j = i « frequency_log2;
00209       auto v = static_cast<std::uint8_t>(j + phase);
00210       color.set_hsv(360.*v/size, 1, luminosity);
00211       e = { static_cast<std::uint8_t>(color.get_red_u() » 8),
00212             static_cast<std::uint8_t>(color.get_green_u() » 8),
00213             static_cast<std::uint8_t>(color.get_blue_u() » 8) };
00214       ++i;
00215     }
00216   }
00217
00218
00219   /* Update the palette and enforce some value invariants for
00220 defensive programming and GUI features */
00221   void update() {
00222     // Implement modulo arithmetic on interval [-1, size - 1]
00223     if (clip < -1)
00224       clip = size - 1;
00225     else if (clip >= size)
00226         clip = -1;
00227     // The frequency can only evolve in this interval
00228     frequency_log2 = std::clamp(frequency_log2, -8, 7);
00229     /* The phase can evolve modulo size.  When there are some negative
00230 numbers it is implementation dependent but that is enough for
00231 this GUI */
00232     phase = phase % size;
00233
00234     switch (k) {
00235       case gray:
00236         gray_palette();
00237         break;
00238       case rainbow:
00239         rainbow_palette();
00240         break;
00241     }
00242   }
00243
00244
00245 /** Set palette parameters and update
00246
00247 \param[in] new_k is the kind of palette
00248
00249 \param[in] new_phase allows the palette to have a phase shift
00250 instead of starting at 0
00251
00252 \param[in] new_clip specifies a value to be enhanced
00253 */
00254   void set(kind new_k, int new_phase, int new_frequency_log2, int new_clip) {
00255     k = new_k;
00256     phase = new_phase;
00257     clip = new_clip;
00258     frequency_log2 = new_frequency_log2;
00259     update();
00260   }
00261
00262
00263 /** Transform a value into an RGB color according to the palette
00264 */
00265   rgb palettize(double data, double min_value, double max_value) {
00266     std::uint8_t v = (data - min_value)*255/(max_value - min_value);
00267     // Write the same value for RGB to have a grey level
00268     return color_mapping[v];
00269   }
00270
00271
00272   void decrease_clip() {
00273     --clip;
00274     update();
00275   }
00276
00277
00278   void increase_clip() {
00279     ++clip;
00280     update();
00281   }
00282
00283
```

```
00284   void decrease_frequency() {
00285     --frequency_log2;
00286     update();
00287   }
00288
00289
00290   void increase_frequency() {
00291     ++frequency_log2;
00292     update();
00293   }
00294
00295
00296   void decrease_phase() {
00297     --phase;
00298     update();
00299   }
00300
00301
00302   void increase_phase() {
00303     ++phase;
00304     update();
00305   }
00306
00307
00308 /// Handle with any key pressed
00309   bool handle_key_press_event(GdkEventKey &event) {
00310     /* Only listen to keys without modifiers (no control, alt...)  but
00311 still allows shift */
00312     if (!(event.state
00313           & gtk_accelerator_get_default_mod_mask()
00314           & ~GDK_SHIFT_MASK)) {
00315       switch (event.keyval) {
00316       case GDK_KEY_0:
00317       case GDK_KEY_KP_0:
00318         k = gray;
00319         update();
00320         break;
00321       case GDK_KEY_1:
00322       case GDK_KEY_KP_1:
00323         k = rainbow;
00324         update();
00325         break;
00326       case GDK_KEY_minus:
00327       case GDK_KEY_KP_Subtract:
00328       case GDK_KEY_underscore:
00329         decrease_frequency();
00330         break;
00331       case GDK_KEY_plus:
00332       case GDK_KEY_KP_Add:
00333       case GDK_KEY_equal:
00334       case GDK_KEY_KP_Equal:
00335         increase_frequency();
00336         break;
00337       case GDK_KEY_h:
00338         increase_phase();
00339         break;
00340       case GDK_KEY_H:
00341         increase_phase();
00342         decrease_clip();
00343         break;
00344       case GDK_KEY_j:
00345         decrease_clip();
00346         break;
00347       case GDK_KEY_k:
00348         increase_clip();
00349         break;
00350       case GDK_KEY_l:
00351         decrease_phase();
00352         break;
00353       case GDK_KEY_L:
00354         decrease_phase();
00355         increase_clip();
00356         break;
00357       default:
00358         // Not a key press for us:  let the other listeners to handle
00359         return false;
00360       }
00361       // We have handled the event:  do not go on with any other listener
00362       return true;
00363     }
00364     // Not the right modifiers:  let the other listeners to handle
00365     return false;
00366   }
00367
00368 };
00369
00370
```

```
00371 /// An application window displaying a grid of tiled images
00372 struct image_grid :  frame_grid {
00373 /// The linearized 2D vector of images
00374   std::vector<Gtk::Image> images;
00375
00376 /// Horizontal image size in pixels
00377   int image_x;
00378
00379 /// Vertical image size in pixels
00380   int image_y;
00381
00382 /// The image pixel zooming factor for both dimension
00383   int zoom;
00384
00385 /// Dispatcher to invoke something in the graphics thread in a safe way
00386   Glib::Dispatcher dispatcher;
00387
00388 /// Protection against concurrent update
00389   std::mutex dispatch_protection;
00390
00391 /// The condition variable used to queue up graphics update submission
00392   std::condition_variable cv;
00393
00394 /// What to dispatch
00395   std::function<void(void)> work_to_dispatch;
00396
00397 /// The RGB palette used to render the image values
00398   palette p;
00399
00400 /// A global done state across all the tile to allow a global shutdown
00401   bool done_snapshot;
00402
00403 /// Set to true once "done" state has been sampled
00404   std::atomic_flag done_has_been_sampled;
00405
00406 /// Barrier to compute the global done state across the tiles
00407   boost::barrier done_barrier { static_cast<unsigned int>(nx*ny) };
00408
00409 /** Create a grid of tiled images
00410
00411 \param[in] nx is the numer of tiles horizontally
00412
00413 \param[in] ny is the numer of tiles vertically
00414
00415 \param[in] image_x is the horizontal image pixel size
00416
00417 \param[in] image_y is the vertical image pixel size
00418
00419 \param[in] zoom is the zooming factor applied to image pixels,
00420 both horizontally and vertically
00421 */
00422   image_grid(int nx, int ny, int image_x, int image_y, int zoom)
00423     :  frame_grid { nx , ny }
00424     , image_x { image_x }
00425     , image_y { image_y }
00426     , zoom { zoom }
00427   {
00428     for (int y = 0; y < ny; ++y)
00429       for (int x = 0; x < nx; ++x) {
00430         auto &f = get_frame(x, y);
00431         auto pb = Gdk::Pixbuf::create(Gdk::Colorspace::COLORSPACE_RGB
00432                                       , false //< has_alpha
00433                                       , 8 //< bits_per_sample
00434                                       , image_x*zoom //< width
00435                                       , image_y*zoom //< height
00436                                         );
00437         images.emplace_back(pb);
00438         // Display the frame with the lower y to the South
00439         f.add(images.back());
00440       }
00441
00442     // Hook a generic dispatcher
00443     dispatcher.connect([&] {
00444         {
00445           // Only 1 customer at a time
00446           std::lock_guard lock { dispatch_protection };
00447           // Skip the work when done to avoid dead lock
00448           if (done)
00449             // Wake-up everybody waiting for sending some work to
00450             // realize that they have to give up on their hope
00451             cv.notify_all();
00452           else
00453             work_to_dispatch();
00454           // Discard the previous work
00455           work_to_dispatch = nullptr;
00456         }
00457         // We can serve the next customer
```

```
00458         cv.notify_one();
00459       });
00460
00461     // Go live
00462     show_all_children();
00463   }
00464
00465
00466 /// Deal with the key pressed
00467   bool on_key_press_event(GdkEventKey* event)override {
00468     // Forward to the palette for action
00469     return p.handle_key_press_event(*event);
00470   }
00471
00472 /// Submit some work to the graphics thread
00473   void submit(std::function<void(void)> f) {
00474     std::unique_lock lock { dispatch_protection };
00475     // Wait for no work being dispatched or the end
00476     cv.wait(lock, [&] { return !work_to_dispatch || done; } );
00477     // Do not submit anything if we are in the shutdown process already
00478     if (!done) {
00479       work_to_dispatch = std::move(f);
00480       // Ask the graphics thread for some work
00481       dispatcher.emit();
00482     }
00483   };
00484
00485
00486 /** Update the image of a tile
00487
00488 \param[in] x is the tile horizontal id
00489
00490 \param[in] y is the tile vertical id
00491
00492 \param[in] data is a 2D MDspan of extent at most image_y by
00493 image_x.  Only the pixels of the extents are drawn
00494
00495 \param[in] min_value is the value represented with minimum of
00496 graphics palette color
00497
00498 \param[in] max_value is the value represented with maximum of
00499 graphics palette color
00500 */
00501   template <typename MDspan, typename RangeValue,
00502             /* Implement a cheap mdspan concept requirement, so that
00503 this function is not called when it is a pointer*/
00504             typename = typename MDspan::extents_type
00505             >
00506   void update_tile_data_image(int x, int y,
00507                               const MDspan &data,
00508                               RangeValue min_value,
00509                               RangeValue max_value) {
00510     /* Painful:  first I cannot use a std::unique_ptr because the
00511 std::function below needs to be copy-constructible, then the
00512 std::make_shared taking an array size comes only in C++20 while
00513 it is available for std::make_unique in C++17...  */
00514     std::shared_ptr<std::uint8_t[]> d { new std::uint8_t[3*image_x*image_y] };
00515     std::mdspan output { reinterpret_cast<rgb*>(d.get()), image_y, image_x };
00516     // For each pixel of the md_span or of the image, which one is smaller
00517     for (int j = 0; j < std::min(static_cast<int>(data.extent(0)), image_y);
00518          ++j)
00519       for (int i = 0; i < std::min(static_cast<int>(data.extent(1)), image_x);
00520            ++i) {
00521         /* Mirror the image vertically to display the pixels in a
00522 mathematical sense */
00523         output[image_y - 1 - j, i] =
00524             p.palettize(data[j, i], min_value, max_value);
00525       }
00526     // Send the graphics updating code
00527     submit([=, this] {
00528         // Create a first buffer, allowing later zooming
00529         auto pb = Gdk::Pixbuf::create_from_data(d.get()
00530                                                 , Gdk::Colorspace::COLORSPACE_RGB
00531                                                 , false//< has_alpha
00532                                                 , 8 //< bits_per_sample
00533                                                 , image_x //< width
00534                                                 , image_y //< height
00535                                                 , image_x*3 //< rowstride
00536                                                );
00537         // Update the pixel buffer of the image with some zooming
00538         images.at(x + nx*y).set(pb->scale_simple(image_x*zoom,
00539                                                  image_y*zoom,
00540                                                  Gdk::INTERP_NEAREST));
00541       });
00542   }
00543
00544
```

```
00545 /** Update the image of a tile of size image_y by image_x from a pointer
00546
00547 \param[in] x is the tile horizontal id
00548
00549 \param[in] y is the tile vertical id
00550
00551 \param[in] data is a pointer to the pixels to be drawns
00552
00553 \param[in] min_value is the value represented with minimum of
00554 graphics palette color
00555
00556 \param[in] max_value is the value represented with maximum of
00557 graphics palette color
00558 */
00559   template <typename DataType, typename RangeValue>
00560   void update_tile_data_image(int x, int y,
00561                               const DataType *data,
00562                               RangeValue min_value,
00563                               RangeValue max_value) {
00564     // Wrap the pointed area into an MDspan
00565     const std::mdspan md { data, image_y, image_x };
00566     update_tile_data_image(x, y, md, min_value, max_value);
00567   }
00568
00569
00570 /// Return the palette used to render the value
00571   palette &get_palette() {
00572     return p;
00573   }
00574
00575
00576 /// Test if the window has been closed after synchronizing with a
00577 /// barrier accross all the tiles
00578   bool is_done_barrier() {
00579     // Only 1 thread sample the graphics status
00580     if (done_has_been_sampled.test_and_set())
00581       done_snapshot = done;
00582     // Then wait for everybody to synchronize
00583     done_barrier.count_down_and_wait();
00584     // Reinitialize the flag for the next iteration
00585     done_has_been_sampled.clear();
00586     return done_snapshot;
00587   }
00588
00589 };
00590
00591
00592 /** A graphics application running in a separate thread to display
00593 images in a grid of tiles */
00594 struct application {
00595   std::thread t;
00596   std::unique_ptr<graphics::image_grid> w;
00597   bool initialized = false;
00598
00599 /** Start the graphics application
00600
00601 \param[inout] argc is the standard C program argument number
00602
00603 \param[inout] argv is the standard C program argument array
00604
00605 \param[in] nx is the numer of tiles horizontally
00606
00607 \param[in] ny is the numer of tiles vertically
00608
00609 \param[in] image_x is the horizontal image pixel size
00610
00611 \param[in] image_y is the vertical image pixel size
00612
00613 \param[in] zoom is the zooming factor applied to image pixels,
00614 both horizontally and vertically
00615
00616 \return a reference on *this to allow operation chaining
00617 */
00618   auto& start(int &argc, char **&argv,
00619               int nx, int ny, int image_x, int image_y, int zoom) {
00620     // To be sure not passing over the asynchronous graphics start
00621     std::promise<void> graphics_initialization;
00622
00623     /* Put all the graphics in its own thread.  Since
00624 Gtk::Application::create might modify argc and argv, capture by
00625 reference.  Since we have to wait for this thread, there should
00626 not be a read from freed memory issue.  */
00627     t = std::thread { [&]() mutable {
00628         // An application allowing several instance running at the same time
00629         auto a =
00630           Gtk::Application::create(argc, argv, "com.xilinx.trisycl.graphics",
00631                                    Gio::APPLICATION_NON_UNIQUE);
```

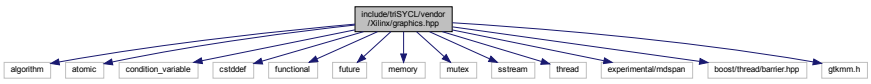
```
00632         /* Create the graphics object in this thread so the dispatcher
00633 is bound to this thread too */
00634         w.reset(new graphics::image_grid { nx, ny, image_x, image_y, zoom });
00635         w->set_close_action([&] {
00636             w->done = true;
00637           });
00638         // OK, the graphics system is in a usable state, unleash the main thread
00639         graphics_initialization.set_value();
00640         // Launch the graphics event loop handling with the graphics life
00641         a->run(*w);
00642 
00643         // Advertise that the graphics is shutting down
00644         w->done = true;
00645       } };
00646     // Wait for the graphics to start
00647     graphics_initialization.get_future().get();
00648 
00649     return *this;
00650   }
00651 
00652 
00653 ///  Wait for the graphics window to end
00654   void wait() {
00655     t.join();
00656   }
00657 
00658 
00659 /// Test if the window has been closed
00660   bool is_done() {
00661     return w->done;
00662   }
00663 
00664 
00665 /// Test if the window has been closed after synchronizing with a barrier
00666   bool is_done_barrier() {
00667     return w->is_done_barrier();
00668   }
00669 
00670 
00671 /** Update the image of a tile
00672 
00673 \param[in] x is the tile horizontal id
00674 
00675 \param[in] y is the tile vertical id
00676 
00677 \param[in] data is a 2D MDspan of extent at most image_y by
00678 image_x (only the pixels of the extents are drawn) or a pointer to
00679 a 2D linearized area of exactly image_y by image_x pixel values
00680 
00681 \param[in] min_value is the value represented with minimum of
00682 graphics palette color
00683 
00684 \param[in] max_value is the value represented with maximum of
00685 graphics palette color
00686 */
00687   template <typename DataType, typename RangeValue>
00688   void update_tile_data_image(int x, int y,
00689                               DataType data,
00690                               RangeValue min_value,
00691                               RangeValue max_value) {
00692     w->update_tile_data_image(x, y, data, min_value, max_value);
00693   };
00694 
00695 
00696 /// Return the image_grid in this application
00697   graphics::image_grid &image_grid() {
00698     return *w;
00699   }
00700 
00701 
00702 /// The destructor waiting for graphics to end
00703   ~application() {
00704     // If the graphics thread is still running, wait for it to exit
00705     if (t.joinable())
00706       wait();
00707   }
00708 
00709 };
00710 
00711 }
00712 
00713 /// @} End the graphics Doxygen group
00714 
00715 /*
00716 # Some Emacs stuff:
00717 ### Local Variables:
00718 ### ispell-local-dictionary:  "american"
```

```
00719 ### eval:  (flyspell-prog-mode)
00720 ### End:
00721 */
00722
00723 #endif // TRISYCL_SYCL_VENDOR_XILINX_GRAPHICS_HPP
```

13.417 include/triSYCL/vendor/Xilinx/graphics.hpp File Reference

```
#include <algorithm>
#include <atomic>
#include <condition_variable>
#include <cstddef>
#include <functional>
#include <future>
#include <memory>
#include <mutex>
#include <sstream>
#include <thread>
#include <experimental/mdspan>
#include <boost/thread/barrier.hpp>
#include <gtkmm.h>
```

Include dependency graph for graphics.hpp:



Classes

- struct trisycl::vendor::xilinx::graphics::frame_grid

  An application window displaying a grid of tiles.
- struct trisycl::vendor::xilinx::graphics::palette

  Color palette to project a linear data space to an RGB color space.
- struct trisycl::vendor::xilinx::graphics::image_grid

  An application window displaying a grid of tiled images.
- struct trisycl::vendor::xilinx::graphics::application

  A graphics application running in a separate thread to display images in a grid of tiles.

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::graphics

13.418 graphics.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_GRAPHICS_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_GRAPHICS_HPP
00003
00004 /** \file Some graphics windowing support useful for debugging
00005
00006 Based on GTK+3 with GTKMM 3 en C++ wrapper.
00007
00008 There are several graphical back-ends available with GTK.
00009
00010 An interesting one is the Broadway X11 back-end, allowing to
00011 display in a Web browser :-)
00012
00013 broadwayd :5 &
00014 xdg-open http://127.0.0.1:8085
00015 GDK_BACKEND=broadway BROADWAY_DISPLAY=:5 acap/wave_propagation
00016
00017 Ronan at Keryell point FR
00018
00019 This file is distributed under the University of Illinois Open Source
00020 License.  See LICENSE.TXT for details.
00021 */
00022
00023 #include <algorithm>
00024 #include <atomic>
00025 #include <condition_variable>
00026 #include <cstddef>
00027 #include <functional>
00028 #include <future>
00029 #include <memory>
00030 #include <mutex>
00031 #include <sstream>
00032 #include <thread>
00033
00034 #include <experimental/mdspan>
00035
00036 #include <boost/thread/barrier.hpp>
00037
00038 #include <gtkmm.h>
00039
00040 /** \defgroup graphics Graphics support for CGRA-like interaction
00041
00042 This can create a graphics application and typically update some
00043 sub-images on a frame
00044 @{
00045 */
00046
00047 namespace trisycl::vendor::xilinx::graphics {
00048
00049 // RGB 8 bit images, so 3 bytes per pixel
00050 using rgb = std::array<std::uint8_t, 3>;
00051
00052 /** An application window displaying a grid of tiles
00053
00054 Each tile is framed with the tile identifiers
00055 */
00056 struct frame_grid :  Gtk::ApplicationWindow {
00057 /// Nice to have some scroll bars around when the main window is too small
00058   Gtk::ScrolledWindow sw;
00059
00060 /// The container to represent the grid of tile images
00061   Gtk::Grid grid;
00062
00063 /** A close button in case the closing function is not provided by
00064 the window manager */
00065   Gtk::Button close_button { "Close" };
00066
00067 /** The linearized 2D vector of frames used to decorate the tile
00068 images with some frames with the tile names */
00069   std::vector<Gtk::Frame> frames;
00070
00071 /// Number of frame columns
00072   int nx;
00073
00074 /// Number of frame lines
00075   int ny;
00076
00077 /// An action to do when the window is closed
00078   std::function<void(void)> close_action;
00079
00080 /// Set to true by the closing handler
00081   std::atomic<bool> done = false;
00082
```

```
00083
00084 /** Create a grid of tiles
00085
00086 \param[in] nx is the numer of tiles horizontally
00087
00088 \param[in] ny is the numer of tiles vertically
00089 */
00090   frame_grid(int nx, int ny) :  nx { nx }, ny { ny } {
00091     set_default_size(1900, 1020);
00092
00093     add(sw);
00094     sw.add(grid);
00095     for (int y = 0; y < ny; ++y)
00096       for (int x = 0; x < nx; ++x) {
00097         std::ostringstream s;
00098         s « "Tile(" « x « ',' « y « ')';
00099         frames.emplace_back(s.str());
00100         frames.back().set_shadow_type(Gtk::SHADOW_ETCHED_OUT);
00101         // A minimal border to save space on main window
00102         frames.back().set_border_width(1);
00103         // Display the frame with the lower y South in a mathematical sense
00104         grid.attach(frames.back(), x, ny - y - 1, 1, 1);
00105       }
00106
00107     grid.add(close_button);
00108
00109     // Make the button the default widget
00110     close_button.set_can_default();
00111     close_button.grab_default();
00112
00113     // Connect the clicked signal of the close button to
00114     close_button.signal_clicked().connect([this] {
00115         // Unmap the window from the screen
00116         hide();
00117         // Call the handler if it does exist
00118         if (close_action)
00119           close_action();
00120       });
00121
00122     // Show all children of the window
00123     show_all_children();
00124   }
00125
00126
00127 /// Set a function to be called on close
00128   void set_close_action(std::function<void(void)> f) {
00129     close_action = f;
00130   }
00131
00132
00133 /// Get the frame at a given grid position
00134   auto &get_frame(int x, int y) {
00135     return frames.at(x + nx*y);
00136   }
00137
00138 };
00139
00140
00141 /** Color palette to project a linear data space to an RGB color space
00142
00143 Reimplement the palette concept by Nicolas Paris in PompC & HyperC
00144 back from the 90's...  :-)
00145 */
00146 struct palette {
00147   static constexpr int size = 256;
00148   std::array<rgb, size> color_mapping;
00149
00150   enum kind { gray, rainbow };
00151   kind k;
00152   int phase;
00153   int clip;
00154   int frequency_log2;
00155
00156 /** Create a palette
00157
00158 \param[in] k is the kind of palette
00159
00160 \param[in] phase is optional and allows the palette to have a
00161 phase shift instead of starting at 0
00162
00163 \param[in] clip is optional and specifies a value to be enhanced
00164 */
00165   palette(kind k = gray, int phase = 0, int frequency_log2 = 0, int clip = -1)
00166     :  k { k }, phase { phase } , clip { clip }, frequency_log2{frequency_log2} {
00167       update();
00168     }
00169
```

```
00170
00171   // Create a gray palette
00172   void gray_palette() {
00173     for (int i = 0; auto &e :  color_mapping) {
00174       if (i == clip)
00175         // Put a full red color for the selected value
00176         e = { 255, 0, 0 };
00177       else {
00178         int j;
00179         // There is no bidirectional shift operator in C++
00180         if (frequency_log2 < 0)
00181           // Divide the step by 2^{-fl}
00182           j = i » -frequency_log2;
00183         else
00184           // Multiply the step by 2^{fl}
00185           j = i « frequency_log2;
00186         auto v = static_cast<std::uint8_t>(j + phase);
00187         // A non-saturated "gray" of luminosity v
00188         e = { v,  v,  v };
00189       }
00190       ++i;
00191     }
00192   }
00193
00194
00195   // Create a rainbow palette
00196   void rainbow_palette() {
00197     Gdk::RGBA color;
00198     for (int i = 0; auto &e :  color_mapping) {
00199       // Put the selected value as black
00200       double luminosity = !(i == clip);
00201       int j;
00202       // There is no bidirectional shift operator in C++
00203       if (frequency_log2 < 0)
00204         // Divide the step by 2^{-fl}
00205         j = i » -frequency_log2;
00206       else
00207         // Multiply the step by 2^{fl}
00208         j = i « frequency_log2;
00209       auto v = static_cast<std::uint8_t>(j + phase);
00210       color.set_hsv(360.*v/size, 1, luminosity);
00211       e = { static_cast<std::uint8_t>(color.get_red_u() » 8),
00212             static_cast<std::uint8_t>(color.get_green_u() » 8),
00213             static_cast<std::uint8_t>(color.get_blue_u() » 8) };
00214       ++i;
00215     }
00216   }
00217
00218
00219   /* Update the palette and enforce some value invariants for
00220 defensive programming and GUI features */
00221   void update() {
00222     // Implement modulo arithmetic on interval [-1, size - 1]
00223     if (clip < -1)
00224       clip = size - 1;
00225     else if (clip >= size)
00226         clip = -1;
00227     // The frequency can only evolve in this interval
00228     frequency_log2 = std::clamp(frequency_log2, -8, 7);
00229     /* The phase can evolve modulo size.  When there are some negative
00230 numbers it is implementation dependent but that is enough for
00231 this GUI */
00232     phase = phase % size;
00233
00234     switch (k) {
00235       case gray:
00236         gray_palette();
00237         break;
00238       case rainbow:
00239         rainbow_palette();
00240         break;
00241     }
00242   }
00243
00244
00245 /** Set palette parameters and update
00246
00247 \param[in] new_k is the kind of palette
00248
00249 \param[in] new_phase allows the palette to have a phase shift
00250 instead of starting at 0
00251
00252 \param[in] new_clip specifies a value to be enhanced
00253 */
00254   void set(kind new_k, int new_phase, int new_frequency_log2, int new_clip) {
00255     k = new_k;
00256     phase = new_phase;
```

```
00257     clip = new_clip;
00258     frequency_log2 = new_frequency_log2;
00259     update();
00260   }
00261 
00262 
00263 /** Transform a value into an RGB color according to the palette
00264 */
00265   rgb palettize(double data, double min_value, double max_value) {
00266     std::uint8_t v = (data - min_value)*255/(max_value - min_value);
00267     // Write the same value for RGB to have a grey level
00268     return color_mapping[v];
00269   }
00270 
00271 
00272   void decrease_clip() {
00273     --clip;
00274     update();
00275   }
00276 
00277 
00278   void increase_clip() {
00279     ++clip;
00280     update();
00281   }
00282 
00283 
00284   void decrease_frequency() {
00285     --frequency_log2;
00286     update();
00287   }
00288 
00289 
00290   void increase_frequency() {
00291     ++frequency_log2;
00292     update();
00293   }
00294 
00295 
00296   void decrease_phase() {
00297     --phase;
00298     update();
00299   }
00300 
00301 
00302   void increase_phase() {
00303     ++phase;
00304     update();
00305   }
00306 
00307 
00308 /// Handle with any key pressed
00309   bool handle_key_press_event(GdkEventKey &event) {
00310     /* Only listen to keys without modifiers (no control, alt...)  but
00311 still allows shift */
00312     if (!(event.state
00313           & gtk_accelerator_get_default_mod_mask()
00314           & ~GDK_SHIFT_MASK)) {
00315       switch (event.keyval) {
00316       case GDK_KEY_0:
00317       case GDK_KEY_KP_0:
00318         k = gray;
00319         update();
00320         break;
00321       case GDK_KEY_1:
00322       case GDK_KEY_KP_1:
00323         k = rainbow;
00324         update();
00325         break;
00326       case GDK_KEY_minus:
00327       case GDK_KEY_KP_Subtract:
00328       case GDK_KEY_underscore:
00329         decrease_frequency();
00330         break;
00331       case GDK_KEY_plus:
00332       case GDK_KEY_KP_Add:
00333       case GDK_KEY_equal:
00334       case GDK_KEY_KP_Equal:
00335         increase_frequency();
00336         break;
00337       case GDK_KEY_h:
00338         increase_phase();
00339         break;
00340       case GDK_KEY_H:
00341         increase_phase();
00342         decrease_clip();
00343         break;
```

```
00344       case GDK_KEY_j:
00345         decrease_clip();
00346         break;
00347       case GDK_KEY_k:
00348         increase_clip();
00349         break;
00350       case GDK_KEY_l:
00351         decrease_phase();
00352         break;
00353       case GDK_KEY_L:
00354         decrease_phase();
00355         increase_clip();
00356         break;
00357       default:
00358         // Not a key press for us:  let the other listeners to handle
00359         return false;
00360       }
00361       // We have handled the event:  do not go on with any other listener
00362       return true;
00363     }
00364     // Not the right modifiers:  let the other listeners to handle
00365     return false;
00366   }
00367
00368 };
00369
00370
00371 /// An application window displaying a grid of tiled images
00372 struct image_grid :  frame_grid {
00373 /// The linearized 2D vector of images
00374   std::vector<Gtk::Image> images;
00375
00376 /// Horizontal image size in pixels
00377   int image_x;
00378
00379 /// Vertical image size in pixels
00380   int image_y;
00381
00382 /// The image pixel zooming factor for both dimension
00383   int zoom;
00384
00385 /// Dispatcher to invoke something in the graphics thread in a safe way
00386   Glib::Dispatcher dispatcher;
00387
00388 /// Protection against concurrent update
00389   std::mutex dispatch_protection;
00390
00391 /// The condition variable used to queue up graphics update submission
00392   std::condition_variable cv;
00393
00394 /// What to dispatch
00395   std::function<void(void)> work_to_dispatch;
00396
00397 /// The RGB palette used to render the image values
00398   palette p;
00399
00400 /// A global done state across all the tile to allow a global shutdown
00401   bool done_snapshot;
00402
00403 /// Set to true once "done" state has been sampled
00404   std::atomic_flag done_has_been_sampled;
00405
00406 /// Barrier to compute the global done state across the tiles
00407   boost::barrier done_barrier { static_cast<unsigned int>(nx*ny) };
00408
00409 /** Create a grid of tiled images
00410
00411 \param[in] nx is the numer of tiles horizontally
00412
00413 \param[in] ny is the numer of tiles vertically
00414
00415 \param[in] image_x is the horizontal image pixel size
00416
00417 \param[in] image_y is the vertical image pixel size
00418
00419 \param[in] zoom is the zooming factor applied to image pixels,
00420 both horizontally and vertically
00421 */
00422   image_grid(int nx, int ny, int image_x, int image_y, int zoom)
00423     :  frame_grid { nx , ny }
00424     , image_x { image_x }
00425     , image_y { image_y }
00426     , zoom { zoom }
00427   {
00428     for (int y = 0; y < ny; ++y)
00429       for (int x = 0; x < nx; ++x) {
00430         auto &f = get_frame(x, y);
```

```
00431         auto pb = Gdk::Pixbuf::create(Gdk::Colorspace::COLORSPACE_RGB
00432                                       , false //< has_alpha
00433                                       , 8 //< bits_per_sample
00434                                       , image_x*zoom //< width
00435                                       , image_y*zoom //< height
00436                                         );
00437         images.emplace_back(pb);
00438         // Display the frame with the lower y to the South
00439         f.add(images.back());
00440       }
00441
00442     // Hook a generic dispatcher
00443     dispatcher.connect([&] {
00444         {
00445           // Only 1 customer at a time
00446           std::lock_guard lock { dispatch_protection };
00447           // Skip the work when done to avoid dead lock
00448           if (done)
00449             // Wake-up everybody waiting for sending some work to
00450             // realize that they have to give up on their hope
00451             cv.notify_all();
00452           else
00453             work_to_dispatch();
00454           // Discard the previous work
00455           work_to_dispatch = nullptr;
00456         }
00457         // We can serve the next customer
00458         cv.notify_one();
00459       });
00460
00461     // Go live
00462     show_all_children();
00463   }
00464
00465
00466 /// Deal with the key pressed
00467   bool on_key_press_event(GdkEventKey* event)override {
00468     // Forward to the palette for action
00469     return p.handle_key_press_event(*event);
00470   }
00471
00472 /// Submit some work to the graphics thread
00473   void submit(std::function<void(void)> f) {
00474     std::unique_lock lock { dispatch_protection };
00475     // Wait for no work being dispatched or the end
00476     cv.wait(lock, [&] { return !work_to_dispatch || done; } );
00477     // Do not submit anything if we are in the shutdown process already
00478     if (!done) {
00479       work_to_dispatch = std::move(f);
00480       // Ask the graphics thread for some work
00481       dispatcher.emit();
00482     }
00483   };
00484
00485
00486 /** Update the image of a tile
00487
00488 \param[in] x is the tile horizontal id
00489
00490 \param[in] y is the tile vertical id
00491
00492 \param[in] data is a 2D MDspan of extent at most image_y by
00493 image_x.  Only the pixels of the extents are drawn
00494
00495 \param[in] min_value is the value represented with minimum of
00496 graphics palette color
00497
00498 \param[in] max_value is the value represented with maximum of
00499 graphics palette color
00500 */
00501   template <typename MDspan, typename RangeValue,
00502             /* Implement a cheap mdspan concept requirement, so that
00503 this function is not called when it is a pointer*/
00504             typename = typename MDspan::extents_type
00505             >
00506   void update_tile_data_image(int x, int y,
00507                               const MDspan &data,
00508                               RangeValue min_value,
00509                               RangeValue max_value) {
00510     /* Painful:  first I cannot use a std::unique_ptr because the
00511 std::function below needs to be copy-constructible, then the
00512 std::make_shared taking an array size comes only in C++20 while
00513 it is available for std::make_unique in C++17...  */
00514     std::shared_ptr<std::uint8_t[]> d { new std::uint8_t[3*image_x*image_y] };
00515     std::mdspan output { reinterpret_cast<rgb*>(d.get()), image_y, image_x };
00516     // For each pixel of the md_span or of the image, which one is smaller
00517     for (int j = 0; j < std::min(static_cast<int>(data.extent(0)), image_y);
```

```
00518          ++j)
00519       for (int i = 0; i < std::min(static_cast<int>(data.extent(1)), image_x);
00520            ++i) {
00521         /* Mirror the image vertically to display the pixels in a
00522 mathematical sense */
00523         output[image_y - 1 - j, i] =
00524             p.palettize(data[j, i], min_value, max_value);
00525       }
00526     // Send the graphics updating code
00527     submit([=, this] {
00528         // Create a first buffer, allowing later zooming
00529         auto pb = Gdk::Pixbuf::create_from_data(d.get()
00530                                                 , Gdk::Colorspace::COLORSPACE_RGB
00531                                                 , false//< has_alpha
00532                                                 , 8 //< bits_per_sample
00533                                                 , image_x //< width
00534                                                 , image_y //< height
00535                                                 , image_x*3 //< rowstride
00536                                                );
00537         // Update the pixel buffer of the image with some zooming
00538         images.at(x + nx*y).set(pb->scale_simple(image_x*zoom,
00539                                                  image_y*zoom,
00540                                                  Gdk::INTERP_NEAREST));
00541       });
00542   }
00543
00544
00545 /** Update the image of a tile of size image_y by image_x from a pointer
00546
00547 \param[in] x is the tile horizontal id
00548
00549 \param[in] y is the tile vertical id
00550
00551 \param[in] data is a pointer to the pixels to be drawns
00552
00553 \param[in] min_value is the value represented with minimum of
00554 graphics palette color
00555
00556 \param[in] max_value is the value represented with maximum of
00557 graphics palette color
00558 */
00559   template <typename DataType, typename RangeValue>
00560   void update_tile_data_image(int x, int y,
00561                               const DataType *data,
00562                               RangeValue min_value,
00563                               RangeValue max_value) {
00564     // Wrap the pointed area into an MDspan
00565     const std::mdspan md { data, image_y, image_x };
00566     update_tile_data_image(x, y, md, min_value, max_value);
00567   }
00568
00569
00570 /// Return the palette used to render the value
00571   palette &get_palette() {
00572     return p;
00573   }
00574
00575
00576 /// Test if the window has been closed after synchronizing with a
00577 /// barrier accross all the tiles
00578   bool is_done_barrier() {
00579     // Only 1 thread sample the graphics status
00580     if (done_has_been_sampled.test_and_set())
00581       done_snapshot = done;
00582     // Then wait for everybody to synchronize
00583     done_barrier.count_down_and_wait();
00584     // Reinitialize the flag for the next iteration
00585     done_has_been_sampled.clear();
00586     return done_snapshot;
00587   }
00588
00589 };
00590
00591
00592 /** A graphics application running in a separate thread to display
00593 images in a grid of tiles */
00594 struct application {
00595   std::thread t;
00596   std::unique_ptr<graphics::image_grid> w;
00597   bool initialized = false;
00598
00599 /** Start the graphics application
00600
00601 \param[inout] argc is the standard C program argument number
00602
00603 \param[inout] argv is the standard C program argument array
00604
```

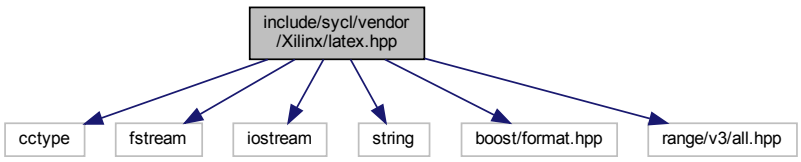
```
00605 \param[in] nx is the numer of tiles horizontally
00606
00607 \param[in] ny is the numer of tiles vertically
00608
00609 \param[in] image_x is the horizontal image pixel size
00610
00611 \param[in] image_y is the vertical image pixel size
00612
00613 \param[in] zoom is the zooming factor applied to image pixels,
00614 both horizontally and vertically
00615
00616 \return a reference on *this to allow operation chaining
00617 */
00618   auto& start(int &argc, char **&argv,
00619               int nx, int ny, int image_x, int image_y, int zoom) {
00620     // To be sure not passing over the asynchronous graphics start
00621     std::promise<void> graphics_initialization;
00622
00623     /* Put all the graphics in its own thread.  Since
00624 Gtk::Application::create might modify argc and argv, capture by
00625 reference.  Since we have to wait for this thread, there should
00626 not be a read from freed memory issue.  */
00627     t = std::thread { [&]() mutable {
00628         // An application allowing several instance running at the same time
00629         auto a =
00630           Gtk::Application::create(argc, argv, "com.xilinx.trisycl.graphics",
00631                                    Gio::APPLICATION_NON_UNIQUE);
00632         /* Create the graphics object in this thread so the dispatcher
00633 is bound to this thread too */
00634         w.reset(new graphics::image_grid { nx, ny, image_x, image_y, zoom });
00635         w->set_close_action([&] {
00636             w->done = true;
00637           });
00638         // OK, the graphics system is in a usable state, unleash the main thread
00639         graphics_initialization.set_value();
00640         // Launch the graphics event loop handling with the graphics life
00641         a->run(*w);
00642
00643         // Advertise that the graphics is shutting down
00644         w->done = true;
00645       } };
00646     // Wait for the graphics to start
00647     graphics_initialization.get_future().get();
00648
00649     return *this;
00650   }
00651
00652
00653 ///  Wait for the graphics window to end
00654   void wait() {
00655     t.join();
00656   }
00657
00658
00659 /// Test if the window has been closed
00660   bool is_done() {
00661     return w->done;
00662   }
00663
00664
00665 /// Test if the window has been closed after synchronizing with a barrier
00666   bool is_done_barrier() {
00667     return w->is_done_barrier();
00668   }
00669
00670
00671 /** Update the image of a tile
00672
00673 \param[in] x is the tile horizontal id
00674
00675 \param[in] y is the tile vertical id
00676
00677 \param[in] data is a 2D MDspan of extent at most image_y by
00678 image_x (only the pixels of the extents are drawn) or a pointer to
00679 a 2D linearized area of exactly image_y by image_x pixel values
00680
00681 \param[in] min_value is the value represented with minimum of
00682 graphics palette color
00683
00684 \param[in] max_value is the value represented with maximum of
00685 graphics palette color
00686 */
00687   template <typename DataType, typename RangeValue>
00688   void update_tile_data_image(int x, int y,
00689                               DataType data,
00690                               RangeValue min_value,
00691                               RangeValue max_value) {
```

```
00692     w->update_tile_data_image(x, y, data, min_value, max_value);
00693   };
00694
00695
00696 /// Return the image_grid in this application
00697   graphics::image_grid &image_grid() {
00698     return *w;
00699   }
00700
00701
00702 /// The destructor waiting for graphics to end
00703   ~application() {
00704     // If the graphics thread is still running, wait for it to exit
00705     if (t.joinable())
00706       wait();
00707   }
00708
00709 };
00710
00711 }
00712
00713 /// @} End the graphics Doxygen group
00714
00715 /*
00716 # Some Emacs stuff:
00717 ### Local Variables:
00718 ### ispell-local-dictionary:  "american"
00719 ### eval:  (flyspell-prog-mode)
00720 ### End:
00721 */
00722
00723 #endif // TRISYCL_SYCL_VENDOR_XILINX_GRAPHICS_HPP
```

13.419 include/sycl/vendor/Xilinx/latex.hpp File Reference

```
#include <cctype>
#include <fstream>
#include <iostream>
#include <string>
#include <boost/format.hpp>
#include <range/v3/all.hpp>
```

Include dependency graph for latex.hpp:



Classes

- class trisycl::vendor::xilinx::latex::context

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::latex

13.420 latex.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_LATEX_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_LATEX_HPP
00003
00004 /** \file Some LaTeX output support useful for debugging
00005
00006 Based on TikZ/PGF
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include <cctype>
00015 #include <fstream>
00016 #include <iostream>
00017 #include <string>
00018
00019 #include <boost/format.hpp>
00020 #include <range/v3/all.hpp>
00021
00022 /** \defgroup latex LaTeX support for graphics output
00023
00024 This can generate LaTeX output used to display internal
00025 configuration for debug or teaching purpose
00026
00027 @{
00028 */
00029
00030 namespace trisycl::vendor::xilinx::latex {
00031
00032 class context {
00033
00034 public:
00035
00036   // There is a limit in TeX for the dimensions to be under 575mm
00037   static double constexpr max_size = 574;
00038
00039 /// Scale all the coordinates by this factor to be under the TeX max size
00040   double scaling_factor;
00041
00042 /// Size of the drawing in the original units (unscaled)
00043   vec<int, 2> size;
00044
00045 /// Name of the file to write into
00046   std::string file_name;
00047
00048 /// Store the LaTeX output
00049   std::string out;
00050
00051 /// Keep track whether the content is finalized for final user consumption
00052   bool finalized = false;
00053
00054 /** Create a graphics context
00055
00056 \param[in] s is the 2D size of the drawing to be generated in mm
00057
00058 \param[in] f_name is the file name of the LaTeX to be generated
00059 into or standard output by default
00060 */
00061   context(const vec<int, 2>& s, const std::string& f_name = {})
00062     :  size { s }, file_name { f_name } {
00063     // Compute the scaling factor to remain under the TeX limit
00064     auto m = std::max(size.x(), size.y());
00065     scaling_factor = m > max_size ?  max_size/m :  1;
00066     // Generate the LaTeX header
00067     out = (boost::format { R"(%% To be compiled with lualatex instead of pdflatex
00068 %% to avoid a bug on _ and to handle huge memory automatically.
00069 \documentclass{article}
00070 %% Use maximum of the page surface
00071 \usepackage[paperwidth=%1%mm,paperheight=%2%mm,top=0mm,bottom=0mm,
00072 left=0mm,right=0mm]{geometry}
00073 %% Use a font allowing arbitrary size
00074 \usepackage{lmodern}
00075 %% The turbo-charged graphics package
00076 \usepackage{tikz}
00077 \usetikzlibrary{backgrounds,calc,decorations.pathmorphing,fit,patterns,mindmap}
00078 \usepackage{tikzlings}
00079 \definecolor{orangeSYCL}{RGB}{242,104,34}
00080 %% Some cool palettes
00081 \usepackage{xcolor-material}
00082 \usepackage{xcolor-solarized}
```

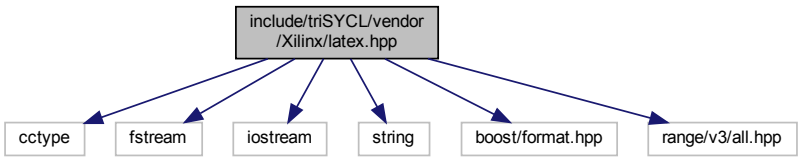
```
00083 %% Consider '_' as a (almost) normal character
00084 \usepackage[strings]{underscore}
00085
00086 \begin{document}
00087 %% No page number, header or footer
00088 \thispagestyle{empty}
00089 %% Skip the usual space at the beginning of a paragraph
00090 \noindent
00091 %% Use a super small font.  Use sans-serif for readability
00092 \fontsize{%3%}{%3%}\selectfont\sffamily
00093 %% Use remembering in every picture so we can use named coordinates
00094 %% across them
00095 \tikzstyle{every picture}+=[remember picture]
00096 \begin{tikzpicture}[%% Scale by 0.1, so the unit is 1mm instead of default 1cm
00097 scale = 0.1,
00098 %% Default style
00099 gray,
00100 style = {line width = %4%mm, ->}]
00101
00102 )" } % scale(size.x()) % scale(size.y()) % scale(1) % scale(0.01)).str();
00103 }
00104
00105
00106 /// Add textual LaTeX content to the context
00107   void add(const std::string& content) {
00108     out += content;
00109   }
00110
00111
00112   // Scale a coordinate according to the scaling factor
00113   double scale(double v)const {
00114     return v*scaling_factor;
00115   }
00116
00117 private:
00118
00119 /// Finalize the content before using it
00120   void finalize() {
00121     if (!finalized) {
00122       // Add the LaTeX epilogue
00123       out += R"(
00124 \end{tikzpicture}
00125
00126 \end{document})";
00127 }
00128 }
00129
00130 public:
00131
00132 /** Display the content
00133
00134 Display the LaTeX textual content on or into a file, to be
00135 compiled for example with lualatex
00136 */
00137   void display() {
00138     finalize();
00139     if (file_name.empty())
00140       std::cerr « out « std::endl;
00141     else
00142       std::ofstream { file_name, std::ios::trunc } « out;
00143   }
00144
00145
00146 /** Build a TikZ/PGF valid name string by concatenating a prefix and a
00147 capitalized node name after having removed any '_'
00148
00149 \input[in] node_name is a string-like TikZ/PGF that may contain some '_'
00150
00151 \input[in] prefix is an optional string-like prefix
00152 */
00153   static std::string clean_node(const std::string_view& node_name,
00154                                 const std::string_view& prefix = "") {
00155     // All the characters but the '_'
00156     auto without_ = ranges::views::remove(node_name, '_');
00157     return ranges::views::concat
00158       (ranges::views::all(prefix),
00159        // Take the first character if any and upper-case it
00160        without_ | ranges::views::take(1)
00161        | ranges::views::transform([] (auto c) { return std::toupper(c); }),
00162        // And take all the character after the first one if any
00163        ranges::views::drop(without_, 1))
00164       | ranges::to<std::string>;
00165   }
00166
00167 };
00168
00169 }
```

```
00170
00171 /// @} End the latex Doxygen group
00172
00173 /*
00174 # Some Emacs stuff:
00175 ### Local Variables:
00176 ### ispell-local-dictionary:  "american"
00177 ### eval:  (flyspell-prog-mode)
00178 ### End:
00179 */
00180
00181 #endif // TRISYCL_SYCL_VENDOR_XILINX_LATEX_HPP
```

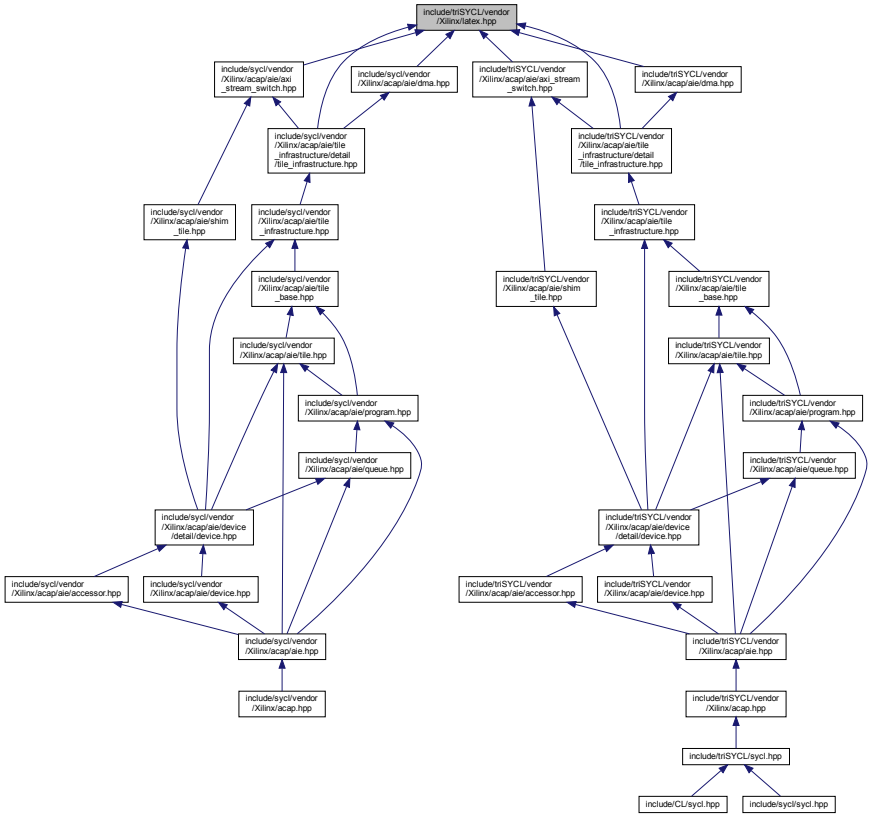
13.421 include/triSYCL/vendor/Xilinx/latex.hpp File Reference

```
#include <cctype>
#include <fstream>
#include <iostream>
#include <string>
#include <boost/format.hpp>
#include <range/v3/all.hpp>
```

Include dependency graph for latex.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class trisycl::vendor::xilinx::latex::context

Namespaces

- namespace trisycl

  The vector type to be used as SYCL vector.
- namespace trisycl::vendor
- namespace trisycl::vendor::xilinx
- namespace trisycl::vendor::xilinx::latex

13.422 latex.hpp

Go to the documentation of this file.

```
00001 #ifndef TRISYCL_SYCL_VENDOR_XILINX_LATEX_HPP
00002 #define TRISYCL_SYCL_VENDOR_XILINX_LATEX_HPP
00003
00004 /** \file Some LaTeX output support useful for debugging
00005
00006 Based on TikZ/PGF
00007
00008 Ronan dot Keryell at Xilinx dot com
00009
```

```
00010 This file is distributed under the University of Illinois Open Source
00011 License.  See LICENSE.TXT for details.
00012 */
00013
00014 #include <cctype>
00015 #include <fstream>
00016 #include <iostream>
00017 #include <string>
00018
00019 #include <boost/format.hpp>
00020 #include <range/v3/all.hpp>
00021
00022 /** \defgroup latex LaTeX support for graphics output
00023
00024 This can generate LaTeX output used to display internal
00025 configuration for debug or teaching purpose
00026
00027 @{
00028 */
00029
00030 namespace trisycl::vendor::xilinx::latex {
00031
00032 class context {
00033
00034 public:
00035
00036   // There is a limit in TeX for the dimensions to be under 575mm
00037   static double constexpr max_size = 574;
00038
00039 /// Scale all the coordinates by this factor to be under the TeX max size
00040   double scaling_factor;
00041
00042 /// Size of the drawing in the original units (unscaled)
00043   vec<int, 2> size;
00044
00045 /// Name of the file to write into
00046   std::string file_name;
00047
00048 /// Store the LaTeX output
00049   std::string out;
00050
00051 /// Keep track whether the content is finalized for final user consumption
00052   bool finalized = false;
00053
00054 /** Create a graphics context
00055
00056 \param[in] s is the 2D size of the drawing to be generated in mm
00057
00058 \param[in] f_name is the file name of the LaTeX to be generated
00059 into or standard output by default
00060 */
00061   context(const vec<int, 2>& s, const std::string& f_name = {})
00062     :  size { s }, file_name { f_name } {
00063     // Compute the scaling factor to remain under the TeX limit
00064     auto m = std::max(size.x(), size.y());
00065     scaling_factor = m > max_size ?  max_size/m :  1;
00066     // Generate the LaTeX header
00067     out = (boost::format { R"(%% To be compiled with lualatex instead of pdflatex
00068 %% to avoid a bug on _ and to handle huge memory automatically.
00069 \documentclass{article}
00070 %% Use maximum of the page surface
00071 \usepackage[paperwidth=%1%mm,paperheight=%2%mm,top=0mm,bottom=0mm,
00072 left=0mm,right=0mm]{geometry}
00073 %% Use a font allowing arbitrary size
00074 \usepackage{lmodern}
00075 %% The turbo-charged graphics package
00076 \usepackage{tikz}
00077 \usetikzlibrary{backgrounds,calc,decorations.pathmorphing,fit,patterns,mindmap}
00078 \usepackage{tikzlings}
00079 \definecolor{orangeSYCL}{RGB}{242,104,34}
00080 %% Some cool palettes
00081 \usepackage{xcolor-material}
00082 \usepackage{xcolor-solarized}
00083 %% Consider '_' as a (almost) normal character
00084 \usepackage[strings]{underscore}
00085
00086 \begin{document}
00087 %% No page number, header or footer
00088 \thispagestyle{empty}
00089 %% Skip the usual space at the beginning of a paragraph
00090 \noindent
00091 %% Use a super small font.  Use sans-serif for readability
00092 \fontsize{%3%}{%3%}\selectfont\sffamily
00093 %% Use remembering in every picture so we can use named coordinates
00094 %% across them
00095 \tikzstyle{every picture}+=[remember picture]
00096 \begin{tikzpicture}[%% Scale by 0.1, so the unit is 1mm instead of default 1cm
```

```
00097 scale = 0.1,
00098 %% Default style
00099 gray,
00100 style = {line width = %4%mm, ->}]
00101
00102 )" } % scale(size.x()) % scale(size.y()) % scale(1) % scale(0.01)).str();
00103 }
00104
00105
00106 /// Add textual LaTeX content to the context
00107   void add(const std::string& content) {
00108     out += content;
00109   }
00110
00111
00112   // Scale a coordinate according to the scaling factor
00113   double scale(double v)const {
00114     return v*scaling_factor;
00115   }
00116
00117 private:
00118
00119 /// Finalize the content before using it
00120   void finalize() {
00121     if (!finalized) {
00122       // Add the LaTeX epilogue
00123       out += R"(
00124 \end{tikzpicture}
00125
00126 \end{document})";
00127 }
00128 }
00129
00130 public:
00131
00132 /** Display the content
00133
00134 Display the LaTeX textual content on or into a file, to be
00135 compiled for example with lualatex
00136 */
00137   void display() {
00138     finalize();
00139     if (file_name.empty())
00140       std::cerr « out « std::endl;
00141     else
00142       std::ofstream { file_name, std::ios::trunc } « out;
00143   }
00144
00145
00146 /** Build a TikZ/PGF valid name string by concatenating a prefix and a
00147 capitalized node name after having removed any '_'
00148
00149 \input[in] node_name is a string-like TikZ/PGF that may contain some '_'
00150
00151 \input[in] prefix is an optional string-like prefix
00152 */
00153   static std::string clean_node(const std::string_view& node_name,
00154                                 const std::string_view& prefix = "") {
00155     // All the characters but the '_'
00156     auto without_ = ranges::views::remove(node_name, '_');
00157     return ranges::views::concat
00158       (ranges::views::all(prefix),
00159        // Take the first character if any and upper-case it
00160        without_ | ranges::views::take(1)
00161        | ranges::views::transform([] (auto c) { return std::toupper(c); }),
00162        // And take all the character after the first one if any
00163        ranges::views::drop(without_, 1))
00164       | ranges::to<std::string>;
00165   }
00166
00167 };
00168
00169 }
00170
00171 /// @} End the latex Doxygen group
00172
00173 /*
00174 # Some Emacs stuff:
00175 ### Local Variables:
00176 ### ispell-local-dictionary:  "american"
00177 ### eval:  (flyspell-prog-mode)
00178 ### End:
00179 */
00180
00181 #endif // TRISYCL_SYCL_VENDOR_XILINX_LATEX_HPP
```

Index

__SYCL_SINGLE_SOURCE__
Manage default configuration and types, 670
__attribute__
opencl_spir.h, 1486, 1499
opencl_spir_req.h, 1675
__constant
opencl_spir_req.h, 1674
__generic
opencl_spir_req.h, 1674
__global
opencl_spir_req.h, 1674
__kernel_exec
opencl_spir.h, 1454
__local
opencl_spir_req.h, 1674
__private
opencl_spir_req.h, 1675
__swizzled_base_vec__
trisycl::detail, 849
__swizzled_vec__
Vector types in SYCL, 817
_ssdm_SpecArrayPartition
Xilinx vendor extensions, 431
_ssdm_op_SpecDataflowPipeline
Xilinx vendor extensions, 430
_ssdm_op_SpecPipeline
Xilinx vendor extensions, 431
~accessor_base
trisycl::detail::accessor_base, 320
~application
trisycl::vendor::xilinx::graphics::application, 888
~buffer
trisycl::detail::buffer< T, Dimensions >, 350
~buffer_base
trisycl::detail::buffer_base, 358
~buffer_waiter
trisycl::detail::buffer_waiter< T, Dimensions, Allocator >, 369
~communicator_port
trisycl::vendor::xilinx::acap::aie::communicator_port, 917, 918
~context
trisycl::detail::context, 532
~device
trisycl::detail::device, 548
~device_selector
trisycl::device_selector, 552
~dma
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >, 1019, 1020
~error_handler
trisycl::error_handler, 684
~event
trisycl::detail::event, 1030
~fiber_pool
trisycl::detail::fiber_pool, 1041
~opencl_context
trisycl::detail::opencl_context, 1113
~opencl_device
trisycl::detail::opencl_device, 1119
~opencl_event
trisycl::detail::opencl_event, 1127
~opencl_platform
trisycl::detail::opencl_platform, 588
~opencl_queue
trisycl::detail::opencl_queue, 1138
~pipe_accessor
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 468
~pipe_reservation
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 489
~platform
trisycl::detail::platform, 592
~pooled_work_stealing
trisycl::detail::pooled_work_stealing, 1183
~program
trisycl::detail::program, 1199
~queue
trisycl::detail::queue, 1212
~router_minion
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1231, 1232

a
opencl_spir.h, 1510
trisycl::vec< DataType, 4 >, 800
Vector types in SYCL, 817
accelerator
Platforms, contexts, accessors and queues, 610
access
trisycl::mixin::accessor< T, Dimensions >, 523
access_mode
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 310
accessor
trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >, 436

trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >, 434
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 308, 309
trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >, 322
trisycl::detail::accessor< T, Dimensions, Mode, Target >, 339, 340
trisycl::drt::accessor< Accessor >, 677
trisycl::host_accessor< DataType, Dimensions, AccessMode >, 439, 440
trisycl::mixin::accessor< T, Dimensions >, 516, 517
trisycl::vendor::xilinx::acap::aie::accessor< Buffer >, 879–883
accessor_detail
trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >, 436
trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >, 433
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 308
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 478
accessor_type
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 487
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 478
AccessorMixIn, 884
accessors
trisycl::detail::task, 1290
acquire
trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device, 1107
acquire_with_value
trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device, 1107, 1108
add
trisycl::vendor::xilinx::latex::context, 931
add_buffer
trisycl::detail::task, 1279
add_buffer_to_task
Data access and storage in SYCL, 375
trisycl::detail, 850
add_postlude
trisycl::detail::task, 1279
add_prelude
trisycl::detail::task, 1280
add_to_task
trisycl::detail::buffer_base, 359
addproperty
trisycl::property_list, 1208
addr_space
Dealing with OpenCL address spaces, 640
address_bits
Platforms, contexts, accessors and queues, 607
address_space
trisycl::access, 841
trisycl::detail::address_space_base< T, AS >, 637
address_space.hpp
TRISYCL_CONSTANT_AS, 1343
TRISYCL_GENERIC_AS, 1343
TRISYCL_GLOBAL_AS, 1343
TRISYCL_LOCAL_AS, 1343
TRISYCL_PRIVATE_AS, 1343
address_space_array
trisycl::detail::address_space_array< T, AS >, 627
address_space_fundamental
trisycl::detail::address_space_fundamental< T, AS >, 630
address_space_object
trisycl::detail::address_space_object< T, AS >, 632
address_space_ptr
trisycl::detail::address_space_ptr< T, AS >, 635
address_space_variable
trisycl::detail::address_space_variable< T, AS >, 639
AI Engine CGRA, 43
aie_d
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 152
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >, 161
alignment_v
trisycl::detail, 853
all
Platforms, contexts, accessors and queues, 610
all_true
trisycl::detail, 849
alloc
trisycl::detail::buffer< T, Dimensions >, 355
allocate_buffer
trisycl::detail::buffer< T, Dimensions >, 351
allocation
trisycl::detail::buffer< T, Dimensions >, 355
allocator_type
trisycl::buffer< T, Dimensions, Allocator >, 326
allow_any_dimension
trisycl::detail, 853
apply_binary_functor
trisycl::detail::vec< DataType, NumElements >, 1306
apply_binary_functor_impl
trisycl::detail::vec< DataType, NumElements >, 1306
apply_ternary_functor_impl
trisycl::detail::vec< DataType, NumElements >, 1307
apply_unary_functor
trisycl::detail::vec< DataType, NumElements >, 1307
apply_unary_functor_impl
trisycl::detail::vec< DataType, NumElements >, 1308

array_size
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >, 429
as
trisycl::detail::vec< DataType, NumElements >, 1309
as_char
opencl_spir.h, 1454
as_char16
opencl_spir.h, 1454
as_char2
opencl_spir.h, 1455
as_char3
opencl_spir.h, 1455
as_char4
opencl_spir.h, 1455
as_char8
opencl_spir.h, 1455
as_double
opencl_spir.h, 1455
as_double16
opencl_spir.h, 1455
as_double2
opencl_spir.h, 1456
as_double3
opencl_spir.h, 1456
as_double4
opencl_spir.h, 1456
as_double8
opencl_spir.h, 1456
as_float
opencl_spir.h, 1456
as_float16
opencl_spir.h, 1456
as_float2
opencl_spir.h, 1457
as_float3
opencl_spir.h, 1457
as_float4
opencl_spir.h, 1457
as_float8
opencl_spir.h, 1457
as_half
opencl_spir.h, 1457
as_half16
opencl_spir.h, 1457
as_half2
opencl_spir.h, 1458
as_half3
opencl_spir.h, 1458
as_half4
opencl_spir.h, 1458
as_half8
opencl_spir.h, 1458
as_int
opencl_spir.h, 1458
as_int16
opencl_spir.h, 1458
as_int2
opencl_spir.h, 1459
as_int3
opencl_spir.h, 1459
as_int4
opencl_spir.h, 1459
as_int8
opencl_spir.h, 1459
as_long
opencl_spir.h, 1459
as_long16
opencl_spir.h, 1459
as_long2
opencl_spir.h, 1460
as_long3
opencl_spir.h, 1460
as_long4
opencl_spir.h, 1460
as_long8
opencl_spir.h, 1460
as_short
opencl_spir.h, 1460
as_short16
opencl_spir.h, 1460
as_short2
opencl_spir.h, 1461
as_short3
opencl_spir.h, 1461
as_short4
opencl_spir.h, 1461
as_short8
opencl_spir.h, 1461
as_uchar
opencl_spir.h, 1461
as_uchar16
opencl_spir.h, 1461
as_uchar2
opencl_spir.h, 1462
as_uchar3
opencl_spir.h, 1462
as_uchar4
opencl_spir.h, 1462
as_uchar8
opencl_spir.h, 1462
as_uint
opencl_spir.h, 1462
as_uint16
opencl_spir.h, 1462
as_uint2
opencl_spir.h, 1463
as_uint3
opencl_spir.h, 1463
as_uint4
opencl_spir.h, 1463
as_uint8
opencl_spir.h, 1463
as_ulong
opencl_spir.h, 1463

as_ulong16
opencl_spir.h, 1463
as_ulong2
opencl_spir.h, 1464
as_ulong3
opencl_spir.h, 1464
as_ulong4
opencl_spir.h, 1464
as_ulong8
opencl_spir.h, 1464
as_ushort
opencl_spir.h, 1464
as_ushort16
opencl_spir.h, 1464
as_ushort2
opencl_spir.h, 1465
as_ushort3
opencl_spir.h, 1465
as_ushort4
opencl_spir.h, 1465
as_ushort8
opencl_spir.h, 1465
assign
trisycl::detail::buffer< T, Dimensions >, 351
assign_from_tuple
trisycl::detail, 850
assume_validity
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 489
async_handler
Error handling, 708
atomic
trisycl::access, 842
automatic
Platforms, contexts, accessors and queues, 610
trisycl, 833
awakened
trisycl::detail::pooled_shared_work, 1177
trisycl::detail::pooled_work_stealing, 1184
axi_packet
trisycl::vendor::xilinx::acap::aie::axi_packet, 73, 74
axi_ss
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1239
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 294
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, 1196
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, 1226
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 179
axi_ss_geo
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 48
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 268
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 163
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 246, 247
axi_ss_t
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 268
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 164
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 247
axi_stream_switch
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 49

b
opencl_spir.h, 1510
trisycl::vec< DataType, 4 >, 800
Vector types in SYCL, 817
barrier
trisycl::h_item< Dimensions >, 727
trisycl::nd_item< Dimensions >, 754
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 187
barrier_
trisycl::detail::pooled_work_stealing::pool_ctx, 1173
base
trisycl::host_accessor< DataType, Dimensions, AccessMode >, 438
trisycl::vendor::xilinx::acap::aie::accessor< Buffer >, 879
trisycl::vendor::xilinx::acap::aie::connection::base, 896, 897
trisycl::vendor::xilinx::acap::aie::connection::input, 1097–1099
trisycl::vendor::xilinx::acap::aie::connection::output, 1143–1145
base_vec
trisycl::vec< DataType, 1 >, 788
trisycl::vec< DataType, 16 >, 810
trisycl::vec< DataType, 2 >, 790
trisycl::vec< DataType, 3 >, 794
trisycl::vec< DataType, 4 >, 799
trisycl::vec< DataType, 8 >, 806
basic_type
trisycl::detail::vec< DataType, NumElements >, 1305
begin
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 310
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 490
trisycl::facade::accessor< AccessorMixIn >, 506
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 481
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >, 425, 426
binary

trisycl::drt::code::program, 1200
binary_size
trisycl::drt::code::program, 1201
bit_ceil_at_least_2
trisycl::pipe< Name, T, MinCapacity, SizeT >, 372
bit_size
trisycl::vendor::trisycl::random::xorshift< Nbits >, 1322
bli_in
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 166, 167
bli_in_connection
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 167, 168
bli_out
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 169, 170
bli_out_connection
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 170, 171
block
trisycl::vendor::xilinx::partition, 871
blocking
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 475
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 494
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 484
blocking_pipe
trisycl::access, 844
BOOST_COMPUTE_CL_VERSION_1_2
Manage default configuration and types, 670
buf
trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >, 323
trisycl::detail::accessor< T, Dimensions, Mode, Target >, 345
buffer
Data access and storage in SYCL, 375, 376
trisycl::buffer< T, Dimensions, Allocator >, 326–331
trisycl::detail::buffer< T, Dimensions >, 348–350
trisycl::drt::accessor< Accessor >, 678
buffer_add_to_task
Data access and storage in SYCL, 376
buffer_allocator
Data access and storage in SYCL, 374
buffer_base
trisycl::detail::buffer_base, 358
buffer_cache
trisycl::detail::buffer_base, 366
buffer_waiter
trisycl::detail::buffer_waiter< T, Dimensions, Allocator >, 369
buffers_in_use
trisycl::detail::task, 1290
build_with_kernel_type
trisycl::program, 1204
built_in_kernels
Platforms, contexts, accessors and queues, 608

c
opencl_spir.h, 1510
trisycl::detail::cache< Key, Value >, 905
trisycl::detail::opencl_context, 1117
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1239
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >, 1026
trisycl::vendor::xilinx::acap::aie::fifo_channel, 1053
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, 1197
cache
trisycl::detail::opencl_context, 1117
trisycl::detail::opencl_device, 1125
trisycl::detail::opencl_event, 1131
trisycl::detail::opencl_kernel, 1135
trisycl::detail::opencl_platform, 591
trisycl::detail::opencl_queue, 1141
call_update_buffer_state
trisycl::detail::buffer< T, Dimensions >, 351, 352
capacity
trisycl::detail::sycl_2_2::pipe< T >, 449
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 469
trisycl::sycl_2_2::pipe< T >, 443
trisycl::sycl_2_2::static_pipe< T, Capacity >, 497
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1239
trisycl::vendor::xilinx::acap::aie::fifo_channel, 1053
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, 1197
capture_arg_v
Expressing parallelism through kernels, 770
cascade
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 973
trisycl::vendor::xilinx::acap::aie::device< Layout >, 998
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 139
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >, 234, 235
cascade_linear_id
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 85
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 188
cascade_linear_x
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 85, 86
cascade_linear_y
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 87, 88
cascade_stream_pipes

trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >, 71
cass
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 968
trisycl::vendor::xilinx::acap::aie::device< Layout >, 994
cb
trisycl::detail::sycl_2_2::pipe< T >, 463
cb_mutex
trisycl::detail::sycl_2_2::pipe< T >, 463
cbegin
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 311
trisycl::facade::accessor< AccessorMixIn >, 506
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 481
cend
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 311
trisycl::facade::accessor< AccessorMixIn >, 507
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 482
cgh
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >, 915
cgh_t
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t, 907
channel
trisycl::pipe< Name, T, MinCapacity, SizeT >, 374
CHAR_BIT
opencl_spir.h, 1465
CHAR_MAX
opencl_spir.h, 1465
CHAR_MIN
opencl_spir.h, 1466
check
trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >, 298, 299
cl, 823
cl::sycl, 823
cl_code
trisycl::cl_exception, 689
cl_exception
trisycl::cl_exception, 689
cl_mem_fence_flags
opencl_spir.h, 1497
CL_SYCL_LANGUAGE_VERSION
Manage default configuration and types, 670
clean_node
trisycl::vendor::xilinx::latex::context, 931
clip
trisycl::vendor::xilinx::graphics::palette, 1168
CLK_A
opencl_spir.h, 1466
CLK_ADDRESS_CLAMP
opencl_spir.h, 1466
CLK_ADDRESS_CLAMP_TO_EDGE
opencl_spir.h, 1466
CLK_ADDRESS_MIRRORED_REPEAT
opencl_spir.h, 1466
CLK_ADDRESS_NONE
opencl_spir.h, 1466
CLK_ADDRESS_REPEAT
opencl_spir.h, 1467
CLK_ARGB
opencl_spir.h, 1467
CLK_BGRA
opencl_spir.h, 1467
CLK_DEPTH
opencl_spir.h, 1467
CLK_DEPTH_STENCIL
opencl_spir.h, 1467
CLK_FILTER_LINEAR
opencl_spir.h, 1468
CLK_FILTER_NEAREST
opencl_spir.h, 1468
CLK_FLOAT
opencl_spir.h, 1468
CLK_GLOBAL_MEM_FENCE
opencl_spir.h, 1468
CLK_HALF_FLOAT
opencl_spir.h, 1468
CLK_INTENSITY
opencl_spir.h, 1468
CLK_LOCAL_MEM_FENCE
opencl_spir.h, 1469
CLK_LUMINANCE
opencl_spir.h, 1469
CLK_NORMALIZED_COORDS_FALSE
opencl_spir.h, 1469
CLK_NORMALIZED_COORDS_TRUE
opencl_spir.h, 1469
CLK_R
opencl_spir.h, 1469
CLK_RA
opencl_spir.h, 1469
CLK_RG
opencl_spir.h, 1470
CLK_RGB
opencl_spir.h, 1470
CLK_RGBA
opencl_spir.h, 1470
CLK_RGBx
opencl_spir.h, 1470
CLK_RGx
opencl_spir.h, 1470
CLK_Rx
opencl_spir.h, 1470
CLK_SIGNED_INT16
opencl_spir.h, 1471
CLK_SIGNED_INT32
opencl_spir.h, 1471
CLK_SIGNED_INT8

opencl_spir.h, 1471
CLK_SNORM_INT16
opencl_spir.h, 1471
CLK_SNORM_INT8
opencl_spir.h, 1471
CLK_UNORM_INT16
opencl_spir.h, 1471
CLK_UNORM_INT24
opencl_spir.h, 1472
CLK_UNORM_INT8
opencl_spir.h, 1472
CLK_UNORM_INT_101010
opencl_spir.h, 1472
CLK_UNORM_SHORT_555
opencl_spir.h, 1472
CLK_UNORM_SHORT_565
opencl_spir.h, 1472
CLK_UNSIGNED_INT16
opencl_spir.h, 1472
CLK_UNSIGNED_INT32
opencl_spir.h, 1473
CLK_UNSIGNED_INT8
opencl_spir.h, 1473
close
trisycl::detail::fiber_pool, 1042
close_action
trisycl::vendor::xilinx::graphics::frame_grid, 1058
close_button
trisycl::vendor::xilinx::graphics::frame_grid, 1059
cmp
opencl_spir.h, 1510
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 968
trisycl::vendor::xilinx::acap::aie::device< Layout >, 994
cnd_
trisycl::detail::pooled_shared_work, 1180
trisycl::detail::pooled_work_stealing, 1187
color
opencl_spir.h, 1510
color_mapping
trisycl::vendor::xilinx::graphics::palette, 1168
command_execution_status
trisycl::info, 859
command_group
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >, 910
commit
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 490
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 482
commit_command
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >, 1021
compile_with_kernel_type
trisycl::program, 1204
compiled
trisycl, 833
complete
trisycl::vendor::xilinx::partition, 871
component_name
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, 1197
config.hpp
TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS, 2222, 2225
TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER, 2222, 2225
connect
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 50, 51
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 974, 975
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 272
trisycl::vendor::xilinx::acap::aie::device< Layout >, 998, 999
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 172
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 250
connect_to
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1233
connection
trisycl::vendor::xilinx::acap::aie::connection, 926, 927
const_func
opencl_spir.h, 1473
opencl_spir_req.h, 1675
const_iterator
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 487
trisycl::facade::accessor< AccessorMixIn >, 505
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 479
const_pointer
trisycl::detail::container_element_aspect< T >, 651
trisycl::mixin::accessor< T, Dimensions >, 515
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 479
const_reference
trisycl::buffer< T, Dimensions, Allocator >, 326
trisycl::detail::container_element_aspect< T >, 651
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 467
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 479
const_reverse_iterator
trisycl::facade::accessor< AccessorMixIn >, 505
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 479

constant
Dealing with OpenCL address spaces, 641
constant_buffer
trisycl::access, 844
constant_ptr
Dealing with OpenCL address spaces, 641
constant_space
trisycl::access, 841
context
Platforms, contexts, accessors and queues, 606
trisycl::context, 526–528
trisycl::info, 860
trisycl::vendor::xilinx::latex::context, 930
context::get_info< info::context::devices >
Platforms, contexts, accessors and queues, 614
context::get_info< info::context::platform >
Platforms, contexts, accessors and queues, 614
context::get_info< info::context::reference_count >
Platforms, contexts, accessors and queues, 614
convert
trisycl::detail::vec< DataType, NumElements >, 1309
coord
opencl_spir.h, 1511
copy_back_cl_buffer
trisycl::detail::accessor< T, Dimensions, Mode, Target >, 341
copy_if_modified
trisycl::detail::buffer< T, Dimensions >, 355
copy_in_cl_buffer
trisycl::detail::accessor< T, Dimensions, Mode, Target >, 341
core_trace
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 944–947
correctly_rounded_divide_sqrt
Platforms, contexts, accessors and queues, 611
cosval
opencl_spir.h, 1511
counter_
trisycl::detail::pooled_work_stealing::pool_ctx, 1173
cpu
Platforms, contexts, accessors and queues, 610
cpu_selector
Platforms, contexts, accessors and queues, 604
crbegin
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 311
trisycl::facade::accessor< AccessorMixIn >, 507
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 482
create_in_cache
trisycl::detail::buffer_base, 359
create_parallel_for_arg
Opencl_spir_helpers, 673
create_pool_ctx
trisycl::detail::pooled_shared_work, 1177
trisycl::detail::pooled_work_stealing, 1184
create_sub_devices
trisycl::device, 541
crend
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 311
trisycl::facade::accessor< AccessorMixIn >, 508
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 482
cross
trisycl, 833
cs
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 989
csp
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 968, 969
trisycl::vendor::xilinx::acap::aie::device< Layout >, 994, 995
ctx
trisycl::detail::pooled_shared_work, 1176
trisycl::detail::pooled_work_stealing, 1183
custom
Platforms, contexts, accessors and queues, 610
cv
trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device, 1109
trisycl::vendor::xilinx::graphics::image_grid, 1092
cyclic
trisycl::vendor::xilinx::partition, 871

d
trisycl::detail::opencl_device, 1125
trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >, 391
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >, 915
data
trisycl::mixin::accessor< T, Dimensions >, 517
trisycl::vendor::xilinx::acap::aie::axi_packet, 74
Data access and storage in old version of SYCL, 432
get_pipe_detail, 498
Data access and storage in SYCL, 302
add_buffer_to_task, 375
buffer, 375, 376
buffer_add_to_task, 376
buffer_allocator, 374
image_allocator, 374
map_allocator, 375
waiter, 377
data_host
trisycl::detail::buffer< T, Dimensions >, 355
data_mover
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >, 1026
data_type
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 48

dataflow
trisycl::vendor::xilinx, 865
DBL_DIG
opencl_spir.h, 1473
DBL_EPSILON
opencl_spir.h, 1474
DBL_MANT_DIG
opencl_spir.h, 1474
DBL_MAX
opencl_spir.h, 1474
DBL_MAX_10_EXP
opencl_spir.h, 1475
DBL_MAX_EXP
opencl_spir.h, 1475
DBL_MIN
opencl_spir.h, 1475
DBL_MIN_10_EXP
opencl_spir.h, 1476
DBL_MIN_EXP
opencl_spir.h, 1476
DBL_RADIX
opencl_spir.h, 1476
dd
trisycl::vendor::xilinx::acap::aie::device< Layout >, 995
Dealing with OpenCL address spaces, 621
addr_space, 640
constant, 641
constant_ptr, 641
generic, 641
global, 642
global_ptr, 642
local, 642
local_ptr, 643
make_ptr, 644, 645
multi_ptr, 643
priv, 643
private_ptr, 644
deallocate_buffer
trisycl::detail::buffer< T, Dimensions >, 352
debug.hpp
TRISYCL_DUMP, 2212
TRISYCL_DUMP_T, 2212
TRISYCL_INTERNAL_DUMP, 2212
debug_mode
trisycl::detail::sycl_2_2::pipe< T >, 463
Debugging and tracing support, 665
trace_kernel, 668
Debugging support for ACAP, 297
decrease_clip
trisycl::vendor::xilinx::graphics::palette, 1151, 1152
decrease_frequency
trisycl::vendor::xilinx::graphics::palette, 1152, 1153
decrease_phase
trisycl::vendor::xilinx::graphics::palette, 1153, 1154
default_device
trisycl::device_type_selector, 555
default_handler
trisycl::error_handler, 684
default_selector
Platforms, contexts, accessors and queues, 604
denorm
Platforms, contexts, accessors and queues, 611
detail::accessor
trisycl::detail::buffer< T, Dimensions >, 355
detail_t
trisycl::facade::device< UserFacing, ImplementationDetail >, 551
dev
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 294
trisycl::vendor::xilinx::acap::aie::memory_infrastructure, 133
device
Platforms, contexts, accessors and queues, 607
trisycl::device, 539, 540
trisycl::info, 860
trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >, 388, 389
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >, 382–384
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 971, 972
trisycl::vendor::xilinx::acap::aie::device< Layout >, 997
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 136
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >, 155
Device-side runtime implementation, 675
kernel_IR, 681
serialize_accessor_arg, 678
serialize_arg, 679
set_kernel, 680
device.hpp
SYCL_VENDOR_TRISYCL_DEVICE_SCOPE, 1975, 2005
device_exec_capabilities
Platforms, contexts, accessors and queues, 605
device_fp_config
Platforms, contexts, accessors and queues, 605
device_queue_properties
Platforms, contexts, accessors and queues, 605
device_scope
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 409
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >, 911
device_scope_type
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >, 380, 381
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 404
device_type
Platforms, contexts, accessors and queues, 607,

609
trisycl::device_type_selector, 556
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 405
device_type_selector
trisycl::device_type_selector, 554
device_typename_selector
trisycl::device_typename_selector< DeviceType >, 557
device_with_scope
trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >, 417
devices
Platforms, contexts, accessors and queues, 606
difference_type
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 479
dimension
trisycl::detail::small_array< BasicType, FinalType, Dims >, 657
dimensionality
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 318
discard_read_write
trisycl::access, 842
discard_write
trisycl::access, 842
dispatch_protection
trisycl::vendor::xilinx::graphics::image_grid, 1092
dispatch_set_arg
trisycl::handler, 560
dispatcher
trisycl::vendor::xilinx::graphics::image_grid, 1093
display
trisycl::detail::display_vector< T >, 667
trisycl::item< Dimensions, with_offset >, 744
trisycl::nd_range< Dimensions >, 763
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 52, 53
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 976, 977
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 273, 274
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1000
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 250, 251
trisycl::vendor::xilinx::latex::context, 932
display_border
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 54
display_core_size
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 275
display_routing_configuration
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 55, 56
display_size
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 57
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 276, 277
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 251
dma
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >, 1018, 1019
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >, 80
dma_0
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939–944, 946
dma_1
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939–944, 946
dma_base
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, 1220
trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >, 1242
dma_command
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >, 1017, 1018
dma_dsl
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >, 76, 77
dma_last
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939–944, 946
done
trisycl::vendor::xilinx::graphics::frame_grid, 1059
done_barrier
trisycl::vendor::xilinx::graphics::image_grid, 1093
done_has_been_sampled
trisycl::vendor::xilinx::graphics::image_grid, 1093
done_snapshot
trisycl::vendor::xilinx::graphics::image_grid, 1093
dot
trisycl, 834
double_fp_config
Platforms, contexts, accessors and queues, 608
driver_version
Platforms, contexts, accessors and queues, 608
dst_stride
opencl_spir.h, 1511
dti
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 247

e
trisycl::detail::opencl_event, 1131
east_0
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942–947

trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255, 1257, 1259–1261
east_1
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942–947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255, 1257, 1259–1261
east_2
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942–947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255–1257, 1259, 1261
east_3
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940–947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255–1257, 1259, 1261
east_last
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940–947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255–1257, 1259, 1261
edge1
opencl_spir.h, 1511
ele_in_each_physical_mem
trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim >, 900
element_type
trisycl::detail::small_array< BasicType, FinalType, Dims >, 655
trisycl::mixin::accessor< T, Dimensions >, 515
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >, 421, 422
elems
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >, 430
empty
trisycl::detail::sycl_2_2::pipe< T >, 450
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 469
empty_with_lock
trisycl::detail::sycl_2_2::pipe< T >, 451
enable_profiling
trisycl::property::queue::enable_profiling, 1028
enabled
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input_port, 1101
end
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 312
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 491
trisycl::facade::accessor< AccessorMixIn >, 508
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 482
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >, 427
enum_count
trisycl::detail, 853
enum_distance
trisycl::detail, 853
enum_type
trisycl::views, 872
enum_underlying_type
trisycl::views, 872
epilogues
trisycl::detail::task, 1291
Error handling, 681
async_handler, 708
exception_ptr, 709
error_correction_support
Platforms, contexts, accessors and queues, 608
even
trisycl::vec< DataType, 16 >, 811
trisycl::vec< DataType, 2 >, 790, 791
trisycl::vec< DataType, 3 >, 795
trisycl::vec< DataType, 4 >, 800
trisycl::vec< DataType, 8 >, 806
event
opencl_spir.h, 1511
trisycl::event, 1034, 1035
trisycl::info, 859
event::get_info< info::event::command_execution_status >
trisycl, 834
event::get_info< info::event::reference_count >
trisycl, 834
event::get_profiling_info< info::event_profiling::command_end >
trisycl, 834
event::get_profiling_info< info::event_profiling::command_start >
trisycl, 834
event::get_profiling_info< info::event_profiling::command_submit >
trisycl, 835
event_command_status
trisycl::info, 859
event_list
opencl_spir.h, 1511
event_profiling
trisycl::info, 860
exception
trisycl::async_exception, 690
trisycl::device_error, 699
trisycl::exception, 687
trisycl::runtime_error, 692
exception_ptr
Error handling, 709
exception_ptr_class
trisycl, 831

exec_kernel
Platforms, contexts, accessors and queues, 610
exec_native_kernel
Platforms, contexts, accessors and queues, 610
execution_capabilities
Platforms, contexts, accessors and queues, 608
execution_capability
Platforms, contexts, accessors and queues, 610
execution_ended
trisycl::detail::task, 1291
exp
opencl_spir.h, 1512
expand
Helpers to do array and tuple conversion, 646, 647
Expressing parallelism through kernels, 709
capture_arg_v, 770
id, 770
parallel_for, 771, 773–775
parallel_for_global_offset, 775
parallel_for_iterate, 776
parallel_for_workgroup, 777
parallel_for_workitem, 778
parallel_for_workitem_in_group, 780
range, 781
to_sycl_id, 782
to_tbb_range, 783, 784
extensions
Platforms, contexts, accessors and queues, 608
Extensions for Xilinx ACAP Versal, 296
extents_cast
trisycl::mixin::accessor< T, Dimensions >, 517

facade
trisycl::detail::accessor< T, Dimensions, Mode, Target >, 338
facade_t
trisycl::vendor::xilinx::acap::aie::device< Layout >, 995
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 247, 248
fence_space
trisycl::access, 842
fiber_executor
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 989
fiber_pool
trisycl::detail::fiber_pool, 1040
fifo
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, 1226
fifo_0
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939–944, 946
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1254–1261
fifo_1
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939–942, 944, 946
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1254–1261
fifo_depth
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 949
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1263
fifo_last
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939–942, 944, 946
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1254–1261
file_name
trisycl::vendor::xilinx::latex::context, 934
fill_tuple
Helpers to do array and tuple conversion, 648
final_write_back
trisycl::detail::buffer< T, Dimensions >, 355
finalize
trisycl::vendor::xilinx::latex::context, 932, 933
finalized
trisycl::vendor::xilinx::latex::context, 934
finish_line
trisycl::detail::fiber_pool, 1045
finished
trisycl::detail::queue, 1215
finished_mutex
trisycl::detail::queue, 1215
flag_
trisycl::detail::pooled_shared_work, 1180
trisycl::detail::pooled_work_stealing, 1188
flatten
trisycl::detail::vec< DataType, NumElements >, 1310
flatten_to_tuple
trisycl::detail::vec< DataType, NumElements >, 1311
FLT_DIG
opencl_spir.h, 1476
FLT_EPSILON
opencl_spir.h, 1477
FLT_MANT_DIG
opencl_spir.h, 1477
FLT_MAX
opencl_spir.h, 1477
FLT_MAX_10_EXP
opencl_spir.h, 1477
FLT_MAX_EXP
opencl_spir.h, 1477
FLT_MIN
opencl_spir.h, 1477
FLT_MIN_10_EXP
opencl_spir.h, 1478
FLT_MIN_EXP
opencl_spir.h, 1478
FLT_RADIX

opencl_spir.h, 1478
fma
Platforms, contexts, accessors and queues, 611
fmax
trisycl, 835
fmin
trisycl, 835
for_each_memory_base
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 140
for_each_tile
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 977, 978
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1000, 1001
for_each_tile_base
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 141
for_each_tile_index
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 979
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1001, 1002
for_each_tile_neighborhood
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 980, 981
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1002
for_each_tile_x_index
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 981, 982
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1003
for_each_tile_y_index
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 983
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1004
fp_config
Platforms, contexts, accessors and queues, 610
FP_ILOGB0
opencl_spir.h, 1478
FP_ILOGBNAN
opencl_spir.h, 1478
frame_grid
trisycl::vendor::xilinx::graphics::frame_grid, 1055, 1056
frames
trisycl::vendor::xilinx::graphics::frame_grid, 1059
frequency_log2
trisycl::vendor::xilinx::graphics::palette, 1169
fresh_ctx
trisycl::detail::buffer_base, 366
full
trisycl::detail::sycl_2_2::pipe< T >, 451
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 469
full_with_lock
trisycl::detail::sycl_2_2::pipe< T >, 452
function_class
trisycl, 831
future
trisycl::detail::fiber_pool, 1040
future_work
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 294
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 180
futures
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1239

g
trisycl::vec< DataType, 4 >, 801
Vector types in SYCL, 817
generate_tiles
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 88, 89
generic
Dealing with OpenCL address spaces, 641
generic_space
trisycl::access, 841
geo
trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >, 69
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 969
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 268, 269
trisycl::vendor::xilinx::acap::aie::device< Layout >, 996
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 137
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >, 155
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 164
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 186
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >, 234
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 248
get
std, 824
trisycl::context, 529
trisycl::detail::context, 532
trisycl::detail::device, 548
trisycl::detail::event, 1030
trisycl::detail::host_context, 535
trisycl::detail::host_device, 1069
trisycl::detail::host_event, 1073
trisycl::detail::host_platform, 584
trisycl::detail::host_queue, 1077
trisycl::detail::kernel, 577
trisycl::detail::opencl_context, 1114
trisycl::detail::opencl_device, 1120

trisycl::detail::opencl_event, 1128
trisycl::detail::opencl_kernel, 1133
trisycl::detail::opencl_platform, 589
trisycl::detail::opencl_queue, 1138
trisycl::detail::platform, 592
trisycl::detail::program, 1199
trisycl::detail::queue, 1213
trisycl::detail::small_array< BasicType, FinalType, Dims >, 656
trisycl::device, 541
trisycl::event, 1035
trisycl::kernel, 575
trisycl::platform, 581
trisycl::program, 1204
trisycl::queue, 599
get_access
trisycl::buffer< T, Dimensions, Allocator >, 332
trisycl::sycl_2_2::pipe< T >, 443
trisycl::sycl_2_2::static_pipe< T, Capacity >, 497, 498
get_address
trisycl::detail::address_space_variable< T, AS >, 639
get_align
trisycl::detail::alignment<::trisycl::vec< DataType, NumElements > >, 886
get_as_router_minion
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 57, 58
get_boost_compute
trisycl::context, 529
trisycl::detail::context, 532
trisycl::detail::device, 549
trisycl::detail::event, 1030
trisycl::detail::host_context, 535
trisycl::detail::host_device, 1069
trisycl::detail::host_event, 1074
trisycl::detail::host_platform, 584
trisycl::detail::host_queue, 1077
trisycl::detail::kernel, 577
trisycl::detail::opencl_context, 1114
trisycl::detail::opencl_device, 1120
trisycl::detail::opencl_event, 1128
trisycl::detail::opencl_kernel, 1133
trisycl::detail::opencl_platform, 589
trisycl::detail::opencl_queue, 1138
trisycl::detail::platform, 592
trisycl::detail::queue, 1213
trisycl::device, 542
trisycl::event, 1035
trisycl::platform, 581
trisycl::queue, 599
get_boost_queue
trisycl::context, 529
trisycl::detail::context, 532
trisycl::detail::host_context, 535
trisycl::detail::opencl_context, 1114
get_buffer
trisycl::detail::accessor< T, Dimensions, Mode, Target >, 341
get_cascade_stream_in
trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >, 69, 70
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 188, 189
get_cascade_stream_out
trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >, 70, 71
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 190, 191
get_cl_buffer
trisycl::detail::accessor< T, Dimensions, Mode, Target >, 342
trisycl::detail::accessor_base, 320
trisycl::detail::buffer_base, 360
get_cl_code
trisycl::cl_exception, 689
get_command_execution_status
trisycl::detail::event, 1031
trisycl::detail::host_event, 1074
trisycl::detail::opencl_event, 1128
get_compute_buffer
trisycl::detail::task, 1280
get_context
trisycl::detail::host_queue, 1077
trisycl::detail::opencl_queue, 1139
trisycl::detail::queue, 1213
trisycl::queue, 599
get_count
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 312
trisycl::buffer< T, Dimensions, Allocator >, 333
trisycl::detail::vec< DataType, NumElements >, 1311
trisycl::mixin::accessor< T, Dimensions >, 518
get_destructor_future
trisycl::detail::buffer< T, Dimensions >, 352
get_device
trisycl::detail::host_queue, 1078
trisycl::detail::opencl_queue, 1139
trisycl::detail::queue, 1213
trisycl::queue, 600
trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >, 416
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 409
get_devices
Platforms, contexts, accessors and queues, 614, 615
trisycl::context, 530
trisycl::detail::context, 532
trisycl::detail::host_context, 536
trisycl::detail::opencl_context, 1114
trisycl::detail::platform, 592
trisycl::device, 542
trisycl::platform, 581

get_extension_strings
trisycl::detail::host_platform, 585
trisycl::detail::opencl_platform, 589
trisycl::detail::platform, 593
get_frame
trisycl::vendor::xilinx::graphics::frame_grid, 1057
get_global_id
trisycl::h_item< Dimensions >, 727, 728
trisycl::nd_item< Dimensions >, 755
get_global_linear_id
trisycl::h_item< Dimensions >, 728
trisycl::nd_item< Dimensions >, 755
get_global_range
trisycl::group< Dimensions >, 714
trisycl::h_item< Dimensions >, 729
trisycl::nd_item< Dimensions >, 756
trisycl::nd_range< Dimensions >, 763
get_group
trisycl::h_item< Dimensions >, 730, 731
trisycl::nd_item< Dimensions >, 756
get_group_linear_id
trisycl::h_item< Dimensions >, 731
trisycl::nd_item< Dimensions >, 757
get_group_range
trisycl::group< Dimensions >, 715, 716
trisycl::h_item< Dimensions >, 732, 733
trisycl::nd_item< Dimensions >, 757
trisycl::nd_range< Dimensions >, 764
get_id
trisycl::group< Dimensions >, 716, 717
trisycl::item< Dimensions, with_offset >, 744, 745
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 191, 192
get_info
trisycl::context, 530
trisycl::detail::device, 549
trisycl::detail::host_device, 1070
trisycl::detail::opencl_device, 1120, 1121
trisycl::device, 543
trisycl::event, 1036
trisycl::platform, 582
trisycl::queue, 600
get_info_string
trisycl::detail::host_platform, 585
trisycl::detail::opencl_platform, 589
trisycl::detail::platform, 593
get_item
trisycl::h_item< Dimensions >, 733
trisycl::nd_item< Dimensions >, 758
get_kernel
trisycl::detail::task, 1281
trisycl::program, 1205
get_latest_producer
trisycl::detail::buffer_base, 361
get_linear_id
trisycl::group< Dimensions >, 718
trisycl::item< Dimensions, with_offset >, 746
get_local_id
trisycl::h_item< Dimensions >, 734
trisycl::nd_item< Dimensions >, 758
get_local_linear_id
trisycl::h_item< Dimensions >, 735
trisycl::nd_item< Dimensions >, 758
get_local_range
trisycl::group< Dimensions >, 718, 719
trisycl::h_item< Dimensions >, 736
trisycl::nd_item< Dimensions >, 759
trisycl::nd_range< Dimensions >, 765
get_master_port_dests
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1233, 1234
get_nd_range
trisycl::group< Dimensions >, 720
trisycl::h_item< Dimensions >, 737
trisycl::nd_item< Dimensions >, 759
get_offset
trisycl::group< Dimensions >, 720, 721
trisycl::h_item< Dimensions >, 737
trisycl::item< Dimensions, with_offset >, 747
trisycl::nd_item< Dimensions >, 759
trisycl::nd_range< Dimensions >, 765
get_or_register
trisycl::detail::cache< Key, Value >, 903
get_order
trisycl::detail::accessor_base, 320
get_palette
trisycl::vendor::xilinx::graphics::image_grid, 1084
get_partition_type
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >, 427, 428
get_pipe_detail
Data access and storage in old version of SYCL, 498
trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >, 436
trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >, 434
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 470
get_platform
trisycl::context, 530
trisycl::detail::context, 533
trisycl::detail::device, 549
trisycl::detail::host_context, 536
trisycl::detail::host_device, 1070
trisycl::detail::opencl_context, 1115
trisycl::detail::opencl_device, 1122
trisycl::device, 543
trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >, 389
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >, 384
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 410
get_platforms

trisycl::platform, 582
get_pointer
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 312
trisycl::facade::accessor< AccessorMixIn >, 509
get_profiling_info
trisycl::detail::event, 1031
trisycl::detail::host_event, 1074
trisycl::detail::opencl_event, 1129
trisycl::event, 1036
get_property
trisycl::property_list, 1209
trisycl::queue, 600
get_queue
trisycl::detail::task, 1281
get_range
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 312
trisycl::buffer< T, Dimensions, Allocator >, 333
trisycl::item< Dimensions, with_offset >, 747, 748
trisycl::mixin::accessor< T, Dimensions >, 519
get_reference_count
trisycl::detail::context, 533
trisycl::detail::event, 1031
trisycl::detail::host_context, 536
trisycl::detail::host_event, 1074
trisycl::detail::opencl_context, 1115
trisycl::detail::opencl_event, 1129
get_size
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 313
trisycl::buffer< T, Dimensions, Allocator >, 333
trisycl::detail::vec< DataType, NumElements >, 1312
trisycl::mixin::accessor< T, Dimensions >, 519
get_storage
trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >, 390
trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >, 400
trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >, 416
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >, 385
trisycl::vendor::trisycl::scope::platform< PlatformStorage >, 396
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 410
get_underlying_device
trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >, 390
get_underlying_platform
trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >, 400
get_underlying_queue
trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >, 417
get_wait_list
trisycl::event, 1036
global
Dealing with OpenCL address spaces, 642
Platforms, contexts, accessors and queues, 612
global_and_local
trisycl::access, 842
global_buffer
trisycl::access, 844
global_config.hpp
TRISYCL_SYCL_NAMESPACE, 1424
TRISYCL_WEAK_ATTRIB_PREFIX, 1425
TRISYCL_WEAK_ATTRIB_SUFFIX, 1425
global_index
trisycl::h_item< Dimensions >, 739
trisycl::item< Dimensions, with_offset >, 751
trisycl::nd_item< Dimensions >, 760
global_mem_cache_line_size
Platforms, contexts, accessors and queues, 608
global_mem_cache_size
Platforms, contexts, accessors and queues, 608
global_mem_cache_type
Platforms, contexts, accessors and queues, 608, 611
global_mem_size
Platforms, contexts, accessors and queues, 608
global_ptr
Dealing with OpenCL address spaces, 642
global_range
trisycl::item< Dimensions, with_offset >, 751
trisycl::nd_range< Dimensions >, 766
global_space
trisycl::access, 841, 842
gpu
Platforms, contexts, accessors and queues, 610
gpu_selector
Platforms, contexts, accessors and queues, 605
Graphics support for CGRA-like interaction, 300
gray
trisycl::vendor::xilinx::graphics::palette, 1149
gray_palette
trisycl::vendor::xilinx::graphics::palette, 1155, 1156
grid
trisycl::vendor::xilinx::graphics::frame_grid, 1059
group
trisycl::group< Dimensions >, 713
group_id
trisycl::group< Dimensions >, 723

h_item
trisycl::h_item< Dimensions >, 726
half
trisycl::half, 1061
half_fp_config
Platforms, contexts, accessors and queues, 608
handle_key_press_event
trisycl::vendor::xilinx::graphics::palette, 1156, 1157
handler
trisycl::detail::accessor< T, Dimensions, Mode, Target >, 345

trisycl::handler, 560
trisycl::kernel, 575
trisycl::program, 1205
has_extension
trisycl::detail::device, 549
trisycl::detail::host_device, 1070
trisycl::detail::host_platform, 585
trisycl::detail::opencl_device, 1123
trisycl::detail::opencl_platform, 590
trisycl::detail::platform, 593
trisycl::device, 544
trisycl::platform, 582
has_kernel
trisycl::program, 1205
has_property
trisycl::property_list, 1209
trisycl::queue, 600
has_ready_fibers
trisycl::detail::pooled_shared_work, 1178
trisycl::detail::pooled_work_stealing, 1185
has_some_storage_p
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >, 386
trisycl::vendor::trisycl::scope::platform< PlatformStorage >, 397
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 413
hash
trisycl::detail::shared_ptr_implementation< Parent, Implementation >, 1247
hash_class
trisycl, 831
hb
trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >, 322
Helpers to do array and tuple conversion, 645
expand, 646, 647
fill_tuple, 648
tuple_to_array, 649
tuple_to_array_iterate, 649
hi
trisycl::vec< DataType, 16 >, 811
trisycl::vec< DataType, 2 >, 791
trisycl::vec< DataType, 3 >, 795
trisycl::vec< DataType, 4 >, 801
trisycl::vec< DataType, 8 >, 807
horizontal_barrier
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 192, 194
host
Platforms, contexts, accessors and queues, 610
host_accessor
trisycl::host_accessor< DataType, Dimensions, AccessMode >, 439
host_buffer
trisycl::access, 844
host_context
trisycl::detail::buffer< T, Dimensions >, 356
host_device.hpp
TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE, 1693
host_image
trisycl::access, 844
host_selector
Platforms, contexts, accessors and queues, 606
host_unified_memory
Platforms, contexts, accessors and queues, 608
HUGE_VAL
opencl_spir.h, 1478
HUGE_VALF
opencl_spir.h, 1479

i
opencl_spir.h, 1512
id
Expressing parallelism through kernels, 770
trisycl::id< Dimensions >, 741
id_
trisycl::detail::pooled_work_stealing, 1188
image
trisycl::access, 844
image2d_max_height
Platforms, contexts, accessors and queues, 607
image2d_max_width
Platforms, contexts, accessors and queues, 607
image3d_max_depth
Platforms, contexts, accessors and queues, 608
image3d_max_height
Platforms, contexts, accessors and queues, 607
image3d_max_width
Platforms, contexts, accessors and queues, 607
image_allocator
Data access and storage in SYCL, 374
image_array
trisycl::access, 844
image_grid
trisycl::vendor::xilinx::graphics::application, 889
trisycl::vendor::xilinx::graphics::image_grid, 1082, 1083
image_max_array_size
Platforms, contexts, accessors and queues, 608
image_max_buffer_size
Platforms, contexts, accessors and queues, 608
image_support
Platforms, contexts, accessors and queues, 607
image_x
trisycl::vendor::xilinx::graphics::image_grid, 1094
image_y
trisycl::vendor::xilinx::graphics::image_grid, 1094
images
trisycl::vendor::xilinx::graphics::image_grid, 1094
implementation
trisycl::detail::shared_ptr_implementation< Parent, Implementation >, 1249
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 475

trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 484
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >, 386
trisycl::vendor::trisycl::scope::platform< PlatformStorage >, 397
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 413
implementation_t
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 308, 318
trisycl::buffer< T, Dimensions, Allocator >, 326, 336
trisycl::context, 526
trisycl::detail::buffer_waiter< T, Dimensions, Allocator >, 369, 370
trisycl::detail::sycl_2_2::pipe< T >, 449
trisycl::device, 539
trisycl::event, 1034, 1038
trisycl::facade::device< UserFacing, ImplementationDetail >, 551
trisycl::kernel, 574, 575
trisycl::program, 1203, 1206
trisycl::queue, 596, 604
trisycl::sycl_2_2::pipe< T >, 442, 444
trisycl::sycl_2_2::static_pipe< T, Capacity >, 496, 498
in
trisycl::vendor::xilinx::acap::aie::connection, 927
trisycl::vendor::xilinx::acap::aie::connection::input, 1099
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 277, 278
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >, 235, 236
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 252
in_connection
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 58, 59
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 278, 279
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >, 236, 237
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 253
include/CL/sycl.hpp, 1325, 1326
include/sycl/sycl.hpp, 1326, 1327
include/sycl/vendor/triSYCL/pipe/cout.hpp, 1876
include/sycl/vendor/triSYCL/random/xorshift.hpp, 1878, 1879
include/sycl/vendor/triSYCL/scope.hpp, 1883, 1884
include/sycl/vendor/triSYCL/scope/detail/util.hpp, 1886, 1887
include/sycl/vendor/triSYCL/scope/device.hpp, 1975, 1976
include/sycl/vendor/triSYCL/scope/device/detail/device.hpp, 1978
include/sycl/vendor/triSYCL/scope/platform.hpp, 1889, 1890
include/sycl/vendor/triSYCL/scope/platform/detail/platform.hpp, 1892
include/sycl/vendor/triSYCL/scope/queue.hpp, 2126, 2128
include/sycl/vendor/triSYCL/scope/queue/detail/queue.hpp, 2130, 2131
include/sycl/vendor/Xilinx/acap.hpp, 1907, 1908
include/sycl/vendor/Xilinx/acap/aie.hpp, 1910, 1911
include/sycl/vendor/Xilinx/acap/aie/accessor.hpp, 1913, 1914
include/sycl/vendor/Xilinx/acap/aie/axi_stream_switch.hpp, 1935, 1937
include/sycl/vendor/Xilinx/acap/aie/cascade_stream.hpp, 1955, 1957
include/sycl/vendor/Xilinx/acap/aie/connection.hpp, 1961, 1963
include/sycl/vendor/Xilinx/acap/aie/device.hpp, 1979, 1981
include/sycl/vendor/Xilinx/acap/aie/device/detail/device.hpp, 1984, 1986
include/sycl/vendor/Xilinx/acap/aie/dma.hpp, 2020, 2023
include/sycl/vendor/Xilinx/acap/aie/geography.hpp, 2033, 2035
include/sycl/vendor/Xilinx/acap/aie/layout.hpp, 2057, 2058
include/sycl/vendor/Xilinx/acap/aie/layout/full.hpp, 2061, 2062
include/sycl/vendor/Xilinx/acap/aie/layout/one_pe.hpp, 2065, 2067
include/sycl/vendor/Xilinx/acap/aie/layout/size.hpp, 2070, 2072
include/sycl/vendor/Xilinx/acap/aie/layout/small.hpp, 2075, 2077
include/sycl/vendor/Xilinx/acap/aie/layout/versal.hpp, 2080, 2082
include/sycl/vendor/Xilinx/acap/aie/lock.hpp, 2086, 2088
include/sycl/vendor/Xilinx/acap/aie/memory.hpp, 2093, 2096
include/sycl/vendor/Xilinx/acap/aie/memory_base.hpp, 2100, 2102
include/sycl/vendor/Xilinx/acap/aie/memory_infrastructure.hpp, 2106, 2108
include/sycl/vendor/Xilinx/acap/aie/program.hpp, 2112, 2114
include/sycl/vendor/Xilinx/acap/aie/queue.hpp, 2132, 2134
include/sycl/vendor/Xilinx/acap/aie/shim_tile.hpp, 2157, 2158
include/sycl/vendor/Xilinx/acap/aie/tile.hpp, 2164, 2166
include/sycl/vendor/Xilinx/acap/aie/tile_base.hpp, 2178, 2180
include/sycl/vendor/Xilinx/acap/aie/tile_infrastructure.hpp, 2186, 2188
include/sycl/vendor/Xilinx/acap/aie/tile_infrastructure/detail/tile_infrastructu 2191, 2193

include/sycl/vendor/Xilinx/acap/debug.hpp, 2208, 2209
include/sycl/vendor/Xilinx/acap/tests/processor_array.cpp, 2218
include/sycl/vendor/Xilinx/config.hpp, 2222, 2223
include/sycl/vendor/Xilinx/fpga.hpp, 2226
include/sycl/vendor/Xilinx/fpga/opt_decorate_func.hpp, 2228
include/sycl/vendor/Xilinx/fpga/partition_array.hpp, 2231, 2232
include/sycl/vendor/Xilinx/fpga/ssdm_inst.hpp, 2239
include/sycl/vendor/Xilinx/graphics.hpp, 2241, 2242
include/sycl/vendor/Xilinx/latex.hpp, 2260, 2261
include/triSYCL/access.hpp, 1331
include/triSYCL/accessor.hpp, 1915, 1916
include/triSYCL/accessor/detail/accessor_base.hpp, 1332, 1334
include/triSYCL/accessor/detail/local_accessor.hpp, 1335, 1336
include/triSYCL/accessor/facade/accessor.hpp, 1922, 1923
include/triSYCL/accessor/mixin/accessor.hpp, 1925, 1927
include/triSYCL/address_space.hpp, 1337, 1339
include/triSYCL/address_space/detail/address_space.hpp, 1341, 1344
include/triSYCL/allocator.hpp, 1349, 1350
include/triSYCL/buffer.hpp, 1351, 1352
include/triSYCL/buffer/detail/accessor.hpp, 1929, 1930
include/triSYCL/buffer/detail/buffer.hpp, 1359, 1361
include/triSYCL/buffer/detail/buffer_base.hpp, 1365, 1366
include/triSYCL/buffer/detail/buffer_waiter.hpp, 1370, 1372
include/triSYCL/buffer_allocator.hpp, 1373, 1374
include/triSYCL/command_group/detail/task.hpp, 1375, 1376
include/triSYCL/context.hpp, 1380, 1382
include/triSYCL/context/detail/context.hpp, 1385, 1386
include/triSYCL/context/detail/context_tail.hpp, 1389
include/triSYCL/context/detail/host_context.hpp, 1390, 1391
include/triSYCL/context/detail/opencl_context.hpp, 1392, 1394
include/triSYCL/detail/alignment_helper.hpp, 1395, 1397
include/triSYCL/detail/all_true.hpp, 1398
include/triSYCL/detail/array_tuple_helpers.hpp, 1399, 1400
include/triSYCL/detail/cache.hpp, 1402, 1403
include/triSYCL/detail/container_element_aspect.hpp, 1405
include/triSYCL/detail/debug.hpp, 2210, 2213
include/triSYCL/detail/default_classes.hpp, 1406, 1407
include/triSYCL/detail/enum.hpp, 1409, 1411
include/triSYCL/detail/fiber/pooled_shared_work.hpp, 1411, 1413
include/triSYCL/detail/fiber/pooled_work_stealing.hpp, 1415, 1416
include/triSYCL/detail/fiber_pool.hpp, 1419, 1421
include/triSYCL/detail/global_config.hpp, 1423, 1425
include/triSYCL/detail/instantiate_kernel.hpp, 1427, 1428
include/triSYCL/detail/linear_id.hpp, 1430, 1431
include/triSYCL/detail/metaprogramming.hpp, 1431, 1433
include/triSYCL/detail/overloaded.hpp, 1433, 1434
include/triSYCL/detail/property.hpp, 1434, 1435
include/triSYCL/detail/ranges.hpp, 1435, 1437
include/triSYCL/detail/shared_ptr_implementation.hpp, 1438, 1439
include/triSYCL/detail/singleton.hpp, 1440, 1441
include/triSYCL/detail/small_array.hpp, 1442, 1444
include/triSYCL/detail/SPIR/opencl_spir.h, 1449, 1516
include/triSYCL/detail/SPIR/opencl_spir_helpers.hpp, 1670, 1672
include/triSYCL/detail/SPIR/opencl_spir_req.h, 1673, 1687
include/triSYCL/detail/unimplemented.hpp, 1689, 1690
include/triSYCL/device.hpp, 1989, 1990
include/triSYCL/device/detail/device.hpp, 1994, 1995
include/triSYCL/device/detail/device_tail.hpp, 1691
include/triSYCL/device/detail/host_device.hpp, 1692, 1694
include/triSYCL/device/detail/opencl_device.hpp, 1696, 1697
include/triSYCL/device/detail/opencl_device_tail.hpp, 1699, 1700
include/triSYCL/device/facade/device.hpp, 1997, 1998
include/triSYCL/device_runtime.hpp, 1702, 1703
include/triSYCL/device_selector.hpp, 1706, 1707
include/triSYCL/device_selector/detail/device_selector_tail.hpp, 1708, 1709
include/triSYCL/error_handler.hpp, 1711, 1712
include/triSYCL/event.hpp, 1713, 1715
include/triSYCL/event/detail/event.hpp, 1717
include/triSYCL/event/detail/host_event.hpp, 1720, 1721
include/triSYCL/event/detail/opencl_event.hpp, 1722
include/triSYCL/exception.hpp, 1724, 1726
include/triSYCL/group.hpp, 1728, 1729
include/triSYCL/h_item.hpp, 1732, 1733
include/triSYCL/half.hpp, 1736, 1737
include/triSYCL/handler.hpp, 1737, 1738
include/triSYCL/id.hpp, 1745, 1746
include/triSYCL/image.hpp, 1747, 1748
include/triSYCL/info/context.hpp, 1387, 1388
include/triSYCL/info/device.hpp, 1998, 2001
include/triSYCL/info/event.hpp, 1718, 1719
include/triSYCL/info/param_traits.hpp, 1748, 1750
include/triSYCL/info/platform.hpp, 1893, 1895
include/triSYCL/info/queue.hpp, 2135, 2136
include/triSYCL/item.hpp, 1750, 1751
include/triSYCL/kernel.hpp, 1753, 1754
include/triSYCL/kernel/detail/kernel.hpp, 1756, 1758
include/triSYCL/kernel/detail/opencl_kernel.hpp, 1759, 1761

include/triSYCL/math.hpp, 1763, 1767
include/triSYCL/nd_item.hpp, 1772, 1773
include/triSYCL/nd_range.hpp, 1776, 1777
include/triSYCL/opencl_types.hpp, 1778, 1784
include/triSYCL/parallelism.hpp, 1786, 1787
include/triSYCL/parallelism/detail/parallelism.hpp, 1788, 1790
include/triSYCL/parallelism/detail/parallelism_tbb.hpp, 1796, 1797
include/triSYCL/pipe.hpp, 1824, 1825
include/triSYCL/platform.hpp, 1896, 1898
include/triSYCL/platform/detail/host_platform.hpp, 1800, 1801
include/triSYCL/platform/detail/host_platform_tail.hpp, 1803
include/triSYCL/platform/detail/opencl_platform.hpp, 1804, 1805
include/triSYCL/platform/detail/opencl_platform_tail.hpp, 1808
include/triSYCL/platform/detail/platform.hpp, 1900, 1901
include/triSYCL/platform/detail/platform_tail.hpp, 1809, 1810
include/triSYCL/program.hpp, 2117, 2118
include/triSYCL/program/detail/program.hpp, 2120, 2121
include/triSYCL/property/queue.hpp, 2137, 2138
include/triSYCL/property_list.hpp, 1810, 1812
include/triSYCL/queue.hpp, 2138, 2139
include/triSYCL/queue/detail/host_queue.hpp, 1813, 1814
include/triSYCL/queue/detail/opencl_queue.hpp, 1815, 1816
include/triSYCL/queue/detail/queue.hpp, 2144, 2145
include/triSYCL/queue/detail/queue_tail.hpp, 1818, 1819
include/triSYCL/range.hpp, 1819, 1821
include/triSYCL/rounding_mode.hpp, 1822, 1823
include/triSYCL/sycl.hpp, 1328, 1329
include/triSYCL/sycl_2_2/pipe.hpp, 1826, 1827
include/triSYCL/sycl_2_2/pipe/detail/pipe.hpp, 1829, 1830
include/triSYCL/sycl_2_2/pipe/detail/pipe_accessor.hpp, 1836, 1837
include/triSYCL/sycl_2_2/pipe_reservation.hpp, 1841, 1842
include/triSYCL/sycl_2_2/pipe_reservation/detail/pipe_reservation.hpp, 1844, 1846
include/triSYCL/sycl_2_2/static_pipe.hpp, 1848, 1849
include/triSYCL/vec.hpp, 1851, 1853
include/triSYCL/vec/detail/swiz2.hpp, 1866
include/triSYCL/vec/detail/swiz3.hpp, 1867
include/triSYCL/vec/detail/swiz4.hpp, 1868
include/triSYCL/vec/detail/swiz_rgba.hpp, 1872
include/triSYCL/vec/detail/vec.hpp, 1860, 1862
include/triSYCL/vendor/triSYCL/pipe/cout.hpp, 1877, 1878
include/triSYCL/vendor/triSYCL/random/xorshift.hpp, 1881, 1882
include/triSYCL/vendor/triSYCL/scope.hpp, 1885, 1886
include/triSYCL/vendor/triSYCL/scope/detail/util.hpp, 1888, 1889
include/triSYCL/vendor/triSYCL/scope/device.hpp, 2004, 2006
include/triSYCL/vendor/triSYCL/scope/device/detail/device.hpp, 2008, 2009
include/triSYCL/vendor/triSYCL/scope/platform.hpp, 1903, 1904
include/triSYCL/vendor/triSYCL/scope/platform/detail/platform.hpp, 1906, 1907
include/triSYCL/vendor/triSYCL/scope/queue.hpp, 2147, 2149
include/triSYCL/vendor/triSYCL/scope/queue/detail/queue.hpp, 2151, 2152
include/triSYCL/vendor/Xilinx/acap.hpp, 1908, 1909
include/triSYCL/vendor/Xilinx/acap/aie.hpp, 1911, 1912
include/triSYCL/vendor/Xilinx/acap/aie/accessor.hpp, 1933, 1935
include/triSYCL/vendor/Xilinx/acap/aie/axi_stream_switch.hpp, 1945, 1947
include/triSYCL/vendor/Xilinx/acap/aie/cascade_stream.hpp, 1958, 1960
include/triSYCL/vendor/Xilinx/acap/aie/connection.hpp, 1968, 1970
include/triSYCL/vendor/Xilinx/acap/aie/device.hpp, 2010, 2012
include/triSYCL/vendor/Xilinx/acap/aie/device/detail/device.hpp, 2015, 2017
include/triSYCL/vendor/Xilinx/acap/aie/dma.hpp, 2027, 2029
include/triSYCL/vendor/Xilinx/acap/aie/geography.hpp, 2045, 2047
include/triSYCL/vendor/Xilinx/acap/aie/layout.hpp, 2059, 2060
include/triSYCL/vendor/Xilinx/acap/aie/layout/full.hpp, 2063, 2065
include/triSYCL/vendor/Xilinx/acap/aie/layout/one_pe.hpp, 2068, 2070
include/triSYCL/vendor/Xilinx/acap/aie/layout/size.hpp, 2073, 2075
include/triSYCL/vendor/Xilinx/acap/aie/layout/small.hpp, 2078, 2080
include/triSYCL/vendor/Xilinx/acap/aie/layout/versal.hpp, 2083, 2085
include/triSYCL/vendor/Xilinx/acap/aie/lock.hpp, 2089, 2092
include/triSYCL/vendor/Xilinx/acap/aie/memory.hpp, 2097, 2099
include/triSYCL/vendor/Xilinx/acap/aie/memory_base.hpp, 2103, 2105
include/triSYCL/vendor/Xilinx/acap/aie/memory_infrastructure.hpp, 2109, 2111
include/triSYCL/vendor/Xilinx/acap/aie/program.hpp, 2121, 2123
include/triSYCL/vendor/Xilinx/acap/aie/queue.hpp, 2153, 2155

include/triSYCL/vendor/Xilinx/acap/aie/shim_tile.hpp, 2160, 2162
include/triSYCL/vendor/Xilinx/acap/aie/tile.hpp, 2171, 2173
include/triSYCL/vendor/Xilinx/acap/aie/tile_base.hpp, 2181, 2184
include/triSYCL/vendor/Xilinx/acap/aie/tile_infrastructure.hpp, 2197, 2199
include/triSYCL/vendor/Xilinx/acap/aie/tile_infrastructure/detail/tile_infrastructure.hpp, 2202, 2204
include/triSYCL/vendor/Xilinx/acap/debug.hpp, 2215, 2217
include/triSYCL/vendor/Xilinx/acap/tests/processor_array.cpp, 2220
include/triSYCL/vendor/Xilinx/config.hpp, 2223, 2225
include/triSYCL/vendor/Xilinx/fpga.hpp, 2226, 2227
include/triSYCL/vendor/Xilinx/fpga/opt_decorate_func.hpp, 2229, 2230
include/triSYCL/vendor/Xilinx/fpga/partition_array.hpp, 2235, 2236
include/triSYCL/vendor/Xilinx/fpga/ssdm_inst.hpp, 2240, 2241
include/triSYCL/vendor/Xilinx/graphics.hpp, 2251, 2252
include/triSYCL/vendor/Xilinx/latex.hpp, 2263, 2264
increase_clip
    trisycl::vendor::xilinx::graphics::palette, 1159
increase_frequency
    trisycl::vendor::xilinx::graphics::palette, 1160, 1161
increase_phase
    trisycl::vendor::xilinx::graphics::palette, 1161, 1162
indices
    trisycl::mixin::accessor< T, Dimensions >::track_index< N >, 1297
inf_nan
    Platforms, contexts, accessors and queues, 611
INFINITY
    opencl_spir.h, 1479
initial_state
    trisycl::vendor::trisycl::random::xorshift< Nbits >, 1322
initialized
    trisycl::vendor::xilinx::graphics::application, 894
input
    trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 59, 60
    trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 280, 281
    trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 173, 174
    trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 254
input_ports
    trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 65
input_shared_pointer
    trisycl::detail::buffer< T, Dimensions >, 356
instance
    trisycl::detail::opencl_context, 1115
    trisycl::detail::opencl_device, 1123
    trisycl::detail::opencl_event, 1129
    trisycl::detail::opencl_kernel, 1134
    trisycl::detail::opencl_platform, 590
    trisycl::detail::opencl_queue, 1139, 1140
    trisycl::detail::singleton< T >, 1273
instantiate_kernel
    trisycl::detail, 850
INT_MAX
    opencl_spir.h, 1479
INT_MIN
    opencl_spir.h, 1479
interconnect
    trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 949
intptr_t
    opencl_spir.h, 1497
iptr
    opencl_spir.h, 1512
is_accelerator
    trisycl::detail::device, 549
    trisycl::detail::host_device, 1071
    trisycl::detail::opencl_device, 1123
    trisycl::device, 544
is_available
    Platforms, contexts, accessors and queues, 608
is_axi_master
    trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 947, 948
    trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1262
is_cached
    trisycl::buffer< T, Dimensions, Allocator >, 333
    trisycl::detail::buffer_base, 361
is_cascade_end
    trisycl::vendor::xilinx::acap::aie::geography< Layout >, 89, 90
    trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 195
is_cascade_start
    trisycl::vendor::xilinx::acap::aie::geography< Layout >, 90
    trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 196
is_compiler_available
    Platforms, contexts, accessors and queues, 608
is_cpu
    trisycl::detail::device, 549
    trisycl::detail::host_device, 1071
    trisycl::detail::opencl_device, 1124
    trisycl::device, 544
is_data_up_to_date
    trisycl::buffer< T, Dimensions, Allocator >, 334
    trisycl::detail::buffer_base, 361
is_done
    trisycl::vendor::xilinx::graphics::application, 889, 890
is_done_barrier

trisycl::vendor::xilinx::graphics::application, 890
trisycl::vendor::xilinx::graphics::image_grid, 1085
is_east_column
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 91
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 196, 197
is_east_valid
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 197, 198
is_endian_little
Platforms, contexts, accessors and queues, 608
is_gpu
trisycl::detail::device, 550
trisycl::detail::host_device, 1071
trisycl::detail::opencl_device, 1124
trisycl::device, 545
is_host
trisycl::context, 530
trisycl::detail::context, 533
trisycl::detail::device, 550
trisycl::detail::event, 1031
trisycl::detail::host_context, 537
trisycl::detail::host_device, 1072
trisycl::detail::host_event, 1075
trisycl::detail::host_platform, 586
trisycl::detail::host_queue, 1078
trisycl::detail::opencl_context, 1116
trisycl::detail::opencl_device, 1124
trisycl::detail::opencl_event, 1130
trisycl::detail::opencl_platform, 590
trisycl::detail::opencl_queue, 1140
trisycl::detail::platform, 593
trisycl::detail::queue, 1213
trisycl::device, 545
trisycl::event, 1036
trisycl::platform, 583
trisycl::queue, 600
is_linker_available
Platforms, contexts, accessors and queues, 608
is_memory_module
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 92, 93
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 198, 199
is_memory_module_east
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 199, 200
is_memory_module_north
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 200, 201
is_memory_module_south
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 202
is_memory_module_west
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 203, 204
is_noc_tile
trisycl::vendor::xilinx::acap::aie::layout::full, 117
is_north_east_valid
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 204, 205
is_north_row
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 94
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 205, 206
is_north_valid
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 206
is_north_west_valid
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 207
is_pl_tile
trisycl::vendor::xilinx::acap::aie::layout::full, 117
is_range_v
trisycl::detail, 853
is_read_access
trisycl::detail::accessor< T, Dimensions, Mode, Target >, 342
is_read_only
trisycl::buffer< T, Dimensions, Allocator >, 334
is_south_east_valid
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 208, 209
is_south_row
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 94, 95
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 209, 210
is_south_valid
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 210, 211
is_south_west_valid
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 211, 212
is_valid_memory_module_offset
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 95, 96
is_west_column
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 96, 97
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 212, 213
is_west_valid
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 213, 214
is_write_access
trisycl::detail::accessor< T, Dimensions, Mode, Target >, 342
is_x_valid
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 97, 98
is_x_y_valid
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 98, 99

is_y_valid
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 100, 101
item
trisycl::item< Dimensions, with_offset >, 744
iterator
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 487
trisycl::facade::accessor< AccessorMixIn >, 505
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 479

join
trisycl::detail::fiber_pool, 1042
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1234
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >, 1022
joinable
trisycl::detail::fiber_pool, 1046

k
trisycl::detail::opencl_kernel, 1135
trisycl::vendor::xilinx::graphics::palette, 1169
kernel
trisycl::detail::task, 1291
trisycl::kernel, 574
kernel.hpp
TRISYCL_ParallelForKernel_RANGE, 1757
kernel_end
trisycl::detail::queue, 1214
kernel_exec
opencl_spir.h, 1479
kernel_IR
Device-side runtime implementation, 681
kernel_outliner
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 142
kernel_start
trisycl::detail::queue, 1214
key_type
trisycl::detail::cache< Key, Value >, 902
kind
trisycl::vendor::xilinx::graphics::palette, 1149

L1_cache
Platforms, contexts, accessors and queues, 612
L2_cache
Platforms, contexts, accessors and queues, 612
L3_cache
Platforms, contexts, accessors and queues, 612
L4_cache
Platforms, contexts, accessors and queues, 612
latency
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 949
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1263
latest_producer
trisycl::detail::buffer_base, 366
latest_producer_mutex
trisycl::detail::buffer_base, 367
LaTeX support for graphics output, 301
launch
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >, 1023
launch_device_kernel
trisycl::detail, 850
Layout, 1105
layout
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 84
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >, 155, 156
lck
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >, 80
length
trisycl, 836
linear_id
Some helpers for the implementation, 663
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 101, 102
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 214, 215
linear_x
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 102, 103
linear_y
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 104, 105
linked
trisycl, 833
lo
opencl_spir.h, 1512
trisycl::vec< DataType, 16 >, 811, 812
trisycl::vec< DataType, 2 >, 791
trisycl::vec< DataType, 3 >, 795
trisycl::vec< DataType, 4 >, 801, 802
trisycl::vec< DataType, 8 >, 807
local
Dealing with OpenCL address spaces, 642
Platforms, contexts, accessors and queues, 612
trisycl::access, 844
local_index
trisycl::h_item< Dimensions >, 739
trisycl::nd_item< Dimensions >, 760
local_mem_size
Platforms, contexts, accessors and queues, 608
local_mem_type
Platforms, contexts, accessors and queues, 608, 611
local_ptr
Dealing with OpenCL address spaces, 643
local_range
trisycl::nd_range< Dimensions >, 766
local_space

trisycl::access, 841, 842
lock
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >, 77, 78
trisycl::vendor::xilinx::acap::aie::lock_unit, 125, 126
trisycl::vendor::xilinx::acap::aie::memory_base, 130
trisycl::vendor::xilinx::acap::aie::memory_infrastructure, 132, 133
lock_id_t
trisycl::vendor::xilinx::acap::aie::lock_unit, 125
lock_number
trisycl::vendor::xilinx::acap::aie::lock_unit, 126
locks
trisycl::vendor::xilinx::acap::aie::lock_unit, 126
LONG_MAX
opencl_spir.h, 1480
LONG_MIN
opencl_spir.h, 1480
lqueue_
trisycl::detail::pooled_shared_work, 1180
lqueue_type
trisycl::detail::pooled_shared_work, 1176

m
trisycl::detail::cache< Key, Value >, 905
trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device, 1109
M_1_PI
opencl_spir.h, 1480
M_1_PI_F
opencl_spir.h, 1480
M_2_PI
opencl_spir.h, 1480
M_2_PI_F
opencl_spir.h, 1481
M_2_SQRTPI
opencl_spir.h, 1481
M_2_SQRTPI_F
opencl_spir.h, 1481
m_dma_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 949
m_dma_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 950
M_E
opencl_spir.h, 1481
M_E_F
opencl_spir.h, 1481
m_east_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 950
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1263
m_east_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 950
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1264
m_fifo_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 951
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1264
m_fifo_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 951
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1264
M_LN10
opencl_spir.h, 1481
M_LN10_F
opencl_spir.h, 1482
M_LN2
opencl_spir.h, 1482
M_LN2_F
opencl_spir.h, 1482
M_LOG10E
opencl_spir.h, 1482
M_LOG10E_F
opencl_spir.h, 1482
M_LOG2E
opencl_spir.h, 1482
M_LOG2E_F
opencl_spir.h, 1483
m_me_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 951
m_me_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 952
m_north_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 952
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1265
m_north_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 952
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1265
M_PI
opencl_spir.h, 1483
M_PI_2
opencl_spir.h, 1483
M_PI_2_F
opencl_spir.h, 1483
M_PI_4
opencl_spir.h, 1483
M_PI_4_F
opencl_spir.h, 1483
M_PI_F
opencl_spir.h, 1484
m_south_range

trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 953
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1265
m_south_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 953
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1266
M_SQRT1_2
opencl_spir.h, 1484
M_SQRT1_2_F
opencl_spir.h, 1484
M_SQRT2
opencl_spir.h, 1484
M_SQRT2_F
opencl_spir.h, 1484
m_tile_ctrl_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 953
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1266
m_tile_ctrl_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 954
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1266
m_west_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 954
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1267
m_west_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 954
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1267
main
processor_array.cpp, 2218, 2220
make_array_with_func
Opencl_spir_helpers, 673
make_ptr
Dealing with OpenCL address spaces, 644, 645
make_spir_h_item
Opencl_spir_helpers, 674
make_spir_nd_range
Opencl_spir_helpers, 674
Manage default configuration and types, 669
__SYCL_SINGLE_SOURCE__, 670
BOOST_COMPUTE_CL_VERSION_1_2, 670
CL_SYCL_LANGUAGE_VERSION, 670
TRISYCL_CL_LANGUAGE_VERSION, 670
TRISYCL_DEBUG_ONLY, 671
TRISYCL_DEVICE_ONLY, 671
TRISYCL_MAKE_BOOST_CIRCULARBUFFER_THREADSAFE, 671
TRISYCL_OPENCL_ONLY, 671
map
trisycl::detail::vec< DataType, NumElements >, 1312
map_allocator
Data access and storage in SYCL, 375
mark_as_written
trisycl::buffer< T, Dimensions, Allocator >, 334
trisycl::detail::buffer< T, Dimensions >, 353
mask
opencl_spir.h, 1512
master_port_layout
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939, 941
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1254, 1256
math.hpp
TRISYCL_MATH_WRAP, 1765
TRISYCL_MATH_WRAP2, 1765
TRISYCL_MATH_WRAP2s, 1766
TRISYCL_MATH_WRAP3, 1766
TRISYCL_MATH_WRAP3s, 1766
TRISYCL_MATH_WRAP3ss, 1767
max
trisycl, 836
trisycl::vendor::trisycl::random::xorshift< Nbits >, 1319
max_clock_frequency
Platforms, contexts, accessors and queues, 607
max_compute_units
Platforms, contexts, accessors and queues, 607
max_constant_args
Platforms, contexts, accessors and queues, 608
max_constant_buffer_size
Platforms, contexts, accessors and queues, 608
max_mem_alloc_size
Platforms, contexts, accessors and queues, 607
max_parameter_size
Platforms, contexts, accessors and queues, 608
max_read_image_args
Platforms, contexts, accessors and queues, 607
max_samplers
Platforms, contexts, accessors and queues, 608
max_size
trisycl::vendor::xilinx::latex::context, 934
max_work_group_size
Platforms, contexts, accessors and queues, 607
max_work_item_dimensions
Platforms, contexts, accessors and queues, 607
max_work_item_sizes
Platforms, contexts, accessors and queues, 607
max_write_image_args
Platforms, contexts, accessors and queues, 607
MAXFLOAT
opencl_spir.h, 1484
maxval
opencl_spir.h, 1513
mds
trisycl::mixin::accessor< T, Dimensions >::track_index< N >, 1297

mdspan
trisycl::mixin::accessor< T, Dimensions >, 515
me_0
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939–946
me_1
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939–944, 946
me_last
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939–944, 946
mem
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 983, 984
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 282
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1004, 1005
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 215–218
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 254, 255
mem_base_addr_align
Platforms, contexts, accessors and queues, 608
mem_east
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 219, 220
mem_north
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 220, 221
mem_south
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 221, 222
mem_t
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 186
mem_trace
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 944–947
mem_west
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 223, 224
memory_infrastructure
trisycl::vendor::xilinx::acap::aie::memory_infrastructure, 132
memory_locking_unit
trisycl::vendor::xilinx::acap::aie::memory_infrastructure, 133
memory_module
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 142–144
memory_module_dx_coordinate
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 105, 106
memory_module_linear_id
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 106, 107
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 224, 225
memory_modules
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 152
memory_modules_bases
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 152
message
trisycl::exception, 688
mi
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 295
trisycl::vendor::xilinx::acap::aie::memory_base, 131
min
trisycl, 837, 838
trisycl::vendor::trisycl::random::xorshift< Nbits >, 1320
min_capacity
trisycl::pipe< Name, T, MinCapacity, SizeT >, 374
minval
opencl_spir.h, 1513
mixin
trisycl::detail::buffer< T, Dimensions >, 348
trisycl::facade::accessor< AccessorMixIn >, 505
mode
trisycl::access, 842
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 476
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 494
modified
trisycl::detail::buffer< T, Dimensions >, 356
move_read_reservation_forward
trisycl::detail::sycl_2_2::pipe< T >, 452
move_write_reservation_forward
trisycl::detail::sycl_2_2::pipe< T >, 453
mpl
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 48, 49
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 269
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 164
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 248
mpl_outputs
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1240
mtx_
trisycl::detail::pooled_shared_work, 1180
trisycl::detail::pooled_work_stealing, 1188
multi_ptr
Dealing with OpenCL address spaces, 643
mutex_class
trisycl, 831

n
opencl_spir.h, 1513
name
Platforms, contexts, accessors and queues, 608
NAN
opencl_spir.h, 1485
native_vector_width_char
Platforms, contexts, accessors and queues, 607
native_vector_width_double
Platforms, contexts, accessors and queues, 607
native_vector_width_float
Platforms, contexts, accessors and queues, 607
native_vector_width_half
Platforms, contexts, accessors and queues, 607
native_vector_width_int
Platforms, contexts, accessors and queues, 607
native_vector_width_long
Platforms, contexts, accessors and queues, 607
native_vector_width_short
Platforms, contexts, accessors and queues, 607
nb_command
trisycl::vendor::xilinx::acap::aie::dma< AXIStream-Switch, SpecializedDMA >, 1026
nb_master_port
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 955
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1267
nb_slave_port
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 955
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1268
nd_item
trisycl::nd_item< Dimensions >, 753, 754
ND_range
trisycl::h_item< Dimensions >, 739
trisycl::nd_item< Dimensions >, 761
nd_range
trisycl::nd_range< Dimensions >, 762
ndr
trisycl::group< Dimensions >, 723
next_partitionable
Platforms, contexts, accessors and queues, 612
no_partition
Platforms, contexts, accessors and queues, 612
non_const_pointer
trisycl::mixin::accessor< T, Dimensions >, 516
none
Platforms, contexts, accessors and queues, 611, 612
trisycl, 833
trisycl::vendor::xilinx::partition, 871
normalize
trisycl, 838
north_0
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942, 944–947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255, 1257–1261
north_1
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942, 944–947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255, 1257–1261
north_2
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942, 944–947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255, 1257, 1259–1261
north_3
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942, 944–947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255, 1257, 1259–1261
north_4
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255, 1257
north_5
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255, 1257
north_last
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942, 944–947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255, 1257, 1259–1261
not_applicable
Platforms, contexts, accessors and queues, 612
notify
trisycl::detail::pooled_shared_work, 1178
trisycl::detail::pooled_work_stealing, 1185
notify_buffer_destructor
trisycl::detail::buffer_base, 367
notify_consumers
trisycl::detail::task, 1282
num_elements
opencl_spir.h, 1513
numa
Platforms, contexts, accessors and queues, 612
number_of_users
trisycl::detail::buffer_base, 367
nx

trisycl::vendor::xilinx::graphics::frame_grid, 1060
ny
trisycl::vendor::xilinx::graphics::frame_grid, 1060

odd
trisycl::vec< DataType, 16 >, 812
trisycl::vec< DataType, 2 >, 792
trisycl::vec< DataType, 3 >, 796
trisycl::vec< DataType, 4 >, 802
trisycl::vec< DataType, 8 >, 807, 808
offset
opencl_spir.h, 1513
trisycl::item< Dimensions, with_offset >, 751
trisycl::nd_range< Dimensions >, 766
ok
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 476
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 494
on_key_press_event
trisycl::vendor::xilinx::graphics::image_grid, 1085, 1086
opencl
Platforms, contexts, accessors and queues, 610
opencl_c_version
Platforms, contexts, accessors and queues, 608
opencl_context
trisycl::detail::opencl_context, 1113
opencl_device
trisycl::detail::opencl_device, 1119
opencl_device_tail.hpp
TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE, 1699
TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE, 1699
opencl_event
trisycl::detail::opencl_event, 1127
opencl_kernel
trisycl::detail::opencl_kernel, 1133
opencl_kernel.hpp
TRISYCL_ParallelForKernel_RANGE, 1760
opencl_platform
trisycl::detail::opencl_platform, 588
opencl_queue
trisycl::detail::opencl_queue, 1137
opencl_spir.h
__attribute__, 1486, 1499
__kernel_exec, 1454
a, 1510
as_char, 1454
as_char16, 1454
as_char2, 1455
as_char3, 1455
as_char4, 1455
as_char8, 1455
as_double, 1455
as_double16, 1455
as_double2, 1456
as_double3, 1456
as_double4, 1456
as_double8, 1456
as_float, 1456
as_float16, 1456
as_float2, 1457
as_float3, 1457
as_float4, 1457
as_float8, 1457
as_half, 1457
as_half16, 1457
as_half2, 1458
as_half3, 1458
as_half4, 1458
as_half8, 1458
as_int, 1458
as_int16, 1458
as_int2, 1459
as_int3, 1459
as_int4, 1459
as_int8, 1459
as_long, 1459
as_long16, 1459
as_long2, 1460
as_long3, 1460
as_long4, 1460
as_long8, 1460
as_short, 1460
as_short16, 1460
as_short2, 1461
as_short3, 1461
as_short4, 1461
as_short8, 1461
as_uchar, 1461
as_uchar16, 1461
as_uchar2, 1462
as_uchar3, 1462
as_uchar4, 1462
as_uchar8, 1462
as_uint, 1462
as_uint16, 1462
as_uint2, 1463
as_uint3, 1463
as_uint4, 1463
as_uint8, 1463
as_ulong, 1463
as_ulong16, 1463
as_ulong2, 1464
as_ulong3, 1464
as_ulong4, 1464
as_ulong8, 1464
as_ushort, 1464
as_ushort16, 1464
as_ushort2, 1465
as_ushort3, 1465
as_ushort4, 1465
as_ushort8, 1465
b, 1510
c, 1510

CHAR_BIT, 1465
CHAR_MAX, 1465
CHAR_MIN, 1466
cl_mem_fence_flags, 1497
CLK_A, 1466
CLK_ADDRESS_CLAMP, 1466
CLK_ADDRESS_CLAMP_TO_EDGE, 1466
CLK_ADDRESS_MIRRORED_REPEAT, 1466
CLK_ADDRESS_NONE, 1466
CLK_ADDRESS_REPEAT, 1467
CLK_ARGB, 1467
CLK_BGRA, 1467
CLK_DEPTH, 1467
CLK_DEPTH_STENCIL, 1467
CLK_FILTER_LINEAR, 1468
CLK_FILTER_NEAREST, 1468
CLK_FLOAT, 1468
CLK_GLOBAL_MEM_FENCE, 1468
CLK_HALF_FLOAT, 1468
CLK_INTENSITY, 1468
CLK_LOCAL_MEM_FENCE, 1469
CLK_LUMINANCE, 1469
CLK_NORMALIZED_COORDS_FALSE, 1469
CLK_NORMALIZED_COORDS_TRUE, 1469
CLK_R, 1469
CLK_RA, 1469
CLK_RG, 1470
CLK_RGB, 1470
CLK_RGBA, 1470
CLK_RGBx, 1470
CLK_RGx, 1470
CLK_Rx, 1470
CLK_SIGNED_INT16, 1471
CLK_SIGNED_INT32, 1471
CLK_SIGNED_INT8, 1471
CLK_SNORM_INT16, 1471
CLK_SNORM_INT8, 1471
CLK_UNORM_INT16, 1471
CLK_UNORM_INT24, 1472
CLK_UNORM_INT8, 1472
CLK_UNORM_INT_101010, 1472
CLK_UNORM_SHORT_555, 1472
CLK_UNORM_SHORT_565, 1472
CLK_UNSIGNED_INT16, 1472
CLK_UNSIGNED_INT32, 1473
CLK_UNSIGNED_INT8, 1473
cmp, 1510
color, 1510
const_func, 1473
coord, 1511
cosval, 1511
DBL_DIG, 1473
DBL_EPSILON, 1474
DBL_MANT_DIG, 1474
DBL_MAX, 1474
DBL_MAX_10_EXP, 1475
DBL_MAX_EXP, 1475
DBL_MIN, 1475
DBL_MIN_10_EXP, 1476
DBL_MIN_EXP, 1476
DBL_RADIX, 1476
dst_stride, 1511
edge1, 1511
event, 1511
event_list, 1511
exp, 1512
FLT_DIG, 1476
FLT_EPSILON, 1477
FLT_MANT_DIG, 1477
FLT_MAX, 1477
FLT_MAX_10_EXP, 1477
FLT_MAX_EXP, 1477
FLT_MIN, 1477
FLT_MIN_10_EXP, 1478
FLT_MIN_EXP, 1478
FLT_RADIX, 1478
FP_ILOGB0, 1478
FP_ILOGBNAN, 1478
HUGE_VAL, 1478
HUGE_VALF, 1479
i, 1512
INFINITY, 1479
INT_MAX, 1479
INT_MIN, 1479
intptr_t, 1497
iptr, 1512
kernel_exec, 1479
lo, 1512
LONG_MAX, 1480
LONG_MIN, 1480
M_1_PI, 1480
M_1_PI_F, 1480
M_2_PI, 1480
M_2_PI_F, 1481
M_2_SQRTPI, 1481
M_2_SQRTPI_F, 1481
M_E, 1481
M_E_F, 1481
M_LN10, 1481
M_LN10_F, 1482
M_LN2, 1482
M_LN2_F, 1482
M_LOG10E, 1482
M_LOG10E_F, 1482
M_LOG2E, 1482
M_LOG2E_F, 1483
M_PI, 1483
M_PI_2, 1483
M_PI_2_F, 1483
M_PI_4, 1483
M_PI_4_F, 1483
M_PI_F, 1484
M_SQRT1_2, 1484
M_SQRT1_2_F, 1484
M_SQRT2, 1484
M_SQRT2_F, 1484

mask, 1512
MAXFLOAT, 1484
maxval, 1513
minval, 1513
n, 1513
NAN, 1485
num_elements, 1513
offset, 1513
p, 1513
p1, 1514
printf, 1509
quo, 1514
readonly, 1485
sampler, 1514
SCHAR_MAX, 1485
SCHAR_MIN, 1485
SHRT_MAX, 1485
SHRT_MIN, 1485
signp, 1514
src, 1514
src_stride, 1515
uchar, 1498
UCHAR_MAX, 1486
uint, 1498
UINT_MAX, 1486
uintptr_t, 1498
ulong, 1498
ULONG_MAX, 1486
ushort, 1498
USHRT_MAX, 1486
val, 1515
x, 1515
y, 1515
z, 1516
Opencl_spir_helpers, 672
create_parallel_for_arg, 673
make_array_with_func, 673
make_spir_h_item, 674
make_spir_nd_range, 674
opencl_spir_req.h
__attribute__, 1675
__constant, 1674
__generic, 1674
__global, 1674
__local, 1674
__private, 1675
const_func, 1675
printf, 1687
readonly, 1675
opencl_type
trisycl::detail::address_space_base< T, AS >, 637
trisycl::detail::address_space_object< T, AS >, 632
trisycl::detail::address_space_variable< T, AS >, 639
opencl_types.hpp
TRISYCL_BOOST_COMPUTE_NAME, 1780
TRISYCL_DECLARE_CL_TYPES, 1780
TRISYCL_DEFINE_TYPES, 1780
TRISYCL_H_DEFINE_TYPE, 1780
TRISYCL_IS_WRAPPER_TRAIT, 1781
TRISYCL_SCALAR_TYPES, 1781
TRISYCL_SIZED_NAME, 1781
TRISYCL_TYPE_ACTUAL_NAME, 1782
TRISYCL_TYPE_CL_NAME, 1782
TRISYCL_TYPE_NAME, 1782
TRISYCL_TYPEDEF_TYPE, 1782
TRISYCL_WRAPPER_CLASS_2, 1782
TRISYCL_WRAPPER_CLASS_3, 1783
TRISYCL_WRAPPER_CLASS_4, 1783
operator BasicType
trisycl::detail::small_array_sycl< BasicType, FinalType, Dims >, 660
operator bool
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 470
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 492
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 483
operator DataType
trisycl::vec< DataType, 1 >, 788
operator FinalType
trisycl::detail::small_array< BasicType, FinalType, Dims >, 657
operator item< Dimensions, true >
trisycl::item< Dimensions, with_offset >, 749
operator opencl_type &
trisycl::detail::address_space_object< T, AS >, 632
trisycl::detail::address_space_variable< T, AS >, 640
operator<
trisycl::detail::shared_ptr_implementation< Parent, Implementation >, 1248
operator<<
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 470
trisycl::vendor::xilinx::acap::aie::communicator_port, 918
operator>
trisycl::half, 1061
operator>>
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 471
trisycl::vendor::xilinx::acap::aie::communicator_port, 919
operator*
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 313
trisycl::facade::accessor< AccessorMixIn >, 509
operator()
std::hash< trisycl::buffer< T, Dimensions, Allocator > >, 1062
std::hash< trisycl::context >, 1063
std::hash< trisycl::device >, 1063

std::hash< trisycl::event >, 1064
std::hash< trisycl::kernel >, 1065
std::hash< trisycl::platform >, 1066
std::hash< trisycl::queue >, 1067
trisycl::device_selector, 553
trisycl::device_type_selector, 555
trisycl::vendor::trisycl::random::xorshift< Nbits >, 1321, 1322
operator::trisycl::device &
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >, 385
operator::trisycl::handler &
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >, 911
operator::trisycl::platform &
trisycl::vendor::trisycl::scope::platform< PlatformStorage >, 396, 397
operator::trisycl::queue &
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 411
operator=
trisycl::detail::pooled_shared_work, 1178, 1179
trisycl::detail::pooled_work_stealing, 1185, 1186
operator==
trisycl::detail::shared_ptr_implementation< Parent, Implementation >, 1248
trisycl::group< Dimensions >, 722
trisycl::h_item< Dimensions >, 738
trisycl::item< Dimensions, with_offset >, 749
trisycl::nd_item< Dimensions >, 759
trisycl::nd_range< Dimensions >, 765
operator[]
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 314–317
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 492
trisycl::drt::accessor< Accessor >, 677
trisycl::facade::accessor< AccessorMixIn >, 510, 511
trisycl::group< Dimensions >, 722
trisycl::item< Dimensions, with_offset >, 750
trisycl::mixin::accessor< T, Dimensions >::track_index< N >, 1296
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 483
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >, 428, 429
order
trisycl::detail::accessor_base, 321
out
trisycl::vendor::xilinx::acap::aie::connection, 927, 928
trisycl::vendor::xilinx::acap::aie::connection::output, 1145
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 282, 283
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >, 238
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 255
trisycl::vendor::xilinx::latex::context, 934
out_connection
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 60, 61
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 284
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >, 238, 239
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 257
output
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 62
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 285, 286
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 174, 175
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 258
output_port
trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >, 1244
output_ports
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 65
outputs
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1240
overloaded
Some helpers for the implementation, 664
owner_queue
trisycl::detail::task, 1291

p
opencl_spir.h, 1513
trisycl::detail::opencl_platform, 591
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 494
trisycl::drt::code::program, 1201
trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >, 401
trisycl::vendor::xilinx::acap::aie::connection, 928
trisycl::vendor::xilinx::acap::aie::connection::base, 898
trisycl::vendor::xilinx::graphics::image_grid, 1094
p1
opencl_spir.h, 1514
palette
trisycl::vendor::xilinx::graphics::palette, 1150
palettize
trisycl::vendor::xilinx::graphics::palette, 1162, 1163
parallel_for
Expressing parallelism through kernels, 771, 773–775
trisycl::handler, 561–566

trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >, 912
parallel_for_global_offset
Expressing parallelism through kernels, 775
parallel_for_iterate
Expressing parallelism through kernels, 776
trisycl::detail::parallel_for_iterate< 0, Range, ParallelForFunctor, Id >, 768
trisycl::detail::parallel_for_iterate< level, Range, ParallelForFunctor, Id >, 767
parallel_for_work_group
trisycl::handler, 566, 567
parallel_for_work_item
trisycl::group< Dimensions >, 722
parallel_for_workgroup
Expressing parallelism through kernels, 777
parallel_for_workitem
Expressing parallelism through kernels, 778
parallel_for_workitem_in_group
Expressing parallelism through kernels, 780
param_traits.hpp
TRISYCL_INFO_PARAM_TRAITS, 1749
TRISYCL_INFO_PARAM_TRAITS_ANY_T, 1749
parent_device
Platforms, contexts, accessors and queues, 608
partition_affinity_domain
Platforms, contexts, accessors and queues, 612
partition_affinity_domains
Platforms, contexts, accessors and queues, 608
partition_array
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >, 422–425
partition_by_affinity_domain
Platforms, contexts, accessors and queues, 612
partition_by_counts
Platforms, contexts, accessors and queues, 612
partition_dim
trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim >, 900
trisycl::vendor::xilinx::partition::complete< PDim >, 925
trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim >, 962
partition_equally
Platforms, contexts, accessors and queues, 612
partition_max_sub_devices
Platforms, contexts, accessors and queues, 608
partition_properties
Platforms, contexts, accessors and queues, 608
partition_property
Platforms, contexts, accessors and queues, 612
partition_type
trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim >, 900
trisycl::vendor::xilinx::partition::complete< PDim >, 925
trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim >, 962
trisycl::vendor::xilinx::partition::none, 1111
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >, 430
partition_type_affinity_domain
Platforms, contexts, accessors and queues, 608
partition_type_property
Platforms, contexts, accessors and queues, 608
pc_shared
trisycl::detail::fiber_pool, 1046
pc_stealing
trisycl::detail::fiber_pool, 1046
phase
trisycl::vendor::xilinx::graphics::palette, 1169
physical_mem_num
trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim >, 962
pick_next
trisycl::detail::pooled_shared_work, 1179
trisycl::detail::pooled_work_stealing, 1186
pipe
trisycl::access, 844
trisycl::detail::sycl_2_2::pipe< T >, 449
trisycl::sycl_2_2::pipe< T >, 443
pipe_accessor
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 468
pipe_of
trisycl::vendor::xilinx::acap::aie::connection::base, 897, 898
pipe_reservation
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 487, 488
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 480, 481
pipeline
trisycl::vendor::xilinx, 865
platform
Platforms, contexts, accessors and queues, 606, 608, 613
trisycl::platform, 580
trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >, 399
trisycl::vendor::trisycl::scope::platform< PlatformStorage >, 394–396
platform.hpp
SYCL_VENDOR_TRISYCL_PLATFORM_SCOPE, 1890, 1904
platform::get_info< info::platform::extensions >
Platforms, contexts, accessors and queues, 616
platform_extensions
trisycl::detail::host_platform, 586
PLATFORM_GET_INFO_STRING
Platforms, contexts, accessors and queues, 604
platform_scope
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 411

trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >, 913
platform_scope_type
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >, 381
trisycl::vendor::trisycl::scope::platform< PlatformStorage >, 394
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 405
platform_type
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >, 381
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 405, 406
platform_with_scope
trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >, 391
Platforms, contexts, accessors and queues, 499
accelerator, 610
address_bits, 607
all, 610
automatic, 610
built_in_kernels, 608
context, 606
context::get_info< info::context::devices >, 614
context::get_info< info::context::platform >, 614
context::get_info< info::context::reference_count >, 614
correctly_rounded_divide_sqrt, 611
cpu, 610
cpu_selector, 604
custom, 610
default_selector, 604
denorm, 611
device, 607
device_exec_capabilities, 605
device_fp_config, 605
device_queue_properties, 605
device_type, 607, 609
devices, 606
double_fp_config, 608
driver_version, 608
error_correction_support, 608
exec_kernel, 610
exec_native_kernel, 610
execution_capabilities, 608
execution_capability, 610
extensions, 608
fma, 611
fp_config, 610
get_devices, 614, 615
global, 612
global_mem_cache_line_size, 608
global_mem_cache_size, 608
global_mem_cache_type, 608, 611
global_mem_size, 608
gpu, 610
gpu_selector, 605
half_fp_config, 608
host, 610
host_selector, 606
host_unified_memory, 608
image2d_max_height, 607
image2d_max_width, 607
image3d_max_depth, 608
image3d_max_height, 607
image3d_max_width, 607
image_max_array_size, 608
image_max_buffer_size, 608
image_support, 607
inf_nan, 611
is_available, 608
is_compiler_available, 608
is_endian_little, 608
is_linker_available, 608
L1_cache, 612
L2_cache, 612
L3_cache, 612
L4_cache, 612
local, 612
local_mem_size, 608
local_mem_type, 608, 611
max_clock_frequency, 607
max_compute_units, 607
max_constant_args, 608
max_constant_buffer_size, 608
max_mem_alloc_size, 607
max_parameter_size, 608
max_read_image_args, 607
max_samplers, 608
max_work_group_size, 607
max_work_item_dimensions, 607
max_work_item_sizes, 607
max_write_image_args, 607
mem_base_addr_align, 608
name, 608
native_vector_width_char, 607
native_vector_width_double, 607
native_vector_width_float, 607
native_vector_width_half, 607
native_vector_width_int, 607
native_vector_width_long, 607
native_vector_width_short, 607
next_partitionable, 612
no_partition, 612
none, 611, 612
not_applicable, 612
numa, 612
opencl, 610
opencl_c_version, 608
parent_device, 608
partition_affinity_domain, 612
partition_affinity_domains, 608
partition_by_affinity_domain, 612
partition_by_counts, 612

partition_equally, 612
partition_max_sub_devices, 608
partition_properties, 608
partition_property, 612
partition_type_affinity_domain, 608
partition_type_property, 608
platform, 606, 608, 613
platform::get_info< info::platform::extensions >, 616
PLATFORM_GET_INFO_STRING, 604
preferred_interop_user_sync, 608
preferred_vector_width_char, 607
preferred_vector_width_double, 607
preferred_vector_width_float, 607
preferred_vector_width_half, 607
preferred_vector_width_int, 607
preferred_vector_width_long, 607
preferred_vector_width_short, 607
printf_buffer_size, 608
profile, 608
profiling_timer_resolution, 608
queue::get_info< info::queue::context >, 616
queue::get_info< info::queue::device >, 616
queue::get_info< info::queue::reference_count >, 617
queue_profiling, 608
read_only, 611
reference_count, 606, 608
round_to_inf, 611
round_to_nearest, 611
round_to_zero, 611
select_device, 617, 618
single_fp_config, 608
soft_float, 611
TRISYCL_INFO_PARAM_TRAITS, 619, 620
TRISYCL_OPENCL_ONLY, 613
TRISYCL_WEAK_ATTRIB_SUFFIX, 620
vendor, 608
vendor_id, 607
version, 608
write_only, 611

pointer
trisycl::detail::container_element_aspect< T >, 651
trisycl::mixin::accessor< T, Dimensions >, 516
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 479

pointer_t
trisycl::detail::address_space_ptr< T, AS >, 634

pool_ctx
trisycl::detail::pooled_shared_work::pool_ctx, 1171
trisycl::detail::pooled_work_stealing::pool_ctx, 1173

pool_ctx_
trisycl::detail::pooled_shared_work, 1181
trisycl::detail::pooled_work_stealing, 1188

pooled_shared_work
trisycl::detail::pooled_shared_work, 1177

pooled_work_stealing
trisycl::detail::pooled_work_stealing, 1183, 1184

pop_command
trisycl::vendor::xilinx::acap::aie::dma< AXIStream-Switch, SpecializedDMA >, 1023, 1024

port
trisycl::vendor::xilinx::acap::aie::port::shim, 1250
trisycl::vendor::xilinx::acap::aie::port::tile, 1294

port_receiver
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, 1192

postlude
trisycl::detail::task, 1283

preferred_interop_user_sync
Platforms, contexts, accessors and queues, 608

preferred_vector_width_char
Platforms, contexts, accessors and queues, 607

preferred_vector_width_double
Platforms, contexts, accessors and queues, 607

preferred_vector_width_float
Platforms, contexts, accessors and queues, 607

preferred_vector_width_half
Platforms, contexts, accessors and queues, 607

preferred_vector_width_int
Platforms, contexts, accessors and queues, 607

preferred_vector_width_long
Platforms, contexts, accessors and queues, 607

preferred_vector_width_short
Platforms, contexts, accessors and queues, 607

prelude
trisycl::detail::task, 1283

prevent_arguments_from_optimization
trisycl::detail, 854

printf
opencl_spir.h, 1509
opencl_spir_req.h, 1687

printf_buffer_size
Platforms, contexts, accessors and queues, 608

priv
Dealing with OpenCL address spaces, 643

private_ptr
Dealing with OpenCL address spaces, 644

private_space
trisycl::access, 841

processor_array.cpp
main, 2218, 2220

producer_tasks
trisycl::detail::task, 1292

profile
Platforms, contexts, accessors and queues, 608

profiling_timer_resolution
Platforms, contexts, accessors and queues, 608

program
trisycl::drt::code::program, 1200
trisycl::program, 1204
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1005, 1006

trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 138
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >, 243
program_state
trisycl, 832
prologues
trisycl::detail::task, 1292
property_list
trisycl::property_list, 1208
property_list.hpp
TRISYCL_PROPERTY_CREATE, 1811
TRISYCL_PROPERTY_HAS_GET, 1811
push_command
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >, 1024

q
trisycl::detail::opencl_context, 1117
trisycl::detail::opencl_queue, 1141
trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >, 417
queue
trisycl::detail::queue, 1212
trisycl::info, 860
trisycl::queue, 596–598
trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >, 415
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 407–409
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1006, 1007
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >, 156
queue.hpp
SYCL_VENDOR_TRISYCL_QUEUE_SCOPE, 2127, 2148
queue::get_info< info::queue::context >
Platforms, contexts, accessors and queues, 616
queue::get_info< info::queue::device >
Platforms, contexts, accessors and queues, 616
queue::get_info< info::queue::reference_count >
Platforms, contexts, accessors and queues, 617
queue_profiling
Platforms, contexts, accessors and queues, 608
trisycl::info, 859
queue_scope_type
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 406
queue_tail.hpp
TRISYCL_DEFAULT_QUEUE_SELECTOR, 1818
quo
opencl_spir.h, 1514

r
trisycl::vec< DataType, 4 >, 802
Vector types in SYCL, 817
r_rid_q
trisycl::detail::sycl_2_2::pipe< T >, 464
rainbow
trisycl::vendor::xilinx::graphics::palette, 1149
rainbow_palette
trisycl::vendor::xilinx::graphics::palette, 1163, 1164
range
Expressing parallelism through kernels, 781
trisycl::range< Dimensions >, 769
rangeify
trisycl::detail, 851
rank
trisycl::buffer< T, Dimensions, Allocator >, 334
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 476
trisycl::group< Dimensions >, 723
trisycl::h_item< Dimensions >, 738
trisycl::id< Dimensions >, 742
trisycl::item< Dimensions, with_offset >, 750
trisycl::mixin::accessor< T, Dimensions >, 520
trisycl::nd_item< Dimensions >, 760
trisycl::nd_range< Dimensions >, 766
trisycl::range< Dimensions >, 769
rbegin
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 317
trisycl::facade::accessor< AccessorMixIn >, 512
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 483
read
trisycl::access, 842
trisycl::detail::sycl_2_2::pipe< T >, 454
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 472
trisycl::pipe< Name, T, MinCapacity, SizeT >, 372
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1235
trisycl::vendor::xilinx::acap::aie::communicator_port, 920
trisycl::vendor::xilinx::acap::aie::fifo_channel, 1050
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, 1193
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, 1221
read_done
trisycl::detail::sycl_2_2::pipe< T >, 464
read_only
Platforms, contexts, accessors and queues, 611
read_reserved_frozen
trisycl::detail::sycl_2_2::pipe< T >, 464
read_write
trisycl::access, 842
readonly
opencl_spir.h, 1485
opencl_spir_req.h, 1675
ready
trisycl::detail::buffer_base, 367
trisycl::detail::sycl_2_2::reserve_id< T >, 445
trisycl::detail::task, 1292
ready_mutex

trisycl::detail::buffer_base, 367
trisycl::detail::task, 1292
receive
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >, 78
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, 1222
receiving_dma
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, 1220, 1221
reference
trisycl::buffer< T, Dimensions, Allocator >, 326
trisycl::detail::container_element_aspect< T >, 651
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 467
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 487
trisycl::mixin::accessor< T, Dimensions >, 516
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 480
reference_count
Platforms, contexts, accessors and queues, 606, 608
trisycl::info, 859, 860
reference_t
trisycl::detail::address_space_ptr< T, AS >, 635
register_accessor
trisycl::detail::accessor< T, Dimensions, Mode, Target >, 343
trisycl::detail::task, 1284
release
trisycl::detail::buffer_base, 362
trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device, 1108
release_buffers
trisycl::detail::task, 1285
release_with_value
trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device, 1108, 1109
remove
trisycl::detail::cache< Key, Value >, 904
rend
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 317
trisycl::facade::accessor< AccessorMixIn >, 512
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 483
report_error
trisycl::error_handler, 684
trisycl::trisycl::default_error_handler, 964
reserve
trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >, 437
trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >, 434
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 473
reserve_id
trisycl::detail::sycl_2_2::reserve_id< T >, 445
reserve_read
trisycl::detail::sycl_2_2::pipe< T >, 455
reserve_write
trisycl::detail::sycl_2_2::pipe< T >, 457
reserved_for_reading
trisycl::detail::sycl_2_2::pipe< T >, 458
reserved_for_writing
trisycl::detail::sycl_2_2::pipe< T >, 459
result_type
trisycl::vendor::trisycl::random::xorshift< Nbits >, 1317
reverse_iterator
trisycl::facade::accessor< AccessorMixIn >, 505
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 480
rgb
trisycl::vendor::xilinx::graphics, 869
rid
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 494
rid_iterator
trisycl::detail::sycl_2_2::pipe< T >, 449
round_robin
trisycl::detail::fiber_pool, 1040
round_to_inf
Platforms, contexts, accessors and queues, 611
round_to_nearest
Platforms, contexts, accessors and queues, 611
round_to_zero
Platforms, contexts, accessors and queues, 611
rounding_mode
trisycl, 833
router_minion
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1229, 1231
rqueue_
trisycl::detail::pooled_shared_work::pool_ctx, 1171
trisycl::detail::pooled_work_stealing, 1188
rqueue_mtx_
trisycl::detail::pooled_shared_work::pool_ctx, 1171
rqueue_type
trisycl::detail::pooled_shared_work, 1176
rte
trisycl, 833
rtn
trisycl, 833
rtp
trisycl, 833
rtz
trisycl, 833
run
trisycl::detail::fiber_pool, 1043
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1007–1009

trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 145–147
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >, 156, 157
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >, 240
running_kernels
trisycl::detail::queue, 1215
rx_dma
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 287
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 225, 227
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 258, 259

s
trisycl::detail::fiber_pool, 1046
s0
Vector types in SYCL, 817
s1
Vector types in SYCL, 818
s2
Vector types in SYCL, 818
s3
Vector types in SYCL, 818
s4
Vector types in SYCL, 818
s5
Vector types in SYCL, 819
s6
Vector types in SYCL, 819
s7
Vector types in SYCL, 819
s8
Vector types in SYCL, 819
s9
Vector types in SYCL, 819
s_dma_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 955
s_dma_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 956
s_east_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 956
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1268
s_east_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 956
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1268
s_fifo_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 956
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1269
s_fifo_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 957
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1269
s_me_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 957
s_me_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 957
s_north_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 958
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1269
s_north_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 958
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1270
s_south_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 958
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1270
s_south_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 959
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1270
s_tile_ctrl_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 959
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1271
s_tile_ctrl_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 959
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1271
s_west_range
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 960
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1271
s_west_size
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 960
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1272
sA
Vector types in SYCL, 820
sampler
opencl_spir.h, 1514
sass
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 969, 970

trisycl::vendor::xilinx::acap::aie::device< Layout >, 996
sB
Vector types in SYCL, 820
sC
Vector types in SYCL, 820
scale
trisycl::vendor::xilinx::latex::context, 933
scaling_factor
trisycl::vendor::xilinx::latex::context, 934
SCHAR_MAX
opencl_spir.h, 1485
SCHAR_MIN
opencl_spir.h, 1485
sched
trisycl::detail::fiber_pool, 1040
schedule
trisycl::detail::task, 1285
schedule_kernel
trisycl::handler, 568
schedule_parallel_for_kernel
trisycl::handler, 569
schedulers_
trisycl::detail::pooled_work_stealing::pool_ctx, 1173
scope.hpp
SYCL_VENDOR_TRISYCL_CONCEPTUAL_API, 1884, 1885
SYCL_VENDOR_TRISYCL_SCOPE, 1884, 1885
scope_storage
trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >, 391
trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >, 401
trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >, 418
sD
Vector types in SYCL, 820
sE
Vector types in SYCL, 820
select_device
Platforms, contexts, accessors and queues, 617, 618
send
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >, 79
trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >, 1243
sending_dma
trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >, 1242, 1243
serialize_accessor_arg
Device-side runtime implementation, 678
serialize_arg
Device-side runtime implementation, 679
set
trisycl::item< Dimensions, with_offset >, 750
trisycl::vendor::xilinx::graphics::palette, 1165
set_access
trisycl::mixin::accessor< T, Dimensions >, 521
set_arg
trisycl::detail::task, 1287, 1288
trisycl::handler, 569, 570
set_args
trisycl::handler, 570
set_buffer
trisycl::detail::accessor< T, Dimensions, Mode, Target >, 344
set_close_action
trisycl::vendor::xilinx::graphics::frame_grid, 1058
set_debug
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 473
set_final_data
trisycl::buffer< T, Dimensions, Allocator >, 335
trisycl::detail::buffer< T, Dimensions >, 353, 354
set_global
trisycl::h_item< Dimensions >, 739
trisycl::nd_item< Dimensions >, 760
set_kernel
Device-side runtime implementation, 680
trisycl::detail::task, 1289
set_kernel_task_marker
trisycl::detail, 852
set_latest_producer
trisycl::detail::buffer_base, 362
set_local
trisycl::h_item< Dimensions >, 739
trisycl::nd_item< Dimensions >, 760
set_memory_infrastructure
trisycl::vendor::xilinx::acap::aie::memory_base, 130
set_order
trisycl::detail::accessor_base, 320
set_program
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >, 240
set_tile_infrastructure
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >, 240, 241
sF
Vector types in SYCL, 821
shared_ptr_class
trisycl, 832
shared_ptr_implementation
trisycl::detail::shared_ptr_implementation< Parent, Implementation >, 1246, 1247
trisycl::platform, 583
shared_work
trisycl::detail::fiber_pool, 1040
shim
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 985, 986
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1010
shim_tile

trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 165
shim_trace
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1259, 1261
shims
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 989
SHRT_MAX
opencl_spir.h, 1485
SHRT_MIN
opencl_spir.h, 1485
shutdown
trisycl::vendor::xilinx::acap::aie::axi_packet, 74
shutdown_request
trisycl::vendor::xilinx::acap::aie::axi_packet, 74
sign
trisycl, 838
signp
opencl_spir.h, 1514
single_fp_config
Platforms, contexts, accessors and queues, 608
single_task
trisycl::detail::kernel, 577
trisycl::detail::opencl_kernel, 1134
trisycl::handler, 571
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >, 914
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 288, 289
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >, 241
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 259, 260
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t, 907
size
trisycl::detail::sycl_2_2::pipe< T >, 460
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 474
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 493
trisycl::detail::sycl_2_2::reserve_id< T >, 445
trisycl::drt::accessor< Accessor >, 678
trisycl::range< Dimensions >, 769
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 484
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1013
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 114
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940–947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255–1257, 1259, 1261
trisycl::vendor::xilinx::graphics::palette, 1169
trisycl::vendor::xilinx::latex::context, 935
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >, 429
size_type
trisycl::pipe< Name, T, MinCapacity, SizeT >, 371
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 480
size_with_lock
trisycl::detail::sycl_2_2::pipe< T >, 461
slave_port_layout
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 943, 945
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1258, 1260
small_array
trisycl::detail::small_array< BasicType, FinalType, Dims >, 655, 656
small_array_sycl
trisycl::detail::small_array_sycl< BasicType, FinalType, Dims >, 659
smp
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 970
trisycl::vendor::xilinx::acap::aie::device< Layout >, 996, 997
soft_float
Platforms, contexts, accessors and queues, 611
Some helpers for the implementation, 650
linear_id, 663
overloaded, 664
TRISYCL_BOOST_OPERATOR_VECTOR_OP, 661
TRISYCL_LOGICAL_OPERATOR_VECTOR_OP, 661
TRISYCL_OPERATOR_BASIC_TYPE_OP, 661
TRISYCL_UNARY_OPERATOR_VECTOR_OP, 662
TRISYCL_UNARY_POSTFIX_OPERATOR_VECTOR_OP, 662
TRISYCL_UNARY_PREFIX_OPERATOR_VECTOR_OP, 662
TRISYCL_UNIMPL, 663
unimplemented, 665
south_0
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939–942, 944, 946
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1254–1261
south_1
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939, 940, 942, 944, 946
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255–1261
south_2
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939, 940,

942, 944, 946
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255–1261
south_3
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939, 940, 942, 944, 946
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255–1261
south_4
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 944, 946, 947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255–1261
south_5
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 944, 946, 947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255–1261
south_6
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1258–1261
south_7
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1258–1261
south_last
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942, 944, 946, 947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255–1261
spi
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >, 382, 386
trisycl::vendor::trisycl::scope::platform< PlatformStorage >, 394, 398
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 406, 413
spl
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 49
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 269
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 165
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 248, 249
src
opencl_spir.h, 1514
src_stride
opencl_spir.h, 1515
ssp
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 970, 971
trisycl::vendor::xilinx::acap::aie::device< Layout >, 997
start
trisycl::detail::sycl_2_2::reserve_id< T >, 445
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 63, 64
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 176, 177
trisycl::vendor::xilinx::graphics::application, 890, 891
starting_block
trisycl::detail::fiber_pool, 1046
starting_mode
trisycl::detail::fiber_pool, 1047
state
trisycl::vendor::trisycl::random::xorshift< Nbits >, 1323
static_pipe
trisycl::sycl_2_2::static_pipe< T, Capacity >, 497
std, 823
get, 824
std::hash< trisycl::buffer< T, Dimensions, Allocator > >, 1061
operator(), 1062
std::hash< trisycl::context >, 1062
operator(), 1063
std::hash< trisycl::device >, 1063
operator(), 1063
std::hash< trisycl::event >, 1064
operator(), 1064
std::hash< trisycl::kernel >, 1065
operator(), 1065
std::hash< trisycl::platform >, 1066
operator(), 1066
std::hash< trisycl::queue >, 1067
operator(), 1067
std::tuple_element< I, T >, 1297
type, 1298
std::tuple_element< I, trisycl::vec< T, Size > >, 1298
type, 1299
std::tuple_size< T >, 1299
std::tuple_size< trisycl::vec< T, Size > >, 1300
steal
trisycl::detail::pooled_work_stealing, 1187
storage_type
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >, 382
trisycl::vendor::trisycl::scope::platform< PlatformStorage >, 394
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 407
storage_type_trait_t
trisycl::vendor::trisycl::scope::detail, 864
stream
trisycl::vendor::trisycl::pipe::cout, 304
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input_port, 1101
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::output_port, 1147
stream_latency

trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 66
string_class
trisycl, 832
submission
trisycl::detail::fiber_pool, 1047
submit
trisycl::detail::fiber_pool, 1044
trisycl::queue, 601
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 412
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >, 158
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 261
trisycl::vendor::xilinx::graphics::image_grid, 1087
super
trisycl::detail::address_space_array< T, AS >, 627
trisycl::detail::address_space_fundamental< T, AS >, 629
trisycl::detail::address_space_ptr< T, AS >, 635
trisycl::detail::address_space_variable< T, AS >, 639
suspend_
trisycl::detail::pooled_shared_work::pool_ctx, 1171
trisycl::detail::pooled_work_stealing::pool_ctx, 1174
suspend_until
trisycl::detail::pooled_shared_work, 1179
trisycl::detail::pooled_work_stealing, 1187
sw
trisycl::vendor::xilinx::graphics::frame_grid, 1060
swizzle
trisycl::detail::vec< DataType, NumElements >, 1313, 1314
sycl, 825
sycl.hpp
TRISYCL_SYCL_NAMESPACE, 1325, 1327
SYCL_VENDOR_TRISYCL_CONCEPTUAL_API
scope.hpp, 1884, 1885
SYCL_VENDOR_TRISYCL_DEVICE_SCOPE
device.hpp, 1975, 2005
SYCL_VENDOR_TRISYCL_PLATFORM_SCOPE
platform.hpp, 1890, 1904
SYCL_VENDOR_TRISYCL_QUEUE_SCOPE
queue.hpp, 2127, 2148
SYCL_VENDOR_TRISYCL_SCOPE
scope.hpp, 1884, 1885
sync_with_host
trisycl::detail::buffer_base, 362

t
trisycl::vendor::xilinx::graphics::application, 895
target
trisycl::access, 842
trisycl::accessor< DataType, Dimensions, AccessMode, Target >, 318
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 476
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 495
task
trisycl::detail::accessor_base, 321
trisycl::detail::task, 1279
trisycl::handler, 572
tb
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 186
thread_count_
trisycl::detail::pooled_work_stealing::pool_ctx, 1174
throw_asynchronous
trisycl::queue, 602
ti
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 990
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >, 243
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t, 908
tikz_edge_angle
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 64, 65
tile
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 987, 988
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1011
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >, 81
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 148, 149
tile_bases
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 152
tile_coordinates
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 114
tile_ctrl
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939–944, 946
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1254–1261
tile_ctrl_last
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 939–944, 946
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1254–1261
tile_infrastructure
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 269, 270
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 249, 250
tileable_memory
trisycl::vendor::xilinx::acap::aie::program< AIEDe-

vice, Tile, Memory >, 137
tileable_tile
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 137, 138
tiles
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 153
tlast
trisycl::vendor::xilinx::acap::aie::axi_packet, 74
to_sycl_id
Expressing parallelism through kernels, 782
to_tbb_range
Expressing parallelism through kernels, 783, 784
trace_kernel
Debugging and tracing support, 668
track_access_mode
trisycl::detail::buffer< T, Dimensions >, 354
track_index
trisycl::mixin::accessor< T, Dimensions >::track_index< N >, 1295
translate_input_port
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 289, 290
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 177, 178
translate_output_port
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 290, 291
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 178, 179
translate_port
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 66
trisycl, 825
automatic, 833
compiled, 833
cross, 833
dot, 834
event::get_info< info::event::command_execution_status >, 834
event::get_info< info::event::reference_count >, 834
event::get_profiling_info< info::event_profiling::command_end >, 834
event::get_profiling_info< info::event_profiling::command_start >, 834
event::get_profiling_info< info::event_profiling::command_submit >, 835
exception_ptr_class, 831
fmax, 835
fmin, 835
function_class, 831
hash_class, 831
length, 836
linked, 833
max, 836
min, 837, 838
mutex_class, 831
none, 833
normalize, 838
program_state, 832
rounding_mode, 833
rte, 833
rtn, 833
rtp, 833
rtz, 833
shared_ptr_class, 832
sign, 838
string_class, 832
TRISYCL_MATH_WRAP, 839
TRISYCL_MATH_WRAP2s, 839
unique_ptr_class, 832
vector_class, 832
weak_ptr_class, 832
y, 840
z, 840
triSYCL extension for storage scopes, 378
trisycl::access, 840
address_space, 841
atomic, 842
blocking_pipe, 844
constant_buffer, 844
constant_space, 841
discard_read_write, 842
discard_write, 842
fence_space, 842
generic_space, 841
global_and_local, 842
global_buffer, 844
global_space, 841, 842
host_buffer, 844
host_image, 844
image, 844
image_array, 844
local, 844
local_space, 841, 842
mode, 842
pipe, 844
private_space, 841
read, 842
read_write, 842
target, 842
write, 842
trisycl::accessor, 305
trisycl::accessor< DataType, 1, AccessMode, access::target::blocking_pipe >, 434
accessor, 436
accessor_detail, 436
get_pipe_detail, 436
reserve, 437
trisycl::accessor< DataType, 1, AccessMode, access::target::pipe >, 432
accessor, 434
accessor_detail, 433
get_pipe_detail, 434
reserve, 434

trisycl::accessor< DataType, Dimensions, AccessMode, Target >
access_mode, 310
accessor, 308, 309
accessor_detail, 308
begin, 310
cbegin, 311
cend, 311
crbegin, 311
crend, 311
dimensionality, 318
end, 312
get_count, 312
get_pointer, 312
get_range, 312
get_size, 313
implementation_t, 308, 318
operator*, 313
operator[], 314–317
rbegin, 317
rend, 317
target, 318
trisycl::accessor_error, 693
trisycl::async_exception, 689
exception, 690
trisycl::buffer, 323
trisycl::buffer< T, Dimensions, Allocator >
allocator_type, 326
buffer, 326–331
const_reference, 326
get_access, 332
get_count, 333
get_range, 333
get_size, 333
implementation_t, 326, 336
is_cached, 333
is_data_up_to_date, 334
is_read_only, 334
mark_as_written, 334
rank, 334
reference, 326
set_final_data, 335
use_count, 336
value_type, 326
trisycl::cl_exception, 688
cl_code, 689
cl_exception, 689
get_cl_code, 689
trisycl::compile_program_error, 699
trisycl::context, 523
context, 526–528
get, 529
get_boost_compute, 529
get_boost_queue, 529
get_devices, 530
get_info, 530
get_platform, 530
implementation_t, 526
is_host, 530
trisycl::detail, 844
__swizzled_base_vec__, 849
add_buffer_to_task, 850
alignment_v, 853
all_true, 849
allow_any_dimension, 853
assign_from_tuple, 850
enum_count, 853
enum_distance, 853
instantiate_kernel, 850
is_range_v, 853
launch_device_kernel, 850
prevent_arguments_from_optimization, 854
rangeify, 851
set_kernel_task_marker, 852
underlying_value, 854
use_native_work_item, 854
trisycl::detail::accessor, 336
trisycl::detail::accessor< T, Dimensions, Mode, access::target::local >, 321
accessor, 322
buf, 323
hb, 322
trisycl::detail::accessor< T, Dimensions, Mode, Target >
accessor, 339, 340
buf, 345
copy_back_cl_buffer, 341
copy_in_cl_buffer, 341
facade, 338
get_buffer, 341
get_cl_buffer, 342
handler, 345
is_read_access, 342
is_write_access, 342
register_accessor, 343
set_buffer, 344
trisycl::detail::accessor_base, 318
~accessor_base, 320
get_cl_buffer, 320
get_order, 320
order, 321
set_order, 320
task, 321
trisycl::detail::address_space_array, 625
trisycl::detail::address_space_array< T, AS >
address_space_array, 627
super, 627
trisycl::detail::address_space_base, 635
trisycl::detail::address_space_base< T, AS >
address_space, 637
opencl_type, 637
type, 637
trisycl::detail::address_space_fundamental, 627
trisycl::detail::address_space_fundamental< T, AS >
address_space_fundamental, 630
super, 629
trisycl::detail::address_space_object, 630

trisycl::detail::address_space_object< T, AS >
address_space_object, 632
opencl_type, 632
operator opencl_type &, 632
trisycl::detail::address_space_ptr, 632
trisycl::detail::address_space_ptr< T, AS >
address_space_ptr, 635
pointer_t, 634
reference_t, 635
super, 635
trisycl::detail::address_space_variable, 637
trisycl::detail::address_space_variable< T, AS >
address_space_variable, 639
get_address, 639
opencl_type, 639
operator opencl_type &, 640
super, 639
variable, 640
trisycl::detail::alignment< T >, 884
value, 885
trisycl::detail::alignment<::trisycl::vec< DataType, NumElements > >, 885
get_align, 886
value, 886
trisycl::detail::bool_pack<... >, 901
trisycl::detail::buffer, 345
trisycl::detail::buffer< T, Dimensions >
~buffer, 350
alloc, 355
allocate_buffer, 351
allocation, 355
assign, 351
buffer, 348–350
call_update_buffer_state, 351, 352
copy_if_modified, 355
data_host, 355
deallocate_buffer, 352
detail::accessor, 355
final_write_back, 355
get_destructor_future, 352
host_context, 356
input_shared_pointer, 356
mark_as_written, 353
mixin, 348
modified, 356
set_final_data, 353, 354
track_access_mode, 354
trisycl::detail::buffer_base, 356
~buffer_base, 358
add_to_task, 359
buffer_base, 358
buffer_cache, 366
create_in_cache, 359
fresh_ctx, 366
get_cl_buffer, 360
get_latest_producer, 361
is_cached, 361
is_data_up_to_date, 361
latest_producer, 366
latest_producer_mutex, 367
notify_buffer_destructor, 367
number_of_users, 367
ready, 367
ready_mutex, 367
release, 362
set_latest_producer, 362
sync_with_host, 362
update_buffer_state, 363
use, 365
wait, 365
trisycl::detail::buffer_waiter, 368
trisycl::detail::buffer_waiter< T, Dimensions, Allocator >
~buffer_waiter, 369
buffer_waiter, 369
implementation_t, 369, 370
trisycl::detail::cache< Key, Value >, 901
c, 905
get_or_register, 903
key_type, 902
m, 905
remove, 904
value_type, 902
trisycl::detail::container_element_aspect, 651
trisycl::detail::container_element_aspect< T >
const_pointer, 651
const_reference, 651
pointer, 651
reference, 651
value_type, 651
trisycl::detail::context, 531
~context, 532
get, 532
get_boost_compute, 532
get_boost_queue, 532
get_devices, 532
get_platform, 533
get_reference_count, 533
is_host, 533
trisycl::detail::debug, 666
trisycl::detail::debug< T >
type_pretty_name, 667
trisycl::detail::device, 547
~device, 548
get, 548
get_boost_compute, 549
get_info, 549
get_platform, 549
has_extension, 549
is_accelerator, 549
is_cpu, 549
is_gpu, 550
is_host, 550
trisycl::detail::display_vector, 667
trisycl::detail::display_vector< T >
display, 667
trisycl::detail::event, 1029

~event, 1030
get, 1030
get_boost_compute, 1030
get_command_execution_status, 1031
get_profiling_info, 1031
get_reference_count, 1031
is_host, 1031
wait, 1031
trisycl::detail::expand_to_vector, 646
trisycl::detail::expand_to_vector< V, Tuple, true >, 646
trisycl::detail::fiber_pool, 1038
~fiber_pool, 1041
close, 1042
fiber_pool, 1040
finish_line, 1045
future, 1040
join, 1042
joinable, 1046
pc_shared, 1046
pc_stealing, 1046
round_robin, 1040
run, 1043
s, 1046
sched, 1040
shared_work, 1040
starting_block, 1046
starting_mode, 1047
submission, 1047
submit, 1044
work_stealing, 1040
working_threads, 1047
trisycl::detail::host_context, 533
get, 535
get_boost_compute, 535
get_boost_queue, 535
get_devices, 536
get_platform, 536
get_reference_count, 536
is_host, 537
trisycl::detail::host_device, 1068
get, 1069
get_boost_compute, 1069
get_info, 1070
get_platform, 1070
has_extension, 1070
is_accelerator, 1071
is_cpu, 1071
is_gpu, 1071
is_host, 1072
trisycl::detail::host_event, 1072
get, 1073
get_boost_compute, 1074
get_command_execution_status, 1074
get_profiling_info, 1074
get_reference_count, 1074
is_host, 1075
wait, 1075
trisycl::detail::host_platform, 583
get, 584
get_boost_compute, 584
get_extension_strings, 585
get_info_string, 585
has_extension, 585
is_host, 586
platform_extensions, 586
trisycl::detail::host_queue, 1076
get, 1077
get_boost_compute, 1077
get_context, 1077
get_device, 1078
is_host, 1078
trisycl::detail::is_range< range< Dimensions > >, 1103
trisycl::detail::is_range< T >, 1101
trisycl::detail::is_small_array, 875
trisycl::detail::kernel, 575
get, 577
get_boost_compute, 577
single_task, 577
TRISYCL_ParallelForKernel_RANGE, 577
trisycl::detail::ocl_type, 623
trisycl::detail::ocl_type< T, access::address_space::constant_space >, 623
type, 623
trisycl::detail::ocl_type< T, access::address_space::global_space >, 623
type, 624
trisycl::detail::ocl_type< T, access::address_space::local_space >, 624
type, 624
trisycl::detail::ocl_type< T, access::address_space::private_space >, 624
type, 625
trisycl::detail::ocl_type< T, AS >
type, 623
trisycl::detail::opencl_context, 1111
~opencl_context, 1113
c, 1117
cache, 1117
get, 1114
get_boost_compute, 1114
get_boost_queue, 1114
get_devices, 1114
get_platform, 1115
get_reference_count, 1115
instance, 1115
is_host, 1116
opencl_context, 1113
q, 1117
trisycl::detail::opencl_device, 1118
~opencl_device, 1119
cache, 1125
d, 1125
get, 1120
get_boost_compute, 1120
get_info, 1120, 1121
get_platform, 1122

has_extension, 1123
instance, 1123
is_accelerator, 1123
is_cpu, 1124
is_gpu, 1124
is_host, 1124
opencl_device, 1119
trisycl::detail::opencl_event, 1126
~opencl_event, 1127
cache, 1131
e, 1131
get, 1128
get_boost_compute, 1128
get_command_execution_status, 1128
get_profiling_info, 1129
get_reference_count, 1129
instance, 1129
is_host, 1130
opencl_event, 1127
wait, 1130
trisycl::detail::opencl_kernel, 1131
cache, 1135
get, 1133
get_boost_compute, 1133
instance, 1134
k, 1135
opencl_kernel, 1133
single_task, 1134
TRISYCL_ParallelForKernel_RANGE, 1134
trisycl::detail::opencl_platform, 586
~opencl_platform, 588
cache, 591
get, 589
get_boost_compute, 589
get_extension_strings, 589
get_info_string, 589
has_extension, 590
instance, 590
is_host, 590
opencl_platform, 588
p, 591
trisycl::detail::opencl_queue, 1136
~opencl_queue, 1138
cache, 1141
get, 1138
get_boost_compute, 1138
get_context, 1139
get_device, 1139
instance, 1139, 1140
is_host, 1140
opencl_queue, 1137
q, 1141
trisycl::detail::overloaded, 652
trisycl::detail::parallel_for_iterate, 766
trisycl::detail::parallel_for_iterate< 0, Range, ParallelForFunctor, Id >, 767
parallel_for_iterate, 768
trisycl::detail::parallel_for_iterate< level, Range, ParallelForFunctor, Id >
parallel_for_iterate, 767
trisycl::detail::platform, 591
~platform, 592
get, 592
get_boost_compute, 592
get_devices, 592
get_extension_strings, 593
get_info_string, 593
has_extension, 593
is_host, 593
trisycl::detail::pooled_shared_work, 1174
awakened, 1177
cnd_, 1180
create_pool_ctx, 1177
ctx, 1176
flag_, 1180
has_ready_fibers, 1178
lqueue_, 1180
lqueue_type, 1176
mtx_, 1180
notify, 1178
operator=, 1178, 1179
pick_next, 1179
pool_ctx_, 1181
pooled_shared_work, 1177
rqueue_type, 1176
suspend_until, 1179
trisycl::detail::pooled_shared_work::pool_ctx, 1170
pool_ctx, 1171
rqueue_, 1171
rqueue_mtx_, 1171
suspend_, 1171
trisycl::detail::pooled_work_stealing, 1181
~pooled_work_stealing, 1183
awakened, 1184
cnd_, 1187
create_pool_ctx, 1184
ctx, 1183
flag_, 1188
has_ready_fibers, 1185
id_, 1188
mtx_, 1188
notify, 1185
operator=, 1185, 1186
pick_next, 1186
pool_ctx_, 1188
pooled_work_stealing, 1183, 1184
rqueue_, 1188
steal, 1187
suspend_until, 1187
trisycl::detail::pooled_work_stealing::pool_ctx, 1172
barrier_, 1173
counter_, 1173
pool_ctx, 1173
schedulers_, 1173
suspend_, 1174

thread_count_, 1174
trisycl::detail::program, 1198
~program, 1199
get, 1199
trisycl::detail::property, 1206
trisycl::detail::queue, 1210
~queue, 1212
finished, 1215
finished_mutex, 1215
get, 1213
get_boost_compute, 1213
get_context, 1213
get_device, 1213
is_host, 1213
kernel_end, 1214
kernel_start, 1214
queue, 1212
running_kernels, 1215
wait_for_kernel_execution, 1214
trisycl::detail::shared_ptr_implementation< Parent, Implementation >, 1245
hash, 1247
implementation, 1249
operator<, 1248
operator==, 1248
shared_ptr_implementation, 1246, 1247
trisycl::detail::singleton< T >, 1273
instance, 1273
trisycl::detail::small_array, 653
trisycl::detail::small_array< BasicType, FinalType, Dims >
dimension, 657
element_type, 655
get, 656
operator FinalType, 657
small_array, 655, 656
trisycl::detail::small_array_sycl, 657
trisycl::detail::small_array_sycl< BasicType, FinalType, Dims >
operator BasicType, 660
small_array_sycl, 659
trisycl::detail::small_array_tag, 652
trisycl::detail::spir, 855
trisycl::detail::sycl_2_2, 855
trisycl::detail::sycl_2_2::pipe, 446
trisycl::detail::sycl_2_2::pipe< T >
capacity, 449
cb, 463
cb_mutex, 463
debug_mode, 463
empty, 450
empty_with_lock, 451
full, 451
full_with_lock, 452
implementation_t, 449
move_read_reservation_forward, 452
move_write_reservation_forward, 453
pipe, 449
r_rid_q, 464
read, 454
read_done, 464
read_reserved_frozen, 464
reserve_read, 455
reserve_write, 457
reserved_for_reading, 458
reserved_for_writing, 459
rid_iterator, 449
size, 460
size_with_lock, 461
used_for_reading, 464
used_for_writing, 464
value_type, 449
w_rid_q, 465
write, 462
write_done, 465
trisycl::detail::sycl_2_2::pipe_accessor, 465
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >
~pipe_accessor, 468
blocking, 475
capacity, 469
const_reference, 467
empty, 469
full, 469
get_pipe_detail, 470
implementation, 475
mode, 476
ok, 476
operator bool, 470
operator<<, 470
operator>>, 471
pipe_accessor, 468
rank, 476
read, 472
reference, 467
reserve, 473
set_debug, 473
size, 474
target, 476
value_type, 467
write, 474
trisycl::detail::sycl_2_2::pipe_reservation, 484
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >
~pipe_reservation, 489
accessor_type, 487
assume_validity, 489
begin, 490
blocking, 494
commit, 490
const_iterator, 487
end, 491
iterator, 487
mode, 494
ok, 494
operator bool, 492

operator[], 492
p, 494
pipe_reservation, 487, 488
reference, 487
rid, 494
size, 493
target, 495
value_type, 487
trisycl::detail::sycl_2_2::reserve_id, 444
trisycl::detail::sycl_2_2::reserve_id< T >
ready, 445
reserve_id, 445
size, 445
start, 445
trisycl::detail::task, 1276
accessors, 1290
add_buffer, 1279
add_postlude, 1279
add_prelude, 1280
buffers_in_use, 1290
epilogues, 1291
execution_ended, 1291
get_compute_buffer, 1280
get_kernel, 1281
get_queue, 1281
kernel, 1291
notify_consumers, 1282
owner_queue, 1291
postlude, 1283
prelude, 1283
producer_tasks, 1292
prologues, 1292
ready, 1292
ready_mutex, 1292
register_accessor, 1284
release_buffers, 1285
schedule, 1285
set_arg, 1287, 1288
set_kernel, 1289
task, 1279
wait, 1289
wait_for_producers, 1289
trisycl::detail::vec< DataType, NumElements >, 1302
apply_binary_functor, 1306
apply_binary_functor_impl, 1306
apply_ternary_functor_impl, 1307
apply_unary_functor, 1307
apply_unary_functor_impl, 1308
as, 1309
basic_type, 1305
convert, 1309
flatten, 1310
flatten_to_tuple, 1311
get_count, 1311
get_size, 1312
map, 1312
swizzle, 1313, 1314
vec, 1305
zip, 1314
trisycl::device, 537
create_sub_devices, 541
device, 539, 540
get, 541
get_boost_compute, 542
get_devices, 542
get_info, 543
get_platform, 543
has_extension, 544
implementation_t, 539
is_accelerator, 544
is_cpu, 544
is_gpu, 545
is_host, 545
type, 546
trisycl::device_error, 697
exception, 699
trisycl::device_selector, 551
~device_selector, 552
operator(), 553
trisycl::device_type_selector, 553
default_device, 555
device_type, 556
device_type_selector, 554
operator(), 555
trisycl::device_typename_selector, 556
trisycl::device_typename_selector< DeviceType >
device_typename_selector, 557
trisycl::drt, 855
trisycl::drt::accessor, 676
trisycl::drt::accessor< Accessor >
accessor, 677
buffer, 678
operator[], 677
size, 678
trisycl::drt::code, 856
trisycl::drt::code::program, 1199
binary, 1200
binary_size, 1201
p, 1201
program, 1200
trisycl::elem, 786
trisycl::error_handler, 683
~error_handler, 684
default_handler, 684
report_error, 684
trisycl::event, 1032
event, 1034, 1035
get, 1035
get_boost_compute, 1035
get_info, 1036
get_profiling_info, 1036
get_wait_list, 1036
implementation_t, 1034, 1038
is_host, 1036
wait, 1037
wait_and_throw, 1037

trisycl::event_error, 695
trisycl::exception, 685
exception, 687
message, 688
what, 687
trisycl::exception_list, 684
trisycl::extension, 856
trisycl::extension::ce, 857
trisycl::facade, 857
trisycl::facade::accessor, 503
trisycl::facade::accessor< AccessorMixIn >
begin, 506
cbegin, 506
cend, 507
const_iterator, 505
const_reverse_iterator, 505
crbegin, 507
crend, 508
end, 508
get_pointer, 509
iterator, 505
mixin, 505
operator*, 509
operator[], 510, 511
rbegin, 512
rend, 512
reverse_iterator, 505
trisycl::facade::device, 550
trisycl::facade::device< UserFacing, ImplementationDetail >
detail_t, 551
implementation_t, 551
trisycl::feature_not_supported, 706
trisycl::group, 711
trisycl::group< Dimensions >
get_global_range, 714
get_group_range, 715, 716
get_id, 716, 717
get_linear_id, 718
get_local_range, 718, 719
get_nd_range, 720
get_offset, 720, 721
group, 713
group_id, 723
ndr, 723
operator==, 722
operator[], 722
parallel_for_work_item, 722
rank, 723
trisycl::h_item, 724
trisycl::h_item< Dimensions >
barrier, 727
get_global_id, 727, 728
get_global_linear_id, 728
get_global_range, 729
get_group, 730, 731
get_group_linear_id, 731
get_group_range, 732, 733
get_item, 733
get_local_id, 734
get_local_linear_id, 735
get_local_range, 736
get_nd_range, 737
get_offset, 737
global_index, 739
h_item, 726
local_index, 739
ND_range, 739
operator==, 738
rank, 738
set_global, 739
set_local, 739
trisycl::half, 1061
half, 1061
operator>, 1061
trisycl::handler, 557
dispatch_set_arg, 560
handler, 560
parallel_for, 561–566
parallel_for_work_group, 566, 567
schedule_kernel, 568
schedule_parallel_for_kernel, 569
set_arg, 569, 570
set_args, 570
single_task, 571
task, 572
trisycl::host_accessor, 437
trisycl::host_accessor< DataType, Dimensions, AccessMode >
accessor, 439, 440
base, 438
host_accessor, 439
trisycl::id, 740
trisycl::id< Dimensions >
id, 741
rank, 742
trisycl::image, 370
trisycl::info, 857
command_execution_status, 859
context, 860
device, 860
event, 859
event_command_status, 859
event_profiling, 860
queue, 860
queue_profiling, 859
reference_count, 859, 860
TRISYCL_INFO_PARAM_TRAITS, 861
TRISYCL_OPENCL_ONLY, 860
trisycl::info::param_traits< T, Param >, 1170
trisycl::invalid_object_error, 701
trisycl::invalid_parameter_error, 696
trisycl::is_wrapper< T >, 1104
trisycl::item, 742
trisycl::item< Dimensions, with_offset >
display, 744

get_id, 744, 745
get_linear_id, 746
get_offset, 747
get_range, 747, 748
global_index, 751
global_range, 751
item, 744
offset, 751
operator item< Dimensions, true >, 749
operator==, 749
operator[], 750
rank, 750
set, 750
trisycl::kernel, 572
get, 575
handler, 575
implementation_t, 574, 575
kernel, 574
trisycl::kernel_error, 692
trisycl::link_program_error, 700
trisycl::memory_allocation_error, 702
trisycl::mixin, 861
trisycl::mixin::accessor, 513
trisycl::mixin::accessor< T, Dimensions >
access, 523
accessor, 516, 517
const_pointer, 515
data, 517
element_type, 515
extents_cast, 517
get_count, 518
get_range, 519
get_size, 519
mdspan, 515
non_const_pointer, 516
pointer, 516
rank, 520
reference, 516
set_access, 521
tuple_indexed_access, 521
tuple_indexed_mdspan_access, 522
update, 523
value_type, 516
trisycl::mixin::accessor< T, Dimensions >::track_index< N >, 1294
indices, 1297
mds, 1297
operator[], 1296
track_index, 1295
trisycl::native, 861
trisycl::nd_item, 751
trisycl::nd_item< Dimensions >
barrier, 754
get_global_id, 755
get_global_linear_id, 755
get_global_range, 756
get_group, 756
get_group_linear_id, 757
get_group_range, 757
get_item, 758
get_local_id, 758
get_local_linear_id, 758
get_local_range, 759
get_nd_range, 759
get_offset, 759
global_index, 760
local_index, 760
nd_item, 753, 754
ND_range, 761
operator==, 759
rank, 760
set_global, 760
set_local, 760
trisycl::nd_range, 761
trisycl::nd_range< Dimensions >
display, 763
get_global_range, 763
get_group_range, 764
get_local_range, 765
get_offset, 765
global_range, 766
local_range, 766
nd_range, 762
offset, 766
operator==, 765
rank, 766
trisycl::nd_range_error, 694
trisycl::non_cl_error, 707
trisycl::pipe, 370
trisycl::pipe< Name, T, MinCapacity, SizeT >
bit_ceil_at_least_2, 372
channel, 374
min_capacity, 374
read, 372
size_type, 371
value_type, 371
write, 373
trisycl::pipe_error, 703
trisycl::platform, 578
get, 581
get_boost_compute, 581
get_devices, 581
get_info, 582
get_platforms, 582
has_extension, 582
is_host, 583
platform, 580
shared_ptr_implementation, 583
trisycl::platform_error, 704
trisycl::profiling_error, 705
trisycl::program, 1202
build_with_kernel_type, 1204
compile_with_kernel_type, 1204
get, 1204
get_kernel, 1205
handler, 1205

has_kernel, 1205
implementation_t, 1203, 1206
program, 1204
trisycl::property, 861
trisycl::property::queue, 862
trisycl::property::queue::enable_profiling, 1028
enable_profiling, 1028
trisycl::property_list, 1207
addproperty, 1208
get_property, 1209
has_property, 1209
property_list, 1208
TRISYCL_PROPERTY_CREATE, 1210
trisycl::queue, 593
get, 599
get_boost_compute, 599
get_context, 599
get_device, 600
get_info, 600
get_property, 600
has_property, 600
implementation_t, 596, 604
is_host, 600
queue, 596–598
submit, 601
throw_asynchronous, 602
wait, 602
wait_and_throw, 603
trisycl::range, 768
trisycl::range< Dimensions >
range, 769
rank, 769
size, 769
trisycl::runtime_error, 691
exception, 692
trisycl::sycl_2_2, 862
trisycl::sycl_2_2::pipe, 441
trisycl::sycl_2_2::pipe< T >
capacity, 443
get_access, 443
implementation_t, 442, 444
pipe, 443
value_type, 442
trisycl::sycl_2_2::pipe_reservation, 477
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >
accessor_detail, 478
accessor_type, 478
begin, 481
blocking, 484
cbegin, 481
cend, 482
commit, 482
const_iterator, 479
const_pointer, 479
const_reference, 479
const_reverse_iterator, 479
crbegin, 482
crend, 482
difference_type, 479
end, 482
implementation, 484
iterator, 479
operator bool, 483
operator[], 483
pipe_reservation, 480, 481
pointer, 479
rbegin, 483
reference, 480
rend, 483
reverse_iterator, 480
size, 484
size_type, 480
value_type, 480
trisycl::sycl_2_2::static_pipe, 495
trisycl::sycl_2_2::static_pipe< T, Capacity >
capacity, 497
get_access, 497, 498
implementation_t, 496, 498
static_pipe, 497
value_type, 496
trisycl::trisycl, 862
trisycl::trisycl::default_error_handler, 963
report_error, 964
trisycl::vec< DataType, 1 >, 787
base_vec, 788
operator DataType, 788
TRISYCL_DECLARE_S, 789
x, 789
trisycl::vec< DataType, 16 >, 809
base_vec, 810
even, 811
hi, 811
lo, 811, 812
odd, 812
TRISYCL_DECLARE_S, 812–814
trisycl::vec< DataType, 2 >, 789
base_vec, 790
even, 790, 791
hi, 791
lo, 791
odd, 792
TRISYCL_DECLARE_S, 792
x, 792
y, 793
trisycl::vec< DataType, 3 >, 793
base_vec, 794
even, 795
hi, 795
lo, 795
odd, 796
TRISYCL_DECLARE_S, 796
x, 796, 797
y, 797
z, 797
trisycl::vec< DataType, 4 >, 798
a, 800

b, 800
base_vec, 799
even, 800
g, 801
hi, 801
lo, 801, 802
odd, 802
r, 802
TRISYCL_DECLARE_S, 803
w, 803
x, 803, 804
y, 804
z, 804
trisycl::vec< DataType, 8 >, 805
base_vec, 806
even, 806
hi, 807
lo, 807
odd, 807, 808
TRISYCL_DECLARE_S, 808, 809
trisycl::vec< typename, int >, 1315
trisycl::vendor, 862
trisycl::vendor::trisycl, 862
trisycl::vendor::trisycl::pipe, 863
trisycl::vendor::trisycl::pipe::cout, 303
stream, 304
write, 304, 305
trisycl::vendor::trisycl::random, 863
trisycl::vendor::trisycl::random::xorshift< Nbits >, 1316
bit_size, 1322
initial_state, 1322
max, 1319
min, 1320
operator(), 1321, 1322
result_type, 1317
state, 1323
value_type, 1318
xorshift, 1318, 1319
trisycl::vendor::trisycl::scope, 863
trisycl::vendor::trisycl::scope::detail, 864
storage_type_trait_t, 864
trisycl::vendor::trisycl::scope::detail::device, 387
trisycl::vendor::trisycl::scope::detail::device< DeviceStorage, ScopedPlatform >
d, 391
device, 388, 389
get_platform, 389
get_storage, 390
get_underlying_device, 390
platform_with_scope, 391
scope_storage, 391
trisycl::vendor::trisycl::scope::detail::platform, 398
trisycl::vendor::trisycl::scope::detail::platform< PlatformStorage >
get_storage, 400
get_underlying_platform, 400
p, 401
platform, 399
scope_storage, 401
trisycl::vendor::trisycl::scope::detail::queue, 413
trisycl::vendor::trisycl::scope::detail::queue< Device, QueueStorage >
device_with_scope, 417
get_device, 416
get_storage, 416
get_underlying_queue, 417
q, 417
queue, 415
scope_storage, 418
trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, StorageType >, 1274
type, 1274
trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, typename ScopedType::storage_type >, 1275
type, 1275
trisycl::vendor::trisycl::scope::device, 378
trisycl::vendor::trisycl::scope::device< DeviceStorage, ScopedPlatform >
device, 382–384
device_scope_type, 380, 381
get_platform, 384
get_storage, 385
has_some_storage_p, 386
implementation, 386
operator::trisycl::device &, 385
platform_scope_type, 381
platform_type, 381
spi, 382, 386
storage_type, 382
trisycl::vendor::trisycl::scope::empty_device_scope, 378
trisycl::vendor::trisycl::scope::empty_platform_scope, 391
trisycl::vendor::trisycl::scope::empty_queue_scope, 401
trisycl::vendor::trisycl::scope::platform, 392
trisycl::vendor::trisycl::scope::platform< PlatformStorage >
get_storage, 396
has_some_storage_p, 397
implementation, 397
operator::trisycl::platform &, 396, 397
platform, 394–396
platform_scope_type, 394
spi, 394, 398
storage_type, 394
trisycl::vendor::trisycl::scope::queue, 401
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >
device_scope, 409
device_scope_type, 404
device_type, 405
get_device, 409
get_platform, 410
get_storage, 410
has_some_storage_p, 413
implementation, 413

operator::trisycl::queue &, 411
platform_scope, 411
platform_scope_type, 405
platform_type, 405, 406
queue, 407–409
queue_scope_type, 406
spi, 406, 413
storage_type, 407
submit, 412
wait, 412
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >::command_group< CGH >, 909
cgh, 915
command_group, 910
d, 915
device_scope, 911
operator::trisycl::handler &, 911
parallel_for, 912
platform_scope, 913
single_task, 914
trisycl::vendor::xilinx, 864
dataflow, 865
pipeline, 865
trisycl::vendor::xilinx::acap, 866
trisycl::vendor::xilinx::acap::aie, 866
trisycl::vendor::xilinx::acap::aie::accessor< Buffer >, 877
accessor, 879–883
base, 879
trisycl::vendor::xilinx::acap::aie::axi_packet, 71
axi_packet, 73, 74
data, 74
shutdown, 74
shutdown_request, 74
tlast, 74
value_type, 73
trisycl::vendor::xilinx::acap::aie::axi_packet::shutdown_t, 1272
trisycl::vendor::xilinx::acap::aie::axi_stream_switch, 45
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >
axi_ss_geo, 48
axi_stream_switch, 49
connect, 50, 51
data_type, 48
display, 52, 53
display_border, 54
display_routing_configuration, 55, 56
display_size, 57
get_as_router_minion, 57, 58
in_connection, 58, 59
input, 59, 60
input_ports, 65
mpl, 48, 49
out_connection, 60, 61
output, 62
output_ports, 65
spl, 49
start, 63, 64
stream_latency, 66
tikz_edge_angle, 64, 65
translate_port, 66
x_coordinate, 67
y_coordinate, 67
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::input_port, 1100
enabled, 1101
stream, 1101
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::output_port, 1146
stream, 1147
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1226
~router_minion, 1231, 1232
axi_ss, 1239
c, 1239
capacity, 1239
connect_to, 1233
futures, 1239
get_master_port_dests, 1233, 1234
join, 1234
mpl_outputs, 1240
outputs, 1240
read, 1235
router_minion, 1229, 1231
try_read, 1235, 1236
try_write, 1236, 1237
value_type, 1229
write, 1237, 1238
trisycl::vendor::xilinx::acap::aie::cascade_stream, 67
trisycl::vendor::xilinx::acap::aie::cascade_stream< Geography >
cascade_stream_pipes, 71
geo, 69
get_cascade_stream_in, 69, 70
get_cascade_stream_out, 70, 71
trisycl::vendor::xilinx::acap::aie::communicator_port, 916
~communicator_port, 917, 918
operator<<, 918
operator>>, 919
read, 920
try_read, 921
try_write, 921, 922
write, 922, 923
trisycl::vendor::xilinx::acap::aie::connection, 925
connection, 926, 927
in, 927
out, 927, 928
p, 928
trisycl::vendor::xilinx::acap::aie::connection::base, 895
base, 896, 897
p, 898
pipe_of, 897, 898
trisycl::vendor::xilinx::acap::aie::connection::input, 1096
base, 1097–1099

in, 1099
trisycl::vendor::xilinx::acap::aie::connection::output, 1142
base, 1143–1145
out, 1145
trisycl::vendor::xilinx::acap::aie::detail, 867
trisycl::vendor::xilinx::acap::aie::detail::device< Layout >, 965
cascade, 973
cass, 968
cmp, 968
connect, 974, 975
cs, 989
csp, 968, 969
device, 971, 972
display, 976, 977
fiber_executor, 989
for_each_tile, 977, 978
for_each_tile_index, 979
for_each_tile_neighborhood, 980, 981
for_each_tile_x_index, 981, 982
for_each_tile_y_index, 983
geo, 969
mem, 983, 984
sass, 969, 970
shim, 985, 986
shims, 989
smp, 970
ssp, 970, 971
ti, 990
tile, 987, 988
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure, 265
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >
axi_ss, 294
axi_ss_geo, 268
axi_ss_t, 268
connect, 272
dev, 294
display, 273, 274
display_core_size, 275
display_size, 276, 277
future_work, 294
geo, 268, 269
in, 277, 278
in_connection, 278, 279
input, 280, 281
mem, 282
mi, 295
mpl, 269
out, 282, 283
out_connection, 284
output, 285, 286
rx_dma, 287
single_task, 288, 289
spl, 269
tile_infrastructure, 269, 270
translate_input_port, 289, 290
translate_output_port, 290, 291
tx_dma, 291, 292
tx_dmas, 295
wait, 292
x, 292, 293
x_coordinate, 295
y, 293, 294
y_coordinate, 295
trisycl::vendor::xilinx::acap::aie::device< Layout >, 990
cascade, 998
cass, 994
cmp, 994
connect, 998, 999
csp, 994, 995
dd, 995
device, 997
display, 1000
facade_t, 995
for_each_tile, 1000, 1001
for_each_tile_index, 1001, 1002
for_each_tile_neighborhood, 1002
for_each_tile_x_index, 1003
for_each_tile_y_index, 1004
geo, 996
mem, 1004, 1005
program, 1005, 1006
queue, 1006, 1007
run, 1007–1009
sass, 996
shim, 1010
size, 1013
smp, 996, 997
ssp, 997
tile, 1011
uniform_run, 1012
x_max, 1013
x_min, 1013
x_size, 1014
y_max, 1014
y_min, 1014
y_size, 1014
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >, 1015
~dma, 1019, 1020
c, 1026
commit_command, 1021
data_mover, 1026
dma, 1018, 1019
dma_command, 1017, 1018
join, 1022
launch, 1023
nb_command, 1026
pop_command, 1023, 1024
push_command, 1024
wait, 1025
waiting_mutex, 1027
waiting_room, 1027

trisycl::vendor::xilinx::acap::aie::dma_dsl, 75
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >
dma, 80
dma_dsl, 76, 77
lck, 80
lock, 77, 78
receive, 78
send, 79
tile, 81
wait, 79, 80
trisycl::vendor::xilinx::acap::aie::fifo_channel, 1048
c, 1053
capacity, 1053
read, 1050
try_read, 1050, 1051
try_write, 1051, 1052
value_type, 1049, 1050
write, 1052
trisycl::vendor::xilinx::acap::aie::geography, 81
trisycl::vendor::xilinx::acap::aie::geography< Layout >
cascade_linear_id, 85
cascade_linear_x, 85, 86
cascade_linear_y, 87, 88
generate_tiles, 88, 89
is_cascade_end, 89, 90
is_cascade_start, 90
is_east_column, 91
is_memory_module, 92, 93
is_north_row, 94
is_south_row, 94, 95
is_valid_memory_module_offset, 95, 96
is_west_column, 96, 97
is_x_valid, 97, 98
is_x_y_valid, 98, 99
is_y_valid, 100, 101
layout, 84
linear_id, 101, 102
linear_x, 102, 103
linear_y, 104, 105
memory_module_dx_coordinate, 105, 106
memory_module_linear_id, 106, 107
size, 114
tile_coordinates, 114
validate_x, 108, 109
validate_x_y, 110, 111
validate_y, 112, 113
x_min, 114
x_size, 115
y_min, 115
y_size, 115
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 935
core_trace, 944–947
dma_0, 939–944, 946
dma_1, 939–944, 946
dma_last, 939–944, 946
east_0, 940, 942–947
east_1, 940, 942–947
east_2, 940, 942–947
east_3, 940–947
east_last, 940–947
fifo_0, 939–944, 946
fifo_1, 939–942, 944, 946
fifo_depth, 949
fifo_last, 939–942, 944, 946
interconnect, 949
is_axi_master, 947, 948
latency, 949
m_dma_range, 949
m_dma_size, 950
m_east_range, 950
m_east_size, 950
m_fifo_range, 951
m_fifo_size, 951
m_me_range, 951
m_me_size, 952
m_north_range, 952
m_north_size, 952
m_south_range, 953
m_south_size, 953
m_tile_ctrl_range, 953
m_tile_ctrl_size, 954
m_west_range, 954
m_west_size, 954
master_port_layout, 939, 941
me_0, 939–946
me_1, 939–944, 946
me_last, 939–944, 946
mem_trace, 944–947
nb_master_port, 955
nb_slave_port, 955
north_0, 940, 942, 944–947
north_1, 940, 942, 944–947
north_2, 940, 942, 944–947
north_3, 940, 942, 944–947
north_4, 940, 942
north_5, 940, 942
north_last, 940, 942, 944–947
s_dma_range, 955
s_dma_size, 956
s_east_range, 956
s_east_size, 956
s_fifo_range, 956
s_fifo_size, 957
s_me_range, 957
s_me_size, 957
s_north_range, 958
s_north_size, 958
s_south_range, 958
s_south_size, 959
s_tile_ctrl_range, 959
s_tile_ctrl_size, 959
s_west_range, 960
s_west_size, 960
size, 940–947

slave_port_layout, 943, 945
south_0, 939–942, 944, 946
south_1, 939, 940, 942, 944, 946
south_2, 939, 940, 942, 944, 946
south_3, 939, 940, 942, 944, 946
south_4, 944, 946, 947
south_5, 944, 946, 947
south_last, 940, 942, 944, 946, 947
tile_ctrl, 939–944, 946
tile_ctrl_last, 939–944, 946
west_0, 940, 942, 944, 946, 947
west_1, 940, 942, 944, 946, 947
west_2, 940, 942, 944, 946, 947
west_3, 940, 942, 944, 946, 947
west_last, 940, 942, 944, 946, 947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1251
east_0, 1255, 1257, 1259–1261
east_1, 1255, 1257, 1259–1261
east_2, 1255–1257, 1259, 1261
east_3, 1255–1257, 1259, 1261
east_last, 1255–1257, 1259, 1261
fifo_0, 1254–1261
fifo_1, 1254–1261
fifo_depth, 1263
fifo_last, 1254–1261
is_axi_master, 1262
latency, 1263
m_east_range, 1263
m_east_size, 1264
m_fifo_range, 1264
m_fifo_size, 1264
m_north_range, 1265
m_north_size, 1265
m_south_range, 1265
m_south_size, 1266
m_tile_ctrl_range, 1266
m_tile_ctrl_size, 1266
m_west_range, 1267
m_west_size, 1267
master_port_layout, 1254, 1256
nb_master_port, 1267
nb_slave_port, 1268
north_0, 1255, 1257–1261
north_1, 1255, 1257–1261
north_2, 1255, 1257, 1259–1261
north_3, 1255, 1257, 1259–1261
north_4, 1255, 1257
north_5, 1255, 1257
north_last, 1255, 1257, 1259–1261
s_east_range, 1268
s_east_size, 1268
s_fifo_range, 1269
s_fifo_size, 1269
s_north_range, 1269
s_north_size, 1270
s_south_range, 1270
s_south_size, 1270
s_tile_ctrl_range, 1271
s_tile_ctrl_size, 1271
s_west_range, 1271
s_west_size, 1272
shim_trace, 1259, 1261
size, 1255–1257, 1259, 1261
slave_port_layout, 1258, 1260
south_0, 1254–1261
south_1, 1255–1261
south_2, 1255–1261
south_3, 1255–1261
south_4, 1255–1261
south_5, 1255–1261
south_6, 1258–1261
south_7, 1258–1261
south_last, 1255–1261
tile_ctrl, 1254–1261
tile_ctrl_last, 1254–1261
west_0, 1255–1261
west_1, 1255, 1257–1261
west_2, 1255, 1257–1261
west_3, 1255, 1257–1261
west_last, 1255, 1257–1261
trisycl::vendor::xilinx::acap::aie::layout, 868
xc10s10, 868
xc10s80, 868
xcvc1902, 868
xcve1152, 869
trisycl::vendor::xilinx::acap::aie::layout::full, 115
is_noc_tile, 117
is_pl_tile, 117
x_max, 118
x_min, 118
y_max, 118
y_min, 118
trisycl::vendor::xilinx::acap::aie::layout::one_pe, 118
x_max, 120
y_max, 120
trisycl::vendor::xilinx::acap::aie::layout::size, 120
trisycl::vendor::xilinx::acap::aie::layout::size< X_size, Y_size >
x_max, 121
y_max, 121
trisycl::vendor::xilinx::acap::aie::layout::small, 121
x_max, 123
y_max, 123
trisycl::vendor::xilinx::acap::aie::layout::vc1902, 1301
trisycl::vendor::xilinx::acap::aie::layout::ve1152, 123
trisycl::vendor::xilinx::acap::aie::lock_unit, 124
lock, 125, 126
lock_id_t, 125
lock_number, 126
locks, 126
trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device, 1105
acquire, 1107
acquire_with_value, 1107, 1108
cv, 1109

m, 1109
release, 1108
release_with_value, 1108, 1109
value, 1110
value_t, 1106, 1107
trisycl::vendor::xilinx::acap::aie::memory, 127
trisycl::vendor::xilinx::acap::aie::memory< AIE, X, Y >
x, 128
y, 128
trisycl::vendor::xilinx::acap::aie::memory_base, 128
lock, 130
mi, 131
set_memory_infrastructure, 130
trisycl::vendor::xilinx::acap::aie::memory_infrastructure, 131
dev, 133
lock, 132, 133
memory_infrastructure, 132
memory_locking_unit, 133
trisycl::vendor::xilinx::acap::aie::port, 1189
trisycl::vendor::xilinx::acap::aie::port::shim, 1250
port, 1250
x, 1250
trisycl::vendor::xilinx::acap::aie::port::tile, 1293
port, 1294
x, 1294
y, 1294
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, 1190
axi_ss, 1196
c, 1197
capacity, 1197
component_name, 1197
port_receiver, 1192
read, 1193
try_read, 1193, 1194
try_write, 1194, 1195
value_type, 1192
write, 1195, 1196
trisycl::vendor::xilinx::acap::aie::program, 133
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >
aie_d, 152
cascade, 139
device, 136
for_each_memory_base, 140
for_each_tile_base, 141
geo, 137
kernel_outliner, 142
memory_module, 142–144
memory_modules, 152
memory_modules_bases, 152
program, 138
run, 145–147
tile, 148, 149
tile_bases, 152
tileable_memory, 137
tileable_tile, 137, 138
tiles, 153
wait, 151
trisycl::vendor::xilinx::acap::aie::queue, 153
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >
aie_d, 161
device, 155
geo, 155
layout, 155, 156
queue, 156
run, 156, 157
submit, 158
uniform_run, 159
wait, 160
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter, 1216
wait, 1217
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, 1218
axi_ss, 1226
dma_base, 1220
fifo, 1226
read, 1221
receive, 1222
receiving_dma, 1220, 1221
try_read, 1223, 1224
try_write, 1224
write, 1225
trisycl::vendor::xilinx::acap::aie::sending_dma< AXIStreamSwitch >, 1241
dma_base, 1242
output_port, 1244
send, 1243
sending_dma, 1242, 1243
trisycl::vendor::xilinx::acap::aie::shim_tile, 161
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >
axi_ss, 179
axi_ss_geo, 163
axi_ss_t, 164
bli_in, 166, 167
bli_in_connection, 167, 168
bli_out, 169, 170
bli_out_connection, 170, 171
connect, 172
future_work, 180
geo, 164
input, 173, 174
mpl, 164
output, 174, 175
shim_tile, 165
spl, 165
start, 176, 177
translate_input_port, 177, 178
translate_output_port, 178, 179
x_coordinate, 180
y_coordinate, 180
trisycl::vendor::xilinx::acap::aie::tile, 180

trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >
barrier, 187
cascade_linear_id, 188
geo, 186
get_cascade_stream_in, 188, 189
get_cascade_stream_out, 190, 191
get_id, 191, 192
horizontal_barrier, 192, 194
is_cascade_end, 195
is_cascade_start, 196
is_east_column, 196, 197
is_east_valid, 197, 198
is_memory_module, 198, 199
is_memory_module_east, 199, 200
is_memory_module_north, 200, 201
is_memory_module_south, 202
is_memory_module_west, 203, 204
is_north_east_valid, 204, 205
is_north_row, 205, 206
is_north_valid, 206
is_north_west_valid, 207
is_south_east_valid, 208, 209
is_south_row, 209, 210
is_south_valid, 210, 211
is_south_west_valid, 211, 212
is_west_column, 212, 213
is_west_valid, 213, 214
linear_id, 214, 215
mem, 215–218
mem_east, 219, 220
mem_north, 220, 221
mem_south, 221, 222
mem_t, 186
mem_west, 223, 224
memory_module_linear_id, 224, 225
rx_dma, 225, 227
tb, 186
tx_dma, 228
vertical_barrier, 229, 230
x, 231
y, 231
trisycl::vendor::xilinx::acap::aie::tile_base, 231
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >
cascade, 234, 235
geo, 234
in, 235, 236
in_connection, 236, 237
out, 238
out_connection, 238, 239
program, 243
run, 240
set_program, 240
set_tile_infrastructure, 240, 241
single_task, 241
ti, 243
wait, 242
trisycl::vendor::xilinx::acap::aie::tile_infrastructure, 243
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >
axi_ss_geo, 246, 247
axi_ss_t, 247
connect, 250
display, 250, 251
display_size, 251
dti, 247
facade_t, 247, 248
geo, 248
in, 252
in_connection, 253
input, 254
mem, 254, 255
mpl, 248
out, 255
out_connection, 257
output, 258
rx_dma, 258, 259
single_task, 259, 260
spl, 248, 249
submit, 261
tile_infrastructure, 249, 250
tx_dma, 261, 262
wait, 262, 263
x, 263
x_size, 263, 264
y, 264
y_size, 264
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >::cgh_t, 906
cgh_t, 907
single_task, 907
ti, 908
trisycl::vendor::xilinx::acap::debug, 869
trisycl::vendor::xilinx::acap::debug::bsp_checker, 298
trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >
check, 298, 299
vector_clock, 299
vector_clock_alloc, 299
trisycl::vendor::xilinx::graphics, 869
rgb, 869
trisycl::vendor::xilinx::graphics::application, 887
~application, 888
image_grid, 889
initialized, 894
is_done, 889, 890
is_done_barrier, 890
start, 890, 891
t, 895
update_tile_data_image, 892, 893
w, 895
wait, 894
trisycl::vendor::xilinx::graphics::frame_grid, 1054
close_action, 1058
close_button, 1059

done, 1059
frame_grid, 1055, 1056
frames, 1059
get_frame, 1057
grid, 1059
nx, 1060
ny, 1060
set_close_action, 1058
sw, 1060
trisycl::vendor::xilinx::graphics::image_grid, 1079
cv, 1092
dispatch_protection, 1092
dispatcher, 1093
done_barrier, 1093
done_has_been_sampled, 1093
done_snapshot, 1093
get_palette, 1084
image_grid, 1082, 1083
image_x, 1094
image_y, 1094
images, 1094
is_done_barrier, 1085
on_key_press_event, 1085, 1086
p, 1094
submit, 1087
update_tile_data_image, 1088–1091
work_to_dispatch, 1095
zoom, 1095
trisycl::vendor::xilinx::graphics::palette, 1147
clip, 1168
color_mapping, 1168
decrease_clip, 1151, 1152
decrease_frequency, 1152, 1153
decrease_phase, 1153, 1154
frequency_log2, 1169
gray, 1149
gray_palette, 1155, 1156
handle_key_press_event, 1156, 1157
increase_clip, 1159
increase_frequency, 1160, 1161
increase_phase, 1161, 1162
k, 1169
kind, 1149
palette, 1150
palettize, 1162, 1163
phase, 1169
rainbow, 1149
rainbow_palette, 1163, 1164
set, 1165
size, 1169
update, 1166, 1167
trisycl::vendor::xilinx::latex, 870
trisycl::vendor::xilinx::latex::context, 928
add, 931
clean_node, 931
context, 930
display, 932
file_name, 934
finalize, 932, 933
finalized, 934
max_size, 934
out, 934
scale, 933
scaling_factor, 934
size, 935
trisycl::vendor::xilinx::partition, 870
block, 871
complete, 871
cyclic, 871
none, 871
type, 870, 871
trisycl::vendor::xilinx::partition::block< ElmInEachPhyMem, PDim >, 899
ele_in_each_physical_mem, 900
partition_dim, 900
partition_type, 900
trisycl::vendor::xilinx::partition::complete< PDim >, 923
partition_dim, 925
partition_type, 925
trisycl::vendor::xilinx::partition::cyclic< PhyMemNum, PDim >, 961
partition_dim, 962
partition_type, 962
physical_mem_num, 962
trisycl::vendor::xilinx::partition::none, 1110
partition_type, 1111
trisycl::vendor::xilinx::partition_array, 419
trisycl::vendor::xilinx::partition_array< ValueType, Size, PartitionType >
array_size, 429
begin, 425, 426
element_type, 421, 422
elems, 430
end, 427
get_partition_type, 427, 428
operator[], 428, 429
partition_array, 422–425
partition_type, 430
size, 429
trisycl::views, 871
enum_type, 872
enum_underlying_type, 872
TRISYCL_BOOST_COMPUTE_NAME
opencl_types.hpp, 1780
TRISYCL_BOOST_OPERATOR_VECTOR_OP
Some helpers for the implementation, 661
TRISYCL_CL_LANGUAGE_VERSION
Manage default configuration and types, 670
TRISYCL_CONSTANT_AS
address_space.hpp, 1343
TRISYCL_DEBUG_ONLY
Manage default configuration and types, 671
TRISYCL_DECLARE_CL_TYPES
opencl_types.hpp, 1780
TRISYCL_DECLARE_S
trisycl::vec< DataType, 1 >, 789

trisycl::vec< DataType, 16 >, 812–814
trisycl::vec< DataType, 2 >, 792
trisycl::vec< DataType, 3 >, 796
trisycl::vec< DataType, 4 >, 803
trisycl::vec< DataType, 8 >, 808, 809
Vector types in SYCL, 814
TRISYCL_DEFAULT_QUEUE_SELECTOR
queue_tail.hpp, 1818
TRISYCL_DEFINE_DEVICE_HOST_INFO_TEMPLATE
host_device.hpp, 1693
TRISYCL_DEFINE_DEVICE_INFO_CL_TEMPLATE
opencl_device_tail.hpp, 1699
TRISYCL_DEFINE_DEVICE_INFO_TEMPLATE
opencl_device_tail.hpp, 1699
TRISYCL_DEFINE_TYPES
opencl_types.hpp, 1780
TRISYCL_DEFINE_VEC_TYPE
Vector types in SYCL, 815
TRISYCL_DEFINE_VEC_TYPE_SIZE
Vector types in SYCL, 815
TRISYCL_DEVICE_ONLY
Manage default configuration and types, 671
TRISYCL_DUMP
debug.hpp, 2212
TRISYCL_DUMP_T
debug.hpp, 2212
TRISYCL_GEN_SWIZ2
Vector types in SYCL, 815
TRISYCL_GEN_SWIZ3
Vector types in SYCL, 816
TRISYCL_GEN_SWIZ4
Vector types in SYCL, 816
TRISYCL_GENERIC_AS
address_space.hpp, 1343
TRISYCL_GLOBAL_AS
address_space.hpp, 1343
TRISYCL_H_DEFINE_TYPE
opencl_types.hpp, 1780
TRISYCL_INFO_PARAM_TRAITS
param_traits.hpp, 1749
Platforms, contexts, accessors and queues, 619, 620
trisycl::info, 861
TRISYCL_INFO_PARAM_TRAITS_ANY_T
param_traits.hpp, 1749
TRISYCL_INTERNAL_DUMP
debug.hpp, 2212
TRISYCL_IS_WRAPPER_TRAIT
opencl_types.hpp, 1781
TRISYCL_LOCAL_AS
address_space.hpp, 1343
TRISYCL_LOGICAL_OPERATOR_VECTOR_OP
Some helpers for the implementation, 661
TRISYCL_MAKE_BOOST_CIRCULARBUFFER_THREAD_SAFE
Manage default configuration and types, 671
TRISYCL_MATH_WRAP
math.hpp, 1765
trisycl, 839
TRISYCL_MATH_WRAP2
math.hpp, 1765
TRISYCL_MATH_WRAP2s
math.hpp, 1766
trisycl, 839
TRISYCL_MATH_WRAP3
math.hpp, 1766
TRISYCL_MATH_WRAP3s
math.hpp, 1766
TRISYCL_MATH_WRAP3ss
math.hpp, 1767
TRISYCL_OPENCL_ONLY
Manage default configuration and types, 671
Platforms, contexts, accessors and queues, 613
trisycl::info, 860
TRISYCL_OPERATOR_BASIC_TYPE_OP
Some helpers for the implementation, 661
TRISYCL_ParallelForKernel_RANGE
kernel.hpp, 1757
opencl_kernel.hpp, 1760
trisycl::detail::kernel, 577
trisycl::detail::opencl_kernel, 1134
TRISYCL_PRIVATE_AS
address_space.hpp, 1343
TRISYCL_PROPERTY_CREATE
property_list.hpp, 1811
trisycl::property_list, 1210
TRISYCL_PROPERTY_HAS_GET
property_list.hpp, 1811
TRISYCL_SCALAR_TYPES
opencl_types.hpp, 1781
TRISYCL_SIZED_NAME
opencl_types.hpp, 1781
TRISYCL_SYCL_NAMESPACE
global_config.hpp, 1424
sycl.hpp, 1325, 1327
TRISYCL_TYPE_ACTUAL_NAME
opencl_types.hpp, 1782
TRISYCL_TYPE_CL_NAME
opencl_types.hpp, 1782
TRISYCL_TYPE_NAME
opencl_types.hpp, 1782
TRISYCL_TYPEDEF_TYPE
opencl_types.hpp, 1782
TRISYCL_UNARY_OPERATOR_VECTOR_OP
Some helpers for the implementation, 662
TRISYCL_UNARY_POSTFIX_OPERATOR_VECTOR_OP
Some helpers for the implementation, 662
TRISYCL_UNARY_PREFIX_OPERATOR_VECTOR_OP
Some helpers for the implementation, 662
TRISYCL_UNIMPL
Some helpers for the implementation, 663
TRISYCL_WEAK_ATTRIB_PREFIX
global_config.hpp, 1425
TRISYCL_WEAK_ATTRIB_SUFFIX
global_config.hpp, 1425
Platforms, contexts, accessors and queues, 620
TRISYCL_WRAPPER_CLASS_2

opencl_types.hpp, 1782
TRISYCL_WRAPPER_CLASS_3
opencl_types.hpp, 1783
TRISYCL_WRAPPER_CLASS_4
opencl_types.hpp, 1783
TRISYCL_XILINX_AIE_FIBER_EXECUTOR_THREADS
config.hpp, 2222, 2225
TRISYCL_XILINX_AIE_TILE_CODE_ON_FIBER
config.hpp, 2222, 2225
try_read
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1235, 1236
trisycl::vendor::xilinx::acap::aie::communicator_port, 921
trisycl::vendor::xilinx::acap::aie::fifo_channel, 1050, 1051
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, 1193, 1194
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, 1223, 1224
try_write
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1236, 1237
trisycl::vendor::xilinx::acap::aie::communicator_port, 921, 922
trisycl::vendor::xilinx::acap::aie::fifo_channel, 1051, 1052
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, 1194, 1195
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, 1224
tuple_indexed_access
trisycl::mixin::accessor< T, Dimensions >, 521
tuple_indexed_mdspan_access
trisycl::mixin::accessor< T, Dimensions >, 522
tuple_to_array
Helpers to do array and tuple conversion, 649
tuple_to_array_iterate
Helpers to do array and tuple conversion, 649
tx_dma
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 291, 292
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 228
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 261, 262
tx_dmas
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 295
type
std::tuple_element< I, T >, 1298
std::tuple_element< I, trisycl::vec< T, Size > >, 1299
trisycl::detail::address_space_base< T, AS >, 637
trisycl::detail::ocl_type< T, access::address_space::constant_space >, 623
trisycl::detail::ocl_type< T, access::address_space::global_space >, 624
trisycl::detail::ocl_type< T, access::address_space::local_space >, 624
trisycl::detail::ocl_type< T, access::address_space::private_space >, 625
trisycl::detail::ocl_type< T, AS >, 623
trisycl::device, 546
trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, StorageType >, 1274
trisycl::vendor::trisycl::scope::detail::storage_type_trait< ScopedType, typename ScopedType::storage_type >, 1275
trisycl::vendor::xilinx::partition, 870, 871
type_pretty_name
trisycl::detail::debug< T >, 667

uchar
opencl_spir.h, 1498
UCHAR_MAX
opencl_spir.h, 1486
uint
opencl_spir.h, 1498
UINT_MAX
opencl_spir.h, 1486
uintptr_t
opencl_spir.h, 1498
ulong
opencl_spir.h, 1498
ULONG_MAX
opencl_spir.h, 1486
underlying_value
trisycl::detail, 854
uniform_run
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1012
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >, 159
unimplemented
Some helpers for the implementation, 665
unique_ptr_class
trisycl, 832
update
trisycl::mixin::accessor< T, Dimensions >, 523
trisycl::vendor::xilinx::graphics::palette, 1166, 1167
update_buffer_state
trisycl::detail::buffer_base, 363
update_tile_data_image
trisycl::vendor::xilinx::graphics::application, 892, 893
trisycl::vendor::xilinx::graphics::image_grid, 1088–1091
use
trisycl::detail::buffer_base, 365
use_count
trisycl::buffer< T, Dimensions, Allocator >, 336
use_native_work_item
trisycl::detail, 854
used_for_reading

trisycl::detail::sycl_2_2::pipe< T >, 464
used_for_writing
trisycl::detail::sycl_2_2::pipe< T >, 464
ushort
opencl_spir.h, 1498
USHRT_MAX
opencl_spir.h, 1486

val
opencl_spir.h, 1515
validate_x
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 108, 109
validate_x_y
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 110, 111
validate_y
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 112, 113
value
trisycl::detail::alignment< T >, 885
trisycl::detail::alignment<::trisycl::vec< DataType, NumElements > >, 886
trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device, 1110
value_t
trisycl::vendor::xilinx::acap::aie::lock_unit::locking_device, 1106, 1107
value_type
trisycl::buffer< T, Dimensions, Allocator >, 326
trisycl::detail::cache< Key, Value >, 902
trisycl::detail::container_element_aspect< T >, 651
trisycl::detail::sycl_2_2::pipe< T >, 449
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 467
trisycl::detail::sycl_2_2::pipe_reservation< PipeAccessor >, 487
trisycl::mixin::accessor< T, Dimensions >, 516
trisycl::pipe< Name, T, MinCapacity, SizeT >, 371
trisycl::sycl_2_2::pipe< T >, 442
trisycl::sycl_2_2::pipe_reservation< PipeAccessor >, 480
trisycl::sycl_2_2::static_pipe< T, Capacity >, 496
trisycl::vendor::trisycl::random::xorshift< Nbits >, 1318
trisycl::vendor::xilinx::acap::aie::axi_packet, 73
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1229
trisycl::vendor::xilinx::acap::aie::fifo_channel, 1049, 1050
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, 1192
variable
trisycl::detail::address_space_variable< T, AS >, 640
vec
trisycl::detail::vec< DataType, NumElements >, 1305
Vector types in SYCL, 784
__swizzled_vec__, 817
a, 817
b, 817
g, 817
r, 817
s0, 817
s1, 818
s2, 818
s3, 818
s4, 818
s5, 819
s6, 819
s7, 819
s8, 819
s9, 819
sA, 820
sB, 820
sC, 820
sD, 820
sE, 820
sF, 821
TRISYCL_DECLARE_S, 814
TRISYCL_DEFINE_VEC_TYPE, 815
TRISYCL_DEFINE_VEC_TYPE_SIZE, 815
TRISYCL_GEN_SWIZ2, 815
TRISYCL_GEN_SWIZ3, 816
TRISYCL_GEN_SWIZ4, 816
w, 821
x, 821
y, 821
z, 821
vector_class
trisycl, 832
vector_clock
trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >, 299
vector_clock_alloc
trisycl::vendor::xilinx::acap::debug::bsp_checker< Geography >, 299
vendor
Platforms, contexts, accessors and queues, 608
vendor_id
Platforms, contexts, accessors and queues, 607
version
Platforms, contexts, accessors and queues, 608
vertical_barrier
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 229, 230

w
trisycl::vec< DataType, 4 >, 803
trisycl::vendor::xilinx::graphics::application, 895
Vector types in SYCL, 821
w_rid_q
trisycl::detail::sycl_2_2::pipe< T >, 465
wait
trisycl::detail::buffer_base, 365
trisycl::detail::event, 1031

trisycl::detail::host_event, 1075
trisycl::detail::opencl_event, 1130
trisycl::detail::task, 1289
trisycl::event, 1037
trisycl::queue, 602
trisycl::vendor::trisycl::scope::queue< Device, QueueStorage >, 412
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 292
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >, 1025
trisycl::vendor::xilinx::acap::aie::dma_dsl< Tile, DMA, LockRef >, 79, 80
trisycl::vendor::xilinx::acap::aie::program< AIEDevice, Tile, Memory >, 151
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >, 160
trisycl::vendor::xilinx::acap::aie::queue< AIEDevice >::queue_waiter, 1217
trisycl::vendor::xilinx::acap::aie::tile_base< AIE_Program >, 242
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 262, 263
trisycl::vendor::xilinx::graphics::application, 894
wait_and_throw
trisycl::event, 1037
trisycl::queue, 603
wait_for_kernel_execution
trisycl::detail::queue, 1214
wait_for_producers
trisycl::detail::task, 1289
waiter
Data access and storage in SYCL, 377
waiting_mutex
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >, 1027
waiting_room
trisycl::vendor::xilinx::acap::aie::dma< AXIStreamSwitch, SpecializedDMA >, 1027
weak_ptr_class
trisycl, 832
west_0
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942, 944, 946, 947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255–1261
west_1
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942, 944, 946, 947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255, 1257–1261
west_2
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942, 944, 946, 947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255, 1257–1261
west_3
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942, 944, 946, 947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255, 1257–1261
west_last
trisycl::vendor::xilinx::acap::aie::geography< Layout >::core_axi_stream_switch, 940, 942, 944, 946, 947
trisycl::vendor::xilinx::acap::aie::geography< Layout >::shim_axi_stream_switch, 1255, 1257–1261
what
trisycl::exception, 687
work_stealing
trisycl::detail::fiber_pool, 1040
work_to_dispatch
trisycl::vendor::xilinx::graphics::image_grid, 1095
working_threads
trisycl::detail::fiber_pool, 1047
write
trisycl::access, 842
trisycl::detail::sycl_2_2::pipe< T >, 462
trisycl::detail::sycl_2_2::pipe_accessor< T, AccessMode, Target >, 474
trisycl::pipe< Name, T, MinCapacity, SizeT >, 373
trisycl::vendor::trisycl::pipe::cout, 304, 305
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >::router_minion, 1237, 1238
trisycl::vendor::xilinx::acap::aie::communicator_port, 922, 923
trisycl::vendor::xilinx::acap::aie::fifo_channel, 1052
trisycl::vendor::xilinx::acap::aie::port_receiver< AXIStreamSwich >, 1195, 1196
trisycl::vendor::xilinx::acap::aie::receiving_dma< AXIStreamSwitch >, 1225
write_done
trisycl::detail::sycl_2_2::pipe< T >, 465
write_only
Platforms, contexts, accessors and queues, 611

x
opencl_spir.h, 1515
trisycl::vec< DataType, 1 >, 789
trisycl::vec< DataType, 2 >, 792
trisycl::vec< DataType, 3 >, 796, 797
trisycl::vec< DataType, 4 >, 803, 804
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 292, 293
trisycl::vendor::xilinx::acap::aie::memory< AIE, X, Y >, 128
trisycl::vendor::xilinx::acap::aie::port::shim, 1250
trisycl::vendor::xilinx::acap::aie::port::tile, 1294

trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 231
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 263
Vector types in SYCL, 821
x_coordinate
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 67
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 295
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 180
x_max
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1013
trisycl::vendor::xilinx::acap::aie::layout::full, 118
trisycl::vendor::xilinx::acap::aie::layout::one_pe, 120
trisycl::vendor::xilinx::acap::aie::layout::size< X_size, Y_size >, 121
trisycl::vendor::xilinx::acap::aie::layout::small, 123
x_min
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1013
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 114
trisycl::vendor::xilinx::acap::aie::layout::full, 118
x_size
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1014
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 115
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 263, 264
xc10s10
trisycl::vendor::xilinx::acap::aie::layout, 868
xc10s80
trisycl::vendor::xilinx::acap::aie::layout, 868
xcvc1902
trisycl::vendor::xilinx::acap::aie::layout, 868
xcve1152
trisycl::vendor::xilinx::acap::aie::layout, 869
Xilinx vendor extensions, 418
_ssdm_SpecArrayPartition, 431
_ssdm_op_SpecDataflowPipeline, 430
_ssdm_op_SpecPipeline, 431
xorshift
trisycl::vendor::trisycl::random::xorshift< Nbits >, 1318, 1319

y
opencl_spir.h, 1515
trisycl, 840
trisycl::vec< DataType, 2 >, 793
trisycl::vec< DataType, 3 >, 797
trisycl::vec< DataType, 4 >, 804
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 293, 294
trisycl::vendor::xilinx::acap::aie::memory< AIE, X, Y >, 128
trisycl::vendor::xilinx::acap::aie::port::tile, 1294
trisycl::vendor::xilinx::acap::aie::tile< AIE_Program, X, Y >, 231
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 264
Vector types in SYCL, 821
y_coordinate
trisycl::vendor::xilinx::acap::aie::axi_stream_switch< AXIStreamGeography >, 67
trisycl::vendor::xilinx::acap::aie::detail::tile_infrastructure< Geography >, 295
trisycl::vendor::xilinx::acap::aie::shim_tile< AIEDevice >, 180
y_max
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1014
trisycl::vendor::xilinx::acap::aie::layout::full, 118
trisycl::vendor::xilinx::acap::aie::layout::one_pe, 120
trisycl::vendor::xilinx::acap::aie::layout::size< X_size, Y_size >, 121
trisycl::vendor::xilinx::acap::aie::layout::small, 123
y_min
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1014
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 115
trisycl::vendor::xilinx::acap::aie::layout::full, 118
y_size
trisycl::vendor::xilinx::acap::aie::device< Layout >, 1014
trisycl::vendor::xilinx::acap::aie::geography< Layout >, 115
trisycl::vendor::xilinx::acap::aie::tile_infrastructure< Geography >, 264

z
opencl_spir.h, 1516
trisycl, 840
trisycl::vec< DataType, 3 >, 797
trisycl::vec< DataType, 4 >, 804
Vector types in SYCL, 821
zip
trisycl::detail::vec< DataType, NumElements >, 1314
zoom
trisycl::vendor::xilinx::graphics::image_grid, 1095