AMBA-PV Extensions to TLM

Version 2.0

Reference Guide

Chapter 1

AMBA-PV Extensions to TLM 2.0 Reference Guide

About this document

This document provides a reference for the classes and interfaces included in the *AMBA-PV Extensions to TLM 2.0* (AMBA-PV). These classes and interfaces provide a Programmer's View (PV) of the AMBA buses.

Intended audience

This document is written for experienced hardware and software developers to aid the development of TLM 2.0 compatible models that communicate over AMBA buses.

You must be familiar with:

- The basic concepts of C++ such as classes and inheritance
- SystemC and TLM standards

Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available on Arm Developer. Each document link in the following lists goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm® product resources

The following Non-Confidential publications provide reference information about AMBA and the Arm architecture:

- AMBA AXI Protocol Specification (ARM IHI0022)
- AMBA AHB-Lite Protocol Specification (ARM IHI0033)
- AMBA APB Protocol Specification (ARM IHI0024)
- AMBA Specification (ARM IHI0011)
- Arm Architecture Reference Manual (ARM DDI0487).

The following Non-Confidential publications provide information about related Arm products and toolkits:

- AMBA-PV Extensions to TLM User Guide (100962).

Non-Arm® resources

This section lists relevant documents published by third parties.

See http://www.accellera.org for further information on the Accellera Systems Initiative.

The following publications provide reference information about SystemC and TLM standards:

- IEEE Std 1666-2011, *SystemC Language Reference Manual*, 9 January 2012.

Note Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader. Adobe PDF reader products can be downloaded at http://www.adobe.com.

Release information

Table 1.1 Document history

Issue	Date	Confidentiality	Change
0200-00	13 Mar 2024	Non-Confidential	First release. Supersedes DUI0847.
0200-01	19 Jun 2024	Non-Confidential	Update for Fast Models 11.26.
0200-02	16 Sep 2024	Non-Confidential	Update for Fast Models 11.27.

Proprietary Notice

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

Arm Limited. Company 02557590 registered in England.
110 Fulbourn Road, Cambridge, England CB1 9NJ.
PRE-1121-V1.0

Confidentiality Status

This document is Non-Confidential.

Product Status

All products and Services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

The information in this document is Final, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on Arm Support.

To provide feedback on the document, fill the following survey: https://developer.arm.com/feedback/survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

This document includes language that can be offensive. We will replace this language in a future issue of this document.
To report offensive language in this document, email terms@arm.com.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 amba_pv Namespace Reference

AMBA-PV namespace.

Namespaces

- namespace atomic_subop_impl

  AMBA-PV atomic operation type implementation namespace.
- namespace ext

  Extensions namespace.

Data Structures

- class amba_pv_ace_bw_transport_if

  AMBA-PV core additional transaction interface for ACE.
- class amba_pv_ace_master_base

  Base class for all AMBA-PV ACE master modules.
- class amba_pv_ace_master_socket

  AMBA-PV ACE socket to be instantiated on the master side.
- class amba_pv_ace_protocol_checker

  AMBA-PV ACE protocol checker model.
- class amba_pv_ace_simple_probe

  AMBA-PV ACE simple probe model.
- class amba_pv_ace_slave_socket

  AMBA-PV ACE socket to be instantiated on the slave side.
- class amba_pv_address_map

  AMBA-PV address mapping information structure.
- class amba_pv_address_region

  AMBA-PV address region structure.
- class amba_pv_atomic

  Provides atomic transaction information used by AMBA AXI buses.
- class amba_pv_atomic_utils

  An utility class that offers the implementation of executing an atomic transaction.
- class amba_pv_attributes

  Provides support for additional user-defined attributes.
- class amba_pv_bw_snoop_if

  AMBA-PV core additional transaction interface for ACE.
- class amba_pv_bw_transport_if

  AMBA-PV core transaction interface.

- class amba_pv_control

  Provides support for additional control information used by the AMBA buses.
- class amba_pv_decoder

  AMBA-PV bus decoder model.
- class amba_pv_dvm

  Provides DVM message information used by the AMBA ACE buses.
- class amba_pv_exclusive_monitor

  AMBA-PV exclusive monitor model.
- class amba_pv_extension

  AMBA-PV extension class.
- class amba_pv_from_tlm_bridge

  TLM 2.0 BP to AMBA-PV bridge module.
- class amba_pv_fw_transport_if

  AMBA-PV core transaction interface.
- class amba_pv_heap_allocator

  AMBA-PV heap memory allocator.
- class amba_pv_if

  AMBA-PV user-layer transaction interface.
- class amba_pv_master_base

  Base class for all AMBA-PV master modules.
- class amba_pv_master_socket

  AMBA-PV socket to be instantiated on the master side.
- class amba_pv_memory

  AMBA-PV advanced memory model.
- class amba_pv_memory_base

  AMBA-PV memory model base class.
- class amba_pv_protocol_checker

  AMBA-PV protocol checker model.
- class amba_pv_protocol_checker_base

  AMBA-PV protocol checker base model.
- struct amba_pv_protocol_types

  AMBA-PV protocol types.
- class amba_pv_response

  AMBA-PV response class.
- class amba_pv_simple_memory

  AMBA-PV simple memory model.
- class amba_pv_simple_probe

  AMBA-PV simple probe model.
- class amba_pv_simple_probe_base

  AMBA-PV simple probe base model.
- class amba_pv_slave_base

  Base class for all AMBA-PV slave modules.
- class amba_pv_slave_socket

  AMBA-PV socket to be instantiated on the slave side.
- class amba_pv_snoop_socket

  AMBA-PV slave socket used to implement the upstream ACE snoop interface.
- class amba_pv_socket_array

  AMBA-PV socket array class.
- class amba_pv_socket_base

  AMBA-PV socket base class.
- class amba_pv_to_tlm_bridge

AMBA-PV to TLM 2.0 BP bridge module.

- class amba_pv_trans_lock

  AMBA-PV transaction lock wrapper.
- class amba_pv_trans_pool

  AMBA-PV transaction pool.
- class amba_pv_trans_ptr

  AMBA-PV transaction smart pointer.
- class nonblocking_transport_if

  Non-blocking transport core interface.
- class signal_export_base

  Signal export base class.
- class signal_from_sc_bridge

  Generic sc_signal to Signal bridge module.
- class signal_if

  Signal interface.
- class signal_master_port

  Signal port to be instantiated on the master side.
- class signal_request

  Signal request type.
- class signal_response

  Signal response type.
- class signal_slave_base

  Base class for all Signal slave modules.
- class signal_slave_export

  Signal export to be instantiated on the slave side.
- class signal_state_from_sc_bridge

  Generic sc_signal to SignalState bridge module.
- class signal_state_if

  SignalState interface.
- class signal_state_master_port

  SignalState port to be instantiated on the master side.
- class signal_state_slave_base

  Base class for all SignalState slave modules.
- class signal_state_slave_export

  SignalState export to be instantiated on the slave side.
- class signal_state_to_sc_bridge

  Generic SignalState to sc_signal bridge module.
- class signal_state_transport_if

  SignalState core interface.
- class signal_to_sc_bridge

  Generic Signal to sc_signal bridge module.
- class signal_transport_if

  Signal core interface.

Typedefs

- typedef amba_pv_protocol_types::tlm_payload_type amba_pv_transaction

  AMBA-PV transaction type.

Enumerations

- enum amba_pv_atomic_op_t {
  AMBA_PV_NONATOMIC ,
  AMBA_PV_ATOMICSTORE ,
  AMBA_PV_ATOMICLOAD ,
  AMBA_PV_ATOMICSWAP ,
  AMBA_PV_ATOMICCOMPARE }

  Atomic transaction type.
- enum amba_pv_atomic_subop_t {
  AMBA_PV_ATOMIC_ADD ,
  AMBA_PV_ATOMIC_BIT_CLEAR ,
  AMBA_PV_ATOMIC_EXCLUSIVE_OR ,
  AMBA_PV_ATOMIC_BIT_SET ,
  AMBA_PV_ATOMIC_SIGNED_MAX ,
  AMBA_PV_ATOMIC_SIGNED_MIN ,
  AMBA_PV_ATOMIC_UNSIGNED_MAX ,
  AMBA_PV_ATOMIC_UNSIGNED_MIN }

  Atomic transaction operation type.
- enum amba_pv_atomic_endianness_t {
  AMBA_PV_LITTLE_ENDIAN ,
  AMBA_PV_BIG_ENDIAN }

  Atomic operation endianness.
- enum amba_pv_snoop_t {
  AMBA_PV_READ_NO_SNOOP ,
  AMBA_PV_READ_ONCE ,
  AMBA_PV_READ_CLEAN ,
  AMBA_PV_READ_NOT_SHARED_DIRTY ,
  AMBA_PV_READ_SHARED ,
  AMBA_PV_READ_UNIQUE ,
  AMBA_PV_CLEAN_UNIQUE ,
  AMBA_PV_CLEAN_SHARED ,
  AMBA_PV_CLEAN_INVALID ,
  AMBA_PV_MAKE_UNIQUE ,
  AMBA_PV_MAKE_INVALID ,
  AMBA_PV_WRITE_NO_SNOOP ,
  AMBA_PV_WRITE_UNIQUE ,
  AMBA_PV_WRITE_LINE_UNIQUE ,
  AMBA_PV_WRITE_BACK ,
  AMBA_PV_WRITE_CLEAN ,
  AMBA_PV_EVICT ,
  AMBA_PV_BARRIER ,
  AMBA_PV_DVM_COMPLETE ,
  AMBA_PV_DVM_MESSAGE }

  Snoop type.
- enum amba_pv_domain_t {
  AMBA_PV_NON_SHAREABLE ,
  AMBA_PV_INNER_SHAREABLE ,
  AMBA_PV_OUTER_SHAREABLE ,
  AMBA_PV_SYSTEM }

  Domain type.
- enum amba_pv_bar_t {
  AMBA_PV_RESPECT_BARRIER ,
  AMBA_PV_MEMORY_BARRIER ,
  AMBA_PV_IGNORE_BARRIER ,
  AMBA_PV_SYNCHRONISATION_BARRIER }

  Barrier type.

- enum amba_pv_service_req_t {
  AMBA_PV_NO_SERVICE_REQUEST ,
  AMBA_PV_PCIE_SERVICE ,
  AMBA_PV_FAR_ATOMIC_SERVICE }

  Service Request type.
- enum amba_pv_physical_address_space_t {
  AMBA_PV_SECURE_PAS ,
  AMBA_PV_NON_SECURE_PAS ,
  AMBA_PV_ROOT_PAS ,
  AMBA_PV_REALM_PAS ,
  AMBA_PV_SYSTEM_AGENT_PAS ,
  AMBA_PV_NON_SECURE_PROTECTED_PAS ,
  AMBA_PV_NA6_PAS ,
  AMBA_PV_NA7_PAS }

  Physical Address Space type.
- enum amba_pv_mmuflow_t {
  AMBA_PV_MMUFLOW_STALL ,
  AMBA_PV_MMUFLOW_ATST ,
  AMBA_PV_MMUFLOW_NoStall ,
  AMBA_PV_MMUFLOW_PRI }

  AxMMUFLOW encodings.
- enum amba_pv_dvm_message_t {
  AMBA_PV_TLB_INVALIDATE ,
  AMBA_PV_BRANCH_PREDICTOR_INVALIDATE ,
  AMBA_PV_PHYSICAL_INSTRUCTION_CACHE_INVALIDATE ,
  AMBA_PV_VIRTUAL_INSTRUCTION_CACHE_INVALIDATE ,
  AMBA_PV_SYNC ,
  AMBA_PV_HINT }

  DVM Message type.
- enum amba_pv_dvm_os_t {
  AMBA_PV_HYPERVISOR_OR_GUEST ,
  AMBA_PV_EL3 ,
  AMBA_PV_GUEST ,
  AMBA_PV_HYPERVISOR }

  DVM message Guest OS or hypervisor type.
- enum amba_pv_dvm_security_t {
  AMBA_PV_SECURE_AND_NON_SECURE ,
  AMBA_PV_SECURE_ONLY ,
  AMBA_PV_NON_SECURE_ONLY }

  DVM message security type.
- enum amba_pv_dvm_stage_t {
  AMBA_PV_DVM_V7 ,
  AMBA_PV_STAGE_1_ONLY ,
  AMBA_PV_STAGE_2_ONLY }

  DVM message Staged Invalidation.
- enum amba_pv_burst_t {
  AMBA_PV_FIXED ,
  AMBA_PV_INCR ,
  AMBA_PV_WRAP }

  Burst type.
- enum amba_pv_resp_t {
  AMBA_PV_INCOMPLETE ,
  AMBA_PV_OKAY ,
  AMBA_PV_EXOKAY ,
  AMBA_PV_SLVERR ,
  AMBA_PV_DECERR }

AMBA-PV response type.

- enum amba_pv_protocol_t {
  AMBA_PV_APB ,
  AMBA_PV_AHB ,
  AMBA_PV_AXI ,
  AMBA_PV_AXI3 ,
  AMBA_PV_AXI4_LITE ,
  AMBA_PV_AXI4 ,
  AMBA_PV_ACE_LITE ,
  AMBA_PV_ACE ,
  AMBA_PV_AXI5 }

  AMBA procotol checks type.
- enum signal_command {
  SIGNAL_SET ,
  SIGNAL_GET }

  Signal request command type.

Functions

- std::string amba_pv_atomic_op_string (const amba_pv_atomic_op_t op)

  converts an atomic op enum to a human readable string
- void swap_bytes (unsigned char *const data, const size_t size)

  Swaps the bytes on a block of memory based on a specified size.
- std::string amba_pv_snoop_read_string (amba_pv_snoop_t snoop, amba_pv_domain_t domain, amba_pv_bar_t bar)

  Returns the text string representation of the specified read snoop type.
- std::string amba_pv_snoop_write_string (amba_pv_snoop_t snoop, amba_pv_domain_t domain, amba_pv_bar_t bar)

  Returns the text string representation of the specified snoop type for write transactions.
- std::string amba_pv_domain_string (amba_pv_domain_t domain)

  Returns the text string representation of the specified domain type.
- std::string amba_pv_bar_string (amba_pv_bar_t bar)

  Returns the text string representation of the specified bar type.
- std::string amba_pv_dvm_message_string (amba_pv_dvm_message_t message_type)

  Returns the text string representation of the specified DVM message type.
- std::string amba_pv_dvm_os_string (amba_pv_dvm_os_t os)

  Returns the text string representation of the specified DVM Guest OS or hypervisor type.
- std::string amba_pv_dvm_security_string (amba_pv_dvm_security_t security)

  Returns the text string representation of the specified DVM security type.
- std::string amba_pv_burst_string (amba_pv_burst_t burst)

  Returns the text string representation of the specified burst type.
- std::string amba_pv_snoop_string (amba_pv_snoop_t snoop)

  Returns the text string representation of the specified snoop type.
- sc_dt::uint64 amba_pv_address (const sc_dt::uint64 &addr, unsigned int length, unsigned int size, amba_pv_burst_t burst, unsigned int n)

  Computes the address of a transfer in a burst.
- std::string amba_pv_resp_string (amba_pv_resp_t resp)

  Returns the text string representation of the specified AMBA-PV response.
- amba_pv_resp_t amba_pv_resp_from_tlm (tlm::tlm_response_status response_status, bool is_↩ exclusive=false)

  Translates the specified TLM 2.0 response status value into an AMBA-PV response.
- tlm::tlm_response_status amba_pv_resp_to_tlm (amba_pv_resp_t resp, bool is_exclusive=false)

  Translates the specified AMBA-PV response value into a TLM 2.0 response status.

6.1.1 Detailed Description

AMBA-PV namespace.
Namespace amba_pv contains all the AMBA-PV classes.

6.1.2 Typedef Documentation

6.1.2.1 amba_pv_transaction

```
typedef amba_pv_protocol_types::tlm_payload_type amba_pv::amba_pv_transaction
```

AMBA-PV transaction type.
The amba_pv_transaction type is equivalent to the tlm_payload_type of the amba_pv_protocol_types struct.

6.1.3 Enumeration Type Documentation

6.1.3.1 amba_pv_atomic_op_t

```
enum amba_pv::amba_pv_atomic_op_t
```

Atomic transaction type.
The atomic transaction type indicates the type of an atomic transaction. For AtomicStore and AtomicLoad, their signals also incorporate endianness and atomic operation type, which are independently indicated by amba_pv_↩ atomic_subop_t and amba_pv_atomic_endianness_t.
The bit representation of this type partially matches the AXI AWATOP[5:0] AMBA5 signals. For non-atomic operation, AtomicSwap and AtomicCompare, the bits fully represent the signals; for AtomicStore and AtomicLoad, the bits only represent the AWATOP[5:4] signals.

Note

AXI AWATOP[5:0] signals:

- 0b000000 Non-atomic operation
- 0b01exxx AtomicStore
- 0b10exxx AtomicLoad
- 0b110000 AtomicSwap
- 0b110001 AtomicCompare e indicates the endianess, and xxx indicates the atomic operation type.

See also

amba_pv_atomic, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t

Enumerator

AMBA_PV_NONATOMIC	Non-atomic operation.
AMBA_PV_ATOMICSTORE	AtomicStore.
AMBA_PV_ATOMICLOAD	AtomicLoad.
AMBA_PV_ATOMICSWAP	AtomicSwap.
AMBA_PV_ATOMICCOMPARE	AtomicCompare.

6.1.3.2 amba_pv_atomic_subop_t

```
enum amba_pv::amba_pv_atomic_subop_t
```

Atomic transaction operation type.
The atomic transaction type indicates the operation type of an atomic transaction.
The bit representation of this type matches the AXI AWATOP[2:0] AMBA5 signals.

Note

Only AtomicStore and AtomicLoad transaction can execute an atomic operation.

See also

amba_pv_atomic, amba_pv_atomic_op_t, amba_pv_atomic_endianness_t

Enumerator

AMBA_PV_ATOMIC_ADD	Addition.
AMBA_PV_ATOMIC_BIT_CLEAR	Clean memory bits with the set bits in the sent data
AMBA_PV_ATOMIC_EXCLUSIVE_OR	Exclusive OR.
AMBA_PV_ATOMIC_BIT_SET	Set memory bits with the set bits in the sent data.
AMBA_PV_ATOMIC_SIGNED_MAX	Maximum of memory value and sent data (assume signed data)
AMBA_PV_ATOMIC_SIGNED_MIN	Minimum of memory value and sent data (assume signed data)
AMBA_PV_ATOMIC_UNSIGNED_MAX	Maximum of memory value and sent data (assume unsigned data)
AMBA_PV_ATOMIC_UNSIGNED_MIN	Minimum of memory value and sent data (assume unsigned data)

6.1.3.3 amba_pv_atomic_endianness_t

`enum amba_pv::amba_pv_atomic_endianness_t`
Atomic operation endianness.
The atomic operation endianness indicates the endianness of an atomic operation.
The bit representation of this type matches the AWATOP[3] AMBA5 signals.

Note

Only AtomicStore and AtomicLoad transaction with non-bitwise operations support big endian.

See also

amba_pv_atomic, amba_pv_atomic_op_t, amba_pv_atomic_subop_t

Enumerator

AMBA_PV_LITTLE_ENDIAN	little-endian operation
AMBA_PV_BIG_ENDIAN	big-endian operation

6.1.3.4 amba_pv_snoop_t

`enum amba_pv::amba_pv_snoop_t`
Snoop type.
The snoop type, together with the barrier and domain information, determines the transaction type for the extended transactions on coherent buses.
The bit representation of this type matches the AxSNOOP AMBA4 signals.

See also

amba_pv_control, amba_pv_domain_t, amba_pv_bar_t

Enumerator

AMBA_PV_READ_NO_SNOOP	Read transaction for non-shareable memory.
AMBA_PV_READ_ONCE	Read transaction for shareable memory, when local caching is not required.
AMBA_PV_READ_CLEAN	Read transaction for shareable memory, that requires a clean copy of a cache line.
AMBA_PV_READ_NOT_SHARED_DIRTY	Read transaction for shareable memory, can accept cache line in any state except SharedDirty.
AMBA_PV_READ_SHARED	Read transaction for shareable memory, can accept cache line in any state.
AMBA_PV_READ_UNIQUE	Read transaction for shareable memory, ensures that cache line is held in Unique state.
AMBA_PV_CLEAN_UNIQUE	Cache clean operation, ensures cache line is held in Unique state.
AMBA_PV_CLEAN_SHARED	Broadcast cache clean operation.
AMBA_PV_CLEAN_INVALID	Broadcast cache clean and invalidate operation.
AMBA_PV_MAKE_UNIQUE	Cache invalidate operation, ensures cache line is held in a Unique state.
AMBA_PV_MAKE_INVALID	Broadcast cache invalidate operation.
AMBA_PV_WRITE_NO_SNOOP	Write transaction for non-shareable memory.
AMBA_PV_WRITE_UNIQUE	Write transaction for shareable memory that will must propogate to main memory.
AMBA_PV_WRITE_LINE_UNIQUE	Shareable write transaction that must propogate to main memory. A full cache line store of all bytes within the cache line must be updated.
AMBA_PV_WRITE_BACK	Write transaction can be used in shareable and non-shareable regions of memory and is a write of a dirty cache line to update main memory. For a shareable region of memory the cache line is no longer allocated.
AMBA_PV_WRITE_CLEAN	Write transaction can be used in shareable and non-shareable regions of memory and is a write of a dirty cache line to update main memory. For a shareable region of memory the cache line remains allocated.
AMBA_PV_EVICT	Indicates that a cache line has been evicted from a master's local cache. Must only be used in a shareable memory region and only used by a master that supports a snoop filer.
AMBA_PV_BARRIER	ACE barrier transactions.
AMBA_PV_DVM_COMPLETE	DVM complete transaction.
AMBA_PV_DVM_MESSAGE	DVM operation or DVM sync transactions.

6.1.3.5 amba_pv_domain_t

`enum amba_pv::amba_pv_domain_t`

Domain type.

The domain type indicates the level of shareablility.

The bit representation of this type matches the AxDOMAIN AMBA4 signals.

See also

amba_pv_control, amba_pv_snoop_t

Enumerator

AMBA_PV_NON_SHAREABLE	The domain contains a single master.
AMBA_PV_INNER_SHAREABLE	The inner domain can include additional masters.
AMBA_PV_OUTER_SHAREABLE	The outer domain contains all masters in the inner domain and can include additonal masters.
AMBA_PV_SYSTEM	The system domain include all masters in the system.

6.1.3.6 amba_pv_bar_t

`enum amba_pv::amba_pv_bar_t`

Barrier type.

The barrier type indicates the type for barraier transactions and the response to barriers for normal accesses.

The bit representation of this type matches the AxBAR AMBA4 signals.

See also

amba_pv_control, amba_pv_snoop_t

Enumerator

AMBA_PV_RESPECT_BARRIER	Normal access, respecting barriers.
AMBA_PV_MEMORY_BARRIER	Memory barrier.
AMBA_PV_IGNORE_BARRIER	Normal access, ignoring barriers.
AMBA_PV_SYNCHRONISATION_BARRIER	Synchronisation barrier.

6.1.3.7 amba_pv_service_req_t

`enum amba_pv::amba_pv_service_req_t`

Service Request type.

This enumeration declares the known Service Requests.

The ServiceRequestNumber forms part of the PVBus TransactionAttributes and so transactions can be routed based on the ServiceRequestNumber

NOTE: This is a model only feature and does not map to anything in AMBA spec.

Enumerator

AMBA_PV_NO_SERVICE_REQUEST	No service request.
AMBA_PV_PCIE_SERVICE	PCIe Service request.
AMBA_PV_FAR_ATOMIC_SERVICE	Far Atomic Service request (deprecated, use amba_pv_atomic_op_t instead)

6.1.3.8 amba_pv_physical_address_space_t

`enum amba_pv::amba_pv_physical_address_space_t`

Physical Address Space type.

This enumeration declares the known Physical Address Spaces.
The Physical Address Space forms part of the PVBus TransactionAttributes and provides the Address space of the access

Enumerator

AMBA_PV_SECURE_PAS	Secure Physical address space.
AMBA_PV_NON_SECURE_PAS	Non-Secure Physical address space.
AMBA_PV_ROOT_PAS	Root Physical address space.
AMBA_PV_REALM_PAS	Realm Physical address space.
AMBA_PV_SYSTEM_AGENT_PAS	System Agent Physical address space.
AMBA_PV_NON_SECURE_PROTECTED_PAS	Non-Secure Protected Physical address space.
AMBA_PV_NA6_PAS	NA6 Physical address space.
AMBA_PV_NA7_PAS	NA7 Physical address space.

6.1.3.9 amba_pv_mmuflow_t

`enum amba_pv::amba_pv_mmuflow_t`
AxMMUFLOW encodings.
This enumeration declares the encodings for translation fault flows.
An untranslated transaction can indicate which flow can be used when an SMMU encounters a translation fault.

Enumerator

AMBA_PV_MMUFLOW_STALL	The SMMU Stall flow can be used.
AMBA_PV_MMUFLOW_ATST	The SMMU ATST flow can be used.
AMBA_PV_MMUFLOW_NoStall	The SMMU NoStall flow can be used.
AMBA_PV_MMUFLOW_PRI	The SMMU PRI flow can be used.

6.1.3.10 amba_pv_dvm_message_t

`enum amba_pv::amba_pv_dvm_message_t`
DVM Message type.
The bit representation of this type matches the encoding of the DVM message type field in the AxADDR AMBA4 signal.

See also

amba_pv_dvm, amba_pv_dvm_os_t, amba_pv_dvm_security_t, amba_pv_dvm_stage_t

Enumerator

AMBA_PV_TLB_INVALIDATE	TLB invalidate.
AMBA_PV_BRANCH_PREDICTOR_INVALIDATE	Branch predictor invalidate.
AMBA_PV_PHYSICAL_INSTRUCTION_CACHE_↩ INVALIDATE	Physical instruction cache invalidate.
AMBA_PV_VIRTUAL_INSTRUCTION_CACHE_↩ INVALIDATE	Virtual instruction cache invalidate.
AMBA_PV_SYNC	Synchronisation message.
AMBA_PV_HINT	Reserved message type for future Hint messages.

6.1.3.11 amba_pv_dvm_os_t

enum amba_pv::amba_pv_dvm_os_t

DVM message Guest OS or hypervisor type.
The bit representation of this type matches the encoding of the DVM guest OS or hypervisor field in the AxADDR AMBA4 signal.

See also

amba_pv_dvm, amba_pv_dvm_message_t, amba_pv_dvm_security_t, amba_pv_dvm_stage_t

Enumerator

AMBA_PV_HYPERVISOR_OR_GUEST	Transaction applies to hypervisor and all Guest OS.
AMBA_PV_EL3	Transaction applies to EL3.
AMBA_PV_GUEST	Transaction applies to Guest OS.
AMBA_PV_HYPERVISOR	Transaction applies to hypervisor.

6.1.3.12 amba_pv_dvm_security_t

enum amba_pv::amba_pv_dvm_security_t

DVM message security type.
The bit representation of this type matches the encoding of the DVM security field in the AxADDR AMBA4 signal.

See also

amba_pv_dvm, amba_pv_dvm_message_t, amba_pv_dvm_os_t, amba_pv_dvm_stage_t

Enumerator

AMBA_PV_SECURE_AND_NON_SECURE	Transaction applies to Secure and Non-secure.
AMBA_PV_SECURE_ONLY	Transaction applies to Secure only.
AMBA_PV_NON_SECURE_ONLY	Transaction applies to Non-secure only.

6.1.3.13 amba_pv_dvm_stage_t

enum amba_pv::amba_pv_dvm_stage_t

DVM message Staged Invalidation.
The bit representation of this type matches the encoding of the DVM Staged Invalidation field in the AxADDR AMBA4 signal.

See also

amba_pv_dvm, amba_pv_dvm_message_t, amba_pv_dvm_os_t, amba_pv_dvm_security_t, amba_pv_dvm_stage_t

Enumerator

AMBA_PV_DVM_V7	Used for DVMv7 transactions.
AMBA_PV_STAGE_1_ONLY	Stage 1 only invalidation required.
AMBA_PV_STAGE_2_ONLY	Stage 2 only invalidation required.

6.1.3.14 amba_pv_burst_t

`enum amba_pv::amba_pv_burst_t`

Burst type.

The burst type, together with the size information, determines how the address for each transfer within the burst is calculated.

Note

AMBA-PV does not support undefined-length bursts.

See also

amba_pv_extension

Enumerator

AMBA_PV_FIXED	is a fixed-address burst.
AMBA_PV_INCR	is an incrementing-address burst.
AMBA_PV_WRAP	is an incrementing-address burst that wraps to a lower address at the wrap boundary.

6.1.3.15 amba_pv_resp_t

`enum amba_pv::amba_pv_resp_t`

AMBA-PV response type.

The bit representation of this type matches the xRESP AMBA signals. The `AMBA_PV_ACE` macro must be defined at compile time to use the extended ACE responses that indicate cache-line dirty state and cache line might be duplicated hints.

See also

amba_pv_response

Enumerator

AMBA_PV_INCOMPLETE	Indicates that the slave did not attempt to perform the access.
AMBA_PV_OKAY	Indicates that a normal access has been successful. Can also indicate an exclusive access has failed.
AMBA_PV_EXOKAY	Indicates that either the read or write potion of an exclusive access has been successful.
AMBA_PV_SLVERR	Indicates that the access has reached the slave successfully, but the slave returned an error condition to the originating master.
AMBA_PV_DECERR	Indicates that there is no slave at the transaction address. This is typically generated by an interconnect component.

6.1.3.16 amba_pv_protocol_t

`enum amba_pv::amba_pv_protocol_t`

AMBA procotol checks type.

See also

amba_pv_protocol_checker_base

Enumerator

AMBA_PV_APB	selects checking against the APB protocol.
AMBA_PV_AHB	selects checking against the AHB protocol.
AMBA_PV_AXI	selects checking against the AXI3 protocol. **Warning** ARM deprecates this value.
AMBA_PV_AXI3	selects checking against the AXI3 protocol.
AMBA_PV_AXI4_LITE	selects checking against the AXI4-Lite protocol.
AMBA_PV_AXI4	selects checking against the AXI4 protocol.
AMBA_PV_ACE_LITE	selects checking against the ACE-Lite protocol.
AMBA_PV_ACE	selects checking against the ACE protocol.
AMBA_PV_AXI5	selects checking against the AXI5 protocol.

6.1.3.17 signal_command

`enum amba_pv::signal_command`

Signal request command type.

Enumerator

SIGNAL_SET	is the command to issue a set operation.
SIGNAL_GET	is the command to issue a get operation.

6.1.4 Function Documentation

6.1.4.1 amba_pv_atomic_op_string()

```
std::string amba_pv::amba_pv_atomic_op_string (
            const amba_pv_atomic_op_t op ) [inline]
```

converts an atomic op enum to a human readable string

This function returns input variable op as a human readable string.

Parameters

op	specifies the atomic op to be converted

Returns

A string containing the op as human readable text is returned.

6.1.4.2 swap_bytes()

```
void amba_pv::swap_bytes (
```

```
            unsigned char *const data,
            const size_t size )  [inline]
```

Swaps the bytes on a block of memory based on a specified size.

Parameters

data	pointer to the beginning of the data to be byte swapped.
size	size of the data in bytes. Sizes [1, 2, 4, 8] are supported.

Note

It is used by AtomicStore and AtomicLoad to handle endianness conversion.

6.1.4.3 amba_pv_snoop_read_string()

```
std::string amba_pv::amba_pv_snoop_read_string (
            amba_pv_snoop_t snoop,
            amba_pv_domain_t domain,
            amba_pv_bar_t bar )  [inline]
```

Returns the text string representation of the specified read snoop type.
The additional arguments help disambiguate aliased enumerations.

Parameters

snoop	read transaction snoop type
domain	the domain for the transaction
bar	the bar type for the transaction

Returns

the text string representation of *snoop*.

6.1.4.4 amba_pv_snoop_write_string()

```
std::string amba_pv::amba_pv_snoop_write_string (
            amba_pv_snoop_t snoop,
            amba_pv_domain_t domain,
            amba_pv_bar_t bar )  [inline]
```

Returns the text string representation of the specified snoop type for write transactions.
The additional arguments help disambiguate aliased enumerations.

Parameters

snoop	write transaction snoop type
domain	the domain for the transaction
bar	the bar type for the transaction

Returns

the text string representation of *snoop*.

6.1.4.5 amba_pv_domain_string()

```
std::string amba_pv::amba_pv_domain_string (
            amba_pv_domain_t domain ) [inline]
```

Returns the text string representation of the specified domain type.

Parameters

domain	domain type

Returns

the text string representation of *domain*.

6.1.4.6 amba_pv_bar_string()

```
std::string amba_pv::amba_pv_bar_string (
            amba_pv_bar_t bar ) [inline]
```

Returns the text string representation of the specified bar type.

Parameters

bar	bar transaction type

Returns

the text string representation of *bar*.

6.1.4.7 amba_pv_dvm_message_string()

```
std::string amba_pv::amba_pv_dvm_message_string (
            amba_pv_dvm_message_t message_type ) [inline]
```

Returns the text string representation of the specified DVM message type.

Parameters

message_type	DVM message type

Returns

the text string representation of *message_type*.

6.1.4.8 amba_pv_dvm_os_string()

```
std::string amba_pv::amba_pv_dvm_os_string (
            amba_pv_dvm_os_t os ) [inline]
```

Returns the text string representation of the specified DVM Guest OS or hypervisor type.

Parameters

os	DVM Guest OS or hypervisor type

Returns

the text string representation of *os*.

6.1.4.9 amba_pv_dvm_security_string()

```
std::string amba_pv::amba_pv_dvm_security_string (
            amba_pv_dvm_security_t security ) [inline]
```

Returns the text string representation of the specified DVM security type.

Parameters

security	DVM security type

Returns

the text string representation of *security*.

6.1.4.10 amba_pv_burst_string()

```
std::string amba_pv::amba_pv_burst_string (
            amba_pv_burst_t burst ) [inline]
```

Returns the text string representation of the specified burst type.

Parameters

burst	burst type as one of AMBA_PV_FIXED, AMBA_PV_INCR, or AMBA_PV_WRAP.

Returns

the text string representation of *burst*.

6.1.4.11 amba_pv_snoop_string()

```
std::string amba_pv::amba_pv_snoop_string (
            amba_pv_snoop_t snoop ) [inline]
```

Returns the text string representation of the specified snoop type.

Parameters

snoop	snoop type (amba_pv_snoop_t).

Returns

the text string representation of *snoop*.

6.1.4.12 amba_pv_address()

```
sc_dt::uint64 amba_pv::amba_pv_address (
            const sc_dt::uint64 & addr,
            unsigned int length,
            unsigned int size,
            amba_pv_burst_t burst,
            unsigned int n )  [inline]
```

Computes the address of a transfer in a burst.

Parameters

addr	burst address.
length	burst length.
size	burst size in bytes.
burst	burst type as one of `AMBA_PV_FIXED`, `AMBA_PV_INCR`, or `AMBA_PV_WRAP`.
n	burst beat number as in [0 .. (*length* - 1)].

6.1.4.13 amba_pv_resp_string()

```
std::string amba_pv::amba_pv_resp_string (
            amba_pv_resp_t resp )  [inline]
```

Returns the text string representation of the specified AMBA-PV response.

Parameters

resp	AMBA-PV response value.

Returns

text string representation of *resp*.

6.1.4.14 amba_pv_resp_from_tlm()

```
amba_pv_resp_t amba_pv::amba_pv_resp_from_tlm (
            tlm::tlm_response_status response_status,
            bool is_exclusive = false )  [inline]
```

Translates the specified TLM 2.0 response status value into an AMBA-PV response.

Parameters

response_status	TLM 2.0 response status value to translate.
is_exclusive	`true` if the corrresponding transaction is an exclusive access, `false` otherwise (default).

Returns

AMBA-PV response.

6.1.4.15 amba_pv_resp_to_tlm()

```
tlm::tlm_response_status amba_pv::amba_pv_resp_to_tlm (
            amba_pv_resp_t resp,
            bool is_exclusive = false )  [inline]
```

Translates the specified AMBA-PV response value into a TLM 2.0 response status.

Note

AMBA-PV does not use the `tlm::TLM_INCOMPLETE_RESPONSE` response status.

Parameters

resp	AMBA-PV response value to translate.
is_exclusive	`true` if the corrresponding transaction is an exclusive access, `false` otherwise (default).

Returns

TLM 2.0 response status.

6.2 amba_pv::atomic_subop_impl Namespace Reference

AMBA-PV atomic operation type implementation namespace.

Data Structures

- struct do_add

  A functor for in-place addition operation.
- struct do_bit_clear

  A functor for in-place bit clear operation.
- struct do_bit_set

  A functor for in-place bit set operation.
- struct do_signed_max

  A functor for in-place signed max operation.
- struct do_signed_min

  A functor for in-place signed min operation.
- struct do_unsigned_max

  A functor for in-place unsigned max operation.
- struct do_unsigned_min

  A functor for in-place unsigned min operation.
- struct do_xor

  A functor for in-place exclusive or operation.

6.2.1 Detailed Description

AMBA-PV atomic operation type implementation namespace.

6.3 amba_pv::ext Namespace Reference

Extensions namespace.

Data Structures

- class amba_pv_ace_base_master_socket

  AMBA-PV ACE base master socket.
- class amba_pv_ace_base_slave_socket

  AMBA-PV base slave socket.
- class amba_pv_ace_bw_transport_if

  AMBA-PV ACE core transaction interface.
- class amba_pv_ace_master_base

  Base class for all AMBA-PV ACE master modules.
- class amba_pv_ace_master_socket

  AMBA-PV ACE socket to be instantiated on the master side.
- class amba_pv_ace_slave_base

  Base class for all AMBA-PV ACE slave modules.
- class amba_pv_ace_slave_socket

  AMBA-PV ACE socket to be instantiated on the slave side.
- class amba_pv_base_master_socket

  AMBA-PV base master socket.
- class amba_pv_base_slave_socket

  AMBA-PV base slave socket.
- class amba_pv_bw_transport_if

  AMBA-PV core transaction interface.
- class amba_pv_fw_transport_if

  AMBA-PV core transaction interface.
- class amba_pv_master_base

  Base class for all AMBA-PV master modules.
- class amba_pv_master_socket

  AMBA-PV socket to be instantiated on the master side.
- class amba_pv_slave_base

  Base class for all AMBA-PV slave modules.
- class amba_pv_slave_socket

  AMBA-PV socket to be instantiated on the slave side.

6.3.1 Detailed Description

Extensions namespace.
Namespace ext contains extented classes for prelimiary and advanced AMBA modelling.

6.4 tlmx Namespace Reference

TLMX namespace.

Data Structures

- class tlmx_blocking_snoop_if

  TLMX blocking snoop transaction interface.
- class tlmx_bw_transport_if

  TLMX combined backward interface.
- struct tlmx_has_get_protocol_types

  Wrapper around tlm_base_(intiator|target)_socket.
- class tlmx_initiator_socket

  TLMX initiator socket.

- class tlmx_snoop_dbg_if

  TLMX snoop debug transport interface.
- class tlmx_target_socket

  TLMX target socket.

6.4.1 Detailed Description

TLMX namespace.
Namespace tlmx contains classes for TLM 2.0 extended support.

Chapter 7

Data Structure Documentation

7.1 amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL > Class Template Reference

AMBA-PV ACE base master socket.

`#include <sockets/amba_pv_ext_ace_master_socket.h>`

Inherits amba_pv::amba_pv_socket_base, and tlmx::tlmx_initiator_socket< 64, amba_pv_protocol_types, 1, sc_core::SC_ONE_OR_

Public Member Functions

- amba_pv_ace_base_master_socket ()

  Default constructor.
- amba_pv_ace_base_master_socket (const char *, int=0)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this socket.
- virtual void bind (typename base_base_type::base_type &)

  Binds this socket to the specified master socket (hierarchical bind).
- void operator() (typename base_base_type::base_type &)

  Binds this socket to the specified master socket (hierarchical bind).
- virtual void bind (typename base_base_type::base_target_socket_type &)

  Binds this socket to the specified slave socket.
- void operator() (typename base_base_type::base_target_socket_type &)

  Binds this socket to the specified slave socket.
- virtual void bind (base_master_socket_type &)

  Binds this socket to the specified master socket (hierarchical bind).
- void operator() (base_master_socket_type &)

  Binds this socket to the specified master socket (hierarchical bind).
- virtual void bind (base_slave_socket_type &)

  Binds this socket to the specified slave socket.
- void operator() (base_slave_socket_type &)

  Binds this socket to the specified slave socket.
- virtual void bind (amba_pv_ace_bw_transport_if &)

  Binds the specified interface to this socket.
- void operator() (amba_pv_ace_bw_transport_if &)

  Binds the specified interface to this socket.

7.1.1 Detailed Description

template<unsigned int BUSWIDTH = 64, int N = 1, sc_core::sc_port_policy POL = sc_core::SC_ONE_OR_MORE_BOUND>
class amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >

AMBA-PV ACE base master socket.
This socket inherits from the tlmx::tlmx_initiator_socket full-duplex socket class and implements a tagged socket. A tagged socket enables a component to determine through which socket an incoming method call arrived. This is useful when there are multiple master sockets such as in, for example, a bus decoder.
To use this class, you must define the `AMBA_PV_INCLUDE_HIERARCHICAL_BINDING` macro at compile time.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.
N	number of bindings. Defaults to 1.
POL	port binding policy. Defaults to `sc_core::SC_ONE_OR_MORE_BOUND`.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 amba_pv_ace_base_master_socket() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >::amba_pv_ace_base_master_↩
socket  [inline]
```

Default constructor.

7.1.2.2 amba_pv_ace_base_master_socket() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >::amba_pv_ace_base_master_↩
socket (
            const char * name,
            int socket_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.
socket↩ _id	socket identifier (defaults to 0).

7.1.3 Member Function Documentation

7.1.3.1 kind()

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
const char * amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >::kind  [inline],
[virtual]
```

Returns the kind string of this socket.
Reimplemented from tlmx::tlmx_initiator_socket< 64, amba_pv_protocol_types, 1, sc_core::SC_ONE_OR_MORE_BOUND >.
Reimplemented in amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL >.

7.1.3.2 bind() [1/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >::bind (
            typename base_base_type::base_type & s )  [inline], [virtual]
```

Binds this socket to the specified master socket (hierarchical bind).

Note

When binding master socket to master socket, the socket of the child must be bound to the socket of the parent.

Parameters

s	tlm::tlm_base_initiator_socket_b master socket to bind to this socket.

7.1.3.3 operator()() [1/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >::operator() (
            typename base_base_type::base_type & s )  [inline]
```

Binds this socket to the specified master socket (hierarchical bind).

Note

When binding master socket to master socket, the socket of the child must be bound to the socket of the parent.

Parameters

s	tlm::tlm_base_initiator_socket_b master socket to bind to this socket.

7.1.3.4 bind() [2/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >::bind (
            typename base_base_type::base_target_socket_type & s )  [inline], [virtual]
```

Binds this socket to the specified slave socket.

Parameters

s	tlm::tlm_baset_target_socket_b slave socket to bind to this socket.

7.1.3.5 operator()() [2/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >::operator() (
            typename base_base_type::base_target_socket_type & s )  [inline]
```

Binds this socket to the specified slave socket.

Parameters

s	tlm::tlm_baset_target_socket_b slave socket to bind to this socket.

7.1.3.6 bind() [3/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >::bind (
            base_master_socket_type & s ) [inline], [virtual]
```

Binds this socket to the specified master socket (hierarchical bind).

Note

When binding master socket to master socket, the socket of the child must be bound to the socket of the parent.

Parameters

s	amba_pv_ace_base_master_socket master socket to bind to this socket.

7.1.3.7 operator()() [3/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >::operator() (
            base_master_socket_type & s ) [inline]
```

Binds this socket to the specified master socket (hierarchical bind).

Note

When binding master socket to master socket, the socket of the child must be bound to the socket of the parent.

Parameters

s	amba_pv_ace_base_master_socket master socket to bind to this socket.

7.1.3.8 bind() [4/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >::bind (
            base_slave_socket_type & s ) [inline], [virtual]
```

Binds this socket to the specified slave socket.

Parameters

s	amba_pv_ace_base_slave_socket slave socket to bind to this socket.

7.1.3.9 operator()() [4/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >::operator() (
            base_slave_socket_type & s ) [inline]
```

Binds this socket to the specified slave socket.

Parameters

s	amba_pv_ace_base_slave_socket slave socket to bind to this socket.

7.1.3.10 bind() [5/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >::bind (
            amba_pv_ace_bw_transport_if & iface ) [inline], [virtual]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_ace_bw_transport_if interface to bind to this socket.

7.1.3.11 operator()() [5/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >::operator() (
            amba_pv_ace_bw_transport_if & iface ) [inline]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_ace_bw_transport_if interface to bind to this socket.

7.2 amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL > Class Template Reference

AMBA-PV base slave socket.

`#include <sockets/amba_pv_ext_ace_slave_socket.h>`

Inherits amba_pv::amba_pv_socket_base, and tlmx::tlmx_target_socket< 64, amba_pv_protocol_types, 1, sc_core::SC_ONE_OR_M

Public Member Functions

- amba_pv_ace_base_slave_socket ()

  Default constructor.
- amba_pv_ace_base_slave_socket (const char *, int=0)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this socket.
- virtual void bind (typename base_base_type::base_type &)

  Binds this socket to the specified slave socket (hierarchical bind).
- void operator() (typename base_base_type::base_type &)

  Binds this socket to the specified slave socket (hierarchical bind).
- virtual void bind (typename base_base_type::base_initiator_socket_type &)

  Binds this socket to the specified master socket.
- void operator() (typename base_base_type::base_initiator_socket_type &)

  Binds this socket to the specified master socket.

- virtual void bind (base_slave_socket_type &)

  Binds this socket to the specified slave socket (hierarchical bind).
- void operator() (base_slave_socket_type &)

  Binds this socket to the specified slave socket (hierarchical bind).
- virtual void bind (base_master_socket_type &)

  Binds this socket to the specified master socket.
- void operator() (base_master_socket_type &)

  Binds this socket to the specified master socket.
- virtual void bind (amba_pv_fw_transport_if &)

  Binds the specified interface to this socket.
- void operator() (amba_pv_fw_transport_if &)

  Binds the specified interface to this socket.

7.2.1 Detailed Description

template<unsigned int BUSWIDTH = 64, int N = 1, sc_core::sc_port_policy POL = sc_core::SC_ONE_OR_MORE_BOUND>
class amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >

AMBA-PV base slave socket.
This socket inherits from the tlmx::tlmx_target_socket full-duplex socket class and implements a tagged socket. A tagged socket allows a component to determine through which socket an incoming method call arrived. This is useful when there are multiple slave sockets such as in, for example, a bus decoder or a multi-port memory.
To use this class, you must define the `AMBA_PV_INCLUDE_HIERARCHICAL_BINDING` macro at compile time.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.
N	number of bindings. Defaults to 1.
POL	port binding policy. Defaults to `sc_core::SC_ONE_OR_MORE_BOUND`.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 amba_pv_ace_base_slave_socket() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >::amba_pv_ace_base_slave_socket
[inline]
```

Default constructor.

7.2.2.2 amba_pv_ace_base_slave_socket() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >::amba_pv_ace_base_slave_socket
(
            const char * name,
            int socket_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.
socket↩ _id	socket identifier (defaults to 0).

7.2.3 Member Function Documentation

7.2.3.1 kind()

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
const char * amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >::kind  [inline],
[virtual]
```

Returns the kind string of this socket.
Reimplemented from tlmx::tlmx_target_socket< 64, amba_pv_protocol_types, 1, sc_core::SC_ONE_OR_MORE_BOUND >.
Reimplemented in amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL >.

7.2.3.2 bind() [1/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >::bind (
            typename base_base_type::base_type & s )  [inline], [virtual]
```

Binds this socket to the specified slave socket (hierarchical bind).

Note

When binding slave socket to slave socket, the socket of the parent must be bound to the socket of the child.

Parameters

s	tlm::tlm_base_target_sockt_b slave socket to bind to this socket.

7.2.3.3 operator()() [1/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >::operator() (
            typename base_base_type::base_type & s )  [inline]
```

Binds this socket to the specified slave socket (hierarchical bind).

Note

When binding slave socket to slave socket, the socket of the parent must be bound to the socket of the child.

Parameters

s	tlm::tlm_base_target_sockt_b slave socket to bind to this socket.

7.2.3.4 bind() [2/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >::bind (
            typename base_base_type::base_initiator_socket_type & s )  [inline], [virtual]
```

Binds this socket to the specified master socket.

Parameters

s	tlm::tlm_base_initiator_socket_b master socket to bind to this socket.

7.2.3.5 operator()() [2/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >::operator() (
            typename base_base_type::base_initiator_socket_type & s )  [inline]
```

Binds this socket to the specified master socket.

Parameters

s	tlm::tlm_base_initiator_socket_b master socket to bind to this socket.

7.2.3.6 bind() [3/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >::bind (
            base_slave_socket_type & s )  [inline], [virtual]
```

Binds this socket to the specified slave socket (hierarchical bind).

Note

When binding slave socket to slave socket, the socket of the parent must be bound to the socket of the child.

Parameters

s	amba_pv_ace_base_slave_socket slave socket to bind to this socket.

7.2.3.7 operator()() [3/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >::operator() (
            base_slave_socket_type & s )  [inline]
```

Binds this socket to the specified slave socket (hierarchical bind).

Note

When binding slave socket to slave socket, the socket of the parent must be bound to the socket of the child.

Parameters

s	amba_pv_ace_base_slave_socket slave socket to bind to this socket.

7.2.3.8 bind() [4/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >::bind (
            base_master_socket_type & s )  [inline], [virtual]
```

Binds this socket to the specified master socket.

Parameters

s	amba_pv_ace_base_master_socket master socket to bind to this socket.

7.2.3.9 operator()() [4/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >::operator() (
            base_master_socket_type & s ) [inline]
```

Binds this socket to the specified master socket.

Parameters

s	amba_pv_ace_base_master_socket master socket to bind to this socket.

7.2.3.10 bind() [5/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >::bind (
            amba_pv_fw_transport_if & iface ) [inline], [virtual]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_ace_fw_transport_if interface to bind to this socket.

7.2.3.11 operator()() [5/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >::operator() (
            amba_pv_fw_transport_if & iface ) [inline]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_ace_fw_transport_if interface to bind to this socket.

7.3 amba_pv::amba_pv_ace_bw_transport_if Class Reference

AMBA-PV core additional transaction interface for ACE.
`#include <core/amba_pv_core_ifs.h>`
Inherits amba_pv::amba_pv_bw_transport_if, and amba_pv::amba_pv_bw_snoop_if.
Inherited by amba_pv::amba_pv_ace_master_base [virtual], amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH > [vir
and amba_pv::amba_pv_ace_simple_probe< BUSWIDTH > [virtual].

Additional Inherited Members

7.3.1 Detailed Description

AMBA-PV core additional transaction interface for ACE.
This is a composite interface that combines amba_pv_bw_transport_if and amba_pv_bw_snoop_if.

Note

AMBA-PV ACE masters must implement the amba_pv_ace_bw_transport_if interface.

7.4 amba_pv::ext::amba_pv_ace_bw_transport_if Class Reference

AMBA-PV ACE core transaction interface.
`#include <core/amba_pv_ext_core_ifs.h>`
Inherits sc_core::sc_interface.
Inherited by amba_pv::ext::amba_pv_ace_master_base `[virtual]`.

Public Member Functions

- virtual void b_snoop (int socket_id, amba_pv_transaction &trans, sc_core::sc_time &t)=0

  Blocking snoop.
- virtual unsigned int snoop_dbg (int socket_id, amba_pv_transaction &trans)=0

  Debug snoop access to a master.
- virtual void invalidate_direct_mem_ptr (int socket_id, sc_dt::uint64 start_range, sc_dt::uint64 end_range)=0

  Invalidates DMI pointers previously established for the specified DMI region.

7.4.1 Detailed Description

AMBA-PV ACE core transaction interface.
This is a tagged variant of the tlmx::tlmx_bw_transport_if interface. This interface is used for the backward snoop path.

Note

AMBA-PV ACE masters must implement the amba_pv_ace_bw_transport_if interface.

7.4.2 Member Function Documentation

7.4.2.1 b_snoop()

```
virtual void amba_pv::ext::amba_pv_ace_bw_transport_if::b_snoop (
            int socket_id,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [pure virtual]
```

Blocking snoop.

Parameters

socket_id	socket identifier (index into bound interfaces on the slave side).
trans	transaction.
t	timing annotation.

Implemented in amba_pv::ext::amba_pv_ace_master_base.

7.4.2.2 snoop_dbg()

```
virtual unsigned int amba_pv::ext::amba_pv_ace_bw_transport_if::snoop_dbg (
            int socket_id,
            amba_pv_transaction & trans )  [pure virtual]
```

Debug snoop access to a master.

This use the same path as the b_snoop() interface. This debug access must be performed without any of the delays, waits, event notifications or side effects associated with a regular snoop transaction. This debug access is, therefore, non-intrusive.

Parameters

socket_id	socket identifier (index into bound interfaces on the slave side).
trans	transaction.

Returns

number of bytes read or written or, if error, 0.

Implemented in amba_pv::ext::amba_pv_ace_master_base.

7.4.2.3 invalidate_direct_mem_ptr()

```
virtual void amba_pv::ext::amba_pv_ace_bw_transport_if::invalidate_direct_mem_ptr (
             int socket_id,
             sc_dt::uint64 start_range,
             sc_dt::uint64 end_range )  [pure virtual]
```

Invalidates DMI pointers previously established for the specified DMI region.

Parameters

socket_id	socket identifier (index into bound interfaces on the slave side).
start_range	DMI region start address.
end_range	DMI region end address.

Implemented in amba_pv::ext::amba_pv_ace_master_base.

7.5 amba_pv::amba_pv_ace_master_base Class Reference

Base class for all AMBA-PV ACE master modules.
`#include <user/amba_pv_ace_master_base.h>`
Inherits amba_pv::amba_pv_ace_bw_transport_if.

Public Member Functions

- amba_pv_ace_master_base (const std::string &)

  Constructor.
- std::string get_name () const

  Returns the name of this master.

Protected Member Functions

- virtual void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)

  Invalidates DMI pointers previously established for the specified DMI region.
- virtual void b_snoop (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking snoop transport.
- virtual unsigned int snoop_dbg (int, amba_pv_transaction &)

  Snoop debug transport.

7.5.1 Detailed Description

Base class for all AMBA-PV ACE master modules.
amba_pv_ace_master_base is intended to be bound to one or more amba_pv_ace_master_socket.

Note

amba_pv_ace_master_base is not an `sc_module`.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 amba_pv_ace_master_base()

```
amba_pv::amba_pv_ace_master_base::amba_pv_ace_master_base (
            const std::string & name )  [inline], [explicit]
```

Constructor.

Parameters

name	master name.

7.5.3 Member Function Documentation

7.5.3.1 get_name()

```
std::string amba_pv::amba_pv_ace_master_base::get_name ( ) const  [inline]
```

Returns the name of this master.

7.5.3.2 invalidate_direct_mem_ptr()

```
void amba_pv::amba_pv_ace_master_base::invalidate_direct_mem_ptr (
            int ,
            sc_dt::uint64 ,
            sc_dt::uint64  )  [inline], [protected], [virtual]
```

Invalidates DMI pointers previously established for the specified DMI region.
This default implementation does nothing.
Implements amba_pv::amba_pv_bw_transport_if.

7.5.3.3 b_snoop()

```
void amba_pv::amba_pv_ace_master_base::b_snoop (
            int ,
            amba_pv_transaction & ,
            sc_core::sc_time &  )  [inline], [protected], [virtual]
```

Blocking snoop transport.
This default implementation does nothing.
Implements amba_pv::amba_pv_bw_snoop_if.

7.5.3.4 snoop_dbg()

```
unsigned int amba_pv::amba_pv_ace_master_base::snoop_dbg (
            int ,
            amba_pv_transaction &  )  [inline], [protected], [virtual]
```

Snoop debug transport.
This default implementation does nothing.
Implements amba_pv::amba_pv_bw_snoop_if.

7.6 amba_pv::ext::amba_pv_ace_master_base Class Reference

Base class for all AMBA-PV ACE master modules.
`#include <user/amba_pv_ext_ace_master_base.h>`
Inherits amba_pv::ext::amba_pv_ace_bw_transport_if.

Public Member Functions

- amba_pv_ace_master_base (const std::string &)

  Constructor.
- std::string get_name () const

  Returns the name of this master.

Protected Member Functions

- virtual void b_snoop (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking snoop.
- virtual unsigned int snoop_dbg (int, amba_pv_transaction &)

  Debug access to a master.
- virtual void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)

  Invalidates DMI pointers previously established for the specified DMI region.

7.6.1 Detailed Description

Base class for all AMBA-PV ACE master modules.
amba_pv_ace_master_base is intended to be bound to one or more amba_pv_ace_master_socket.

Note

amba_pv_ace_master_base is not an `sc_module`.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 amba_pv_ace_master_base()

```
amba_pv::ext::amba_pv_ace_master_base::amba_pv_ace_master_base (
            const std::string & name )  [inline], [explicit]
```

Constructor.

Parameters

name	master name.

7.6.3 Member Function Documentation

7.6.3.1 get_name()

```
std::string amba_pv::ext::amba_pv_ace_master_base::get_name ( ) const  [inline]
```

Returns the name of this master.

7.6.3.2 b_snoop()

```
void amba_pv::ext::amba_pv_ace_master_base::b_snoop (
            int ,
            amba_pv_transaction & ,
            sc_core::sc_time &  )  [inline], [protected], [virtual]
```

Blocking snoop.
This version of the method does nothing.
Implements amba_pv::ext::amba_pv_ace_bw_transport_if.

7.6.3.3 snoop_dbg()

```
unsigned int amba_pv::ext::amba_pv_ace_master_base::snoop_dbg (
            int ,
            amba_pv_transaction &  )  [inline], [protected], [virtual]
```

Debug access to a master.
This version of the method returns 0.
Implements amba_pv::ext::amba_pv_ace_bw_transport_if.

7.6.3.4 invalidate_direct_mem_ptr()

```
void amba_pv::ext::amba_pv_ace_master_base::invalidate_direct_mem_ptr (
            int ,
            sc_dt::uint64 ,
            sc_dt::uint64  )  [inline], [protected], [virtual]
```

Invalidates DMI pointers previously established for the specified DMI region.
This version of the method does nothing.
Implements amba_pv::ext::amba_pv_ace_bw_transport_if.

7.7 amba_pv::amba_pv_ace_master_socket< BUSWIDTH > Class Template Reference

AMBA-PV ACE socket to be instantiated on the master side.
`#include <sockets/amba_pv_ace_master_socket.h>`
Inherits amba_pv::amba_pv_master_socket< 64 >.

Public Member Functions

- amba_pv_ace_master_socket ()
  Default constructor.
- amba_pv_ace_master_socket (const char *, int=0)
  Constructor.
- virtual const char * kind () const
  Returns the kind string for this socket.
- void bind (amba_pv_ace_slave_socket< BUSWIDTH > &)
  Binds the specified ACE slave socket to this ACE master socket.
- void operator() (amba_pv_ace_slave_socket< BUSWIDTH > &)
  Binds the specified ACE slave socket to this ACE master socket.
- void bind (amba_pv_ace_bw_transport_if &)
  Binds the specified interface to this socket.
- void operator() (amba_pv_ace_bw_transport_if &)

Binds the specified interface to this socket.

7.7.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_ace_master_socket< BUSWIDTH >

AMBA-PV ACE socket to be instantiated on the master side.
This socket is for use as an AMBA ACE master socket bound to one or more AMBA ACE slave sockets. The amba_pv_ace_master_socket directly inherits from the amba_pv_master_socket class, but in addition includes an extra upstream TLM interface as well as the downstream TLM interface. The upstream TLM interface is used to model the snoop channels that the AMBA ACE bus architecture requires.
amba_pv_ace_master_socket provides implementations for the amba_pv_bw_transport_if and amba_pv_bw_snoop_if user-layer interfaces the composite interface name is amba_pv_bw_transport_and_snoop_if.
The upstream path is implemented using an additional master/slave socket pair that are private data members of amba_pv_ace_slave_socket and amba_pv_ace_master_socket respectively. This extra upstream socket pair are automatically bound when the downstream master to slave sockets are bound.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 amba_pv_ace_master_socket() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_ace_master_socket< BUSWIDTH >::amba_pv_ace_master_socket  [inline]
```

Default constructor.

7.7.2.2 amba_pv_ace_master_socket() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_ace_master_socket< BUSWIDTH >::amba_pv_ace_master_socket (
            const char * name,
            int socket_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.
socket_id	socket identifier (defaults to 0).

7.7.3 Member Function Documentation

7.7.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_ace_master_socket< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string for this socket.
Reimplemented from amba_pv::amba_pv_master_socket< 64 >.

7.7.3.2 bind() [1/2]

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_ace_master_socket< BUSWIDTH >::bind (
            amba_pv_ace_slave_socket< BUSWIDTH > & slave ) [inline]
```

Binds the specified ACE slave socket to this ACE master socket.
This method will also bind the ACE snoop master socket in the ACE slave socket to the ACE snoop slave socket in this socket.

Parameters

slave	amba_pv_ace_slave_socket to bind to this socket.

7.7.3.3 operator()() [1/2]

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_ace_master_socket< BUSWIDTH >::operator() (
            amba_pv_ace_slave_socket< BUSWIDTH > & slave ) [inline]
```

Binds the specified ACE slave socket to this ACE master socket.

Parameters

slave	amba_pv_ace_slave_socket to bind to this socket.

7.7.3.4 bind() [2/2]

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_ace_master_socket< BUSWIDTH >::bind (
            amba_pv_ace_bw_transport_if & iface ) [inline]
```

Binds the specified interface to this socket.
This method will also bind the ACE snoop slave socket in this socket to the interface.

Parameters

iface	amba_pv_ace_bw_transport_if interface to bind to this socket.

7.7.3.5 operator()() [2/2]

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_ace_master_socket< BUSWIDTH >::operator() (
            amba_pv_ace_bw_transport_if & iface ) [inline]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_ace_bw_transport_if interface to bind to this socket.

7.8 amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL > Class Template Reference

AMBA-PV ACE socket to be instantiated on the master side.
`#include <sockets/amba_pv_ext_ace_master_socket.h>`
Inherits amba_pv::ext::amba_pv_ace_base_master_socket< 64, 1, sc_core::SC_ONE_OR_MORE_BOUND >.

Public Member Functions

- amba_pv_ace_master_socket ()
  Default constructor.
- amba_pv_ace_master_socket (const char *, int=0)
  Constructor.
- virtual const char * kind () const
  Returns the kind string of this socket.
- void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)
  Blocking transport.
- void b_transport (amba_pv_transaction &, sc_core::sc_time &)
  Blocking transport.
- unsigned int transport_dbg (int, amba_pv_transaction &)
  Debug access to a target.
- unsigned int transport_dbg (amba_pv_transaction &)
  Debug access to a target.
- bool get_direct_mem_ptr (int, amba_pv_transaction &, tlm::tlm_dmi &)
  Requests a DMI access based on the specified transaction.
- bool get_direct_mem_ptr (amba_pv_transaction &, tlm::tlm_dmi &)
  Requests a DMI access based on the specified transaction.

7.8.1 Detailed Description

template<unsigned int BUSWIDTH = 64, int N = 1, sc_core::sc_port_policy POL = sc_core::SC_ONE_OR_MORE_BOUND>
class amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL >

AMBA-PV ACE socket to be instantiated on the master side.
This socket is for use as an AMBA-PV ACE master socket bound to one or more AMBA-PV ACE slave sockets.
amba_pv_ace_master_socket provides convenience methods for the amba_pv_ace_fw_transport_if interface.
To use this class, you must define the `AMBA_PV_INCLUDE_HIERARCHICAL_BINDING` macro at compile time.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.
N	number of bindings. Defaults to 1.
POL	port binding policy. Defaults to `sc_core::SC_ONE_OR_MORE_BOUND`.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 amba_pv_ace_master_socket() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL >::amba_pv_ace_master_socket  [inline]
```
Default constructor.

7.8.2.2 amba_pv_ace_master_socket() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL >::amba_pv_ace_master_socket (
            const char * name,
            int socket_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.
socket_id	socket identifier (defaults to 0).

7.8.3 Member Function Documentation

7.8.3.1 kind()

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
const char * amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL >::kind  [inline],
[virtual]
```

Returns the kind string of this socket.

Reimplemented from amba_pv::ext::amba_pv_ace_base_master_socket< 64, 1, sc_core::SC_ONE_OR_MORE_BOUND >.

7.8.3.2 b_transport() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL >::b_transport (
            int index,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline]
```

Blocking transport.

This version of the method forwards the b_transport() call to the *index'ed* slave socket bound to this master socket.

Parameters

index	interface index (for sockets bound more than once).
trans	transaction.
t	timing annotation.

7.8.3.3 b_transport() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL >::b_transport (
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline]
```

Blocking transport.

Parameters

trans	transaction.
t	timing annotation.

7.8.3.4 transport_dbg() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
unsigned int amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL >::transport_dbg (
            int index,
            amba_pv_transaction & trans )  [inline]
```

Debug access to a target.
This version of the method forwards the transport_dbg() call to the *index'ed* slave socket bound to this master socket.

Parameters

index	interface index (for sockets bound more than once).
trans	transaction.

Returns

number of bytes read or written or, if error, 0.

7.8.3.5 transport_dbg() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
unsigned int amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL >::transport_dbg (
            amba_pv_transaction & trans )  [inline]
```

Debug access to a target.

Parameters

trans	transaction.

Returns

number of bytes read or written or, if error, 0.

7.8.3.6 get_direct_mem_ptr() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
bool amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL >::get_direct_mem_ptr (
            int index,
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [inline]
```

Requests a DMI access based on the specified transaction.
This version of the method forwards the get_direct_mem_ptr() call to the *index'ed* slave socket bound to this master socket.

Parameters

index	interface index (for sockets bound more than once).
trans	transaction.
dmi_data	DMI Descriptor.

Returns

`true` if DMI access is granted, `false` otherwise.

7.8.3.7 get_direct_mem_ptr() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
bool amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL >::get_direct_mem_ptr (
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [inline]
```

Requests a DMI access based on the specified transaction.

Parameters

trans	transaction.
dmi_data	DMI Descriptor.

Returns

`true` if DMI access is granted, `false` otherwise.

7.9 amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH > Class Template Reference

AMBA-PV ACE protocol checker model.

`#include <models/amba_pv_ace_protocol_checker.h>`

Inherits amba_pv::amba_pv_fw_transport_if, amba_pv::amba_pv_ace_bw_transport_if, and amba_pv::amba_pv_protocol_checker_b

Public Member Functions

- amba_pv_ace_protocol_checker (const sc_core::sc_module_name &, bool=true)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this protocol checker.

Data Fields

- amba_pv_ace_slave_socket< BUSWIDTH > amba_pv_s

  Slave socket.
- amba_pv_ace_master_socket< BUSWIDTH > amba_pv_m

  Master socket.

Protected Member Functions

- virtual void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking transport.
- virtual unsigned int transport_dbg (int, amba_pv_transaction &)

  Debug access to a target.
- virtual bool get_direct_mem_ptr (int, amba_pv_transaction &, tlm::tlm_dmi &)

  Requests a DMI access based on the specified transaction.
- virtual void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)

  Invalidates DMI pointers previously established for the specified DMI region.
- virtual void b_snoop (int, amba_pv_transaction &, sc_core::sc_time &)

Blocking snoop.

- virtual unsigned int snoop_dbg (int, amba_pv_transaction &)

  Debug access to a master.

7.9.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >

AMBA-PV ACE protocol checker model.
The amba_pv_ace_protocol_checker model is used for confirming that your model complies with the AMBA-PV ACE protocol.
You can instantiate the protocol checker between any pair of AMBA-PV ACE master and slave sockets. The transactions that pass through are checked against the AMBA-PV ACE protocol and errors reported using the SystemC reporting mechanism. All errors are reported with a message type of "amba_pv_ace_protocol_checker" and with a severity of SC_ERROR. Recommendations are reported with a severity of SC_WARNING.

Note

The AMBA-PV protocol checker model does not perform any OSCI TLM 2.0 BP checks.

The AMBA-PV protocol checker model might have an effect on performance.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 amba_pv_ace_protocol_checker()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >::amba_pv_ace_protocol_checker (
            const sc_core::sc_module_name & name,
            bool recommend_on = true )  [inline], [explicit]
```

Constructor.
Constructs a new amba_pv_ace_protocol_checker with parameter for configuring recommended rules.

Parameters

name	protocol checker name.
recommend_on	`true` to enable reporting of protocol recommendations, `false` otherwise.

See also

recommend_on()

7.9.3 Member Function Documentation

7.9.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string of this protocol checker.

7.9.3.2 b_transport()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >::b_transport (
             int socket_id,
             amba_pv_transaction & trans,
             sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Blocking transport.
This version of the method completes the transaction and checks it complies with the AMBA buses protocols.
Implements amba_pv::amba_pv_fw_transport_if.

7.9.3.3 transport_dbg()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >::transport_dbg (
             int socket_id,
             amba_pv_transaction & trans )  [inline], [protected], [virtual]
```

Debug access to a target.
This version of the method forwards this debug access to the slave and checks it complies with the AMBA buses protocols.
Implements amba_pv::amba_pv_fw_transport_if.

7.9.3.4 get_direct_mem_ptr()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >::get_direct_mem_ptr (
             int socket_id,
             amba_pv_transaction & trans,
             tlm::tlm_dmi & dmi_data )  [inline], [protected], [virtual]
```

Requests a DMI access based on the specified transaction.
This version of the method forwards this DMI access request to the slave and checks it complies with the AMBA buses protocols.
Implements amba_pv::amba_pv_fw_transport_if.

7.9.3.5 invalidate_direct_mem_ptr()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >::invalidate_direct_mem_ptr (
             int socket_id,
             sc_dt::uint64 start_range,
             sc_dt::uint64 end_range )  [inline], [protected], [virtual]
```

Invalidates DMI pointers previously established for the specified DMI region.
This version of the method forwards this DMI call to the master.
Implements amba_pv::amba_pv_bw_transport_if.

7.9.3.6 b_snoop()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >::b_snoop (
             int socket_id,
             amba_pv_transaction & trans,
             sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Blocking snoop.

This version of the method completes the snoop transaction and checks it complies with the AMBA buses protocols.
Implements amba_pv::amba_pv_bw_snoop_if.

7.9.3.7 snoop_dbg()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >::snoop_dbg (
            int socket_id,
            amba_pv_transaction & trans )  [inline], [protected], [virtual]
```

Debug access to a master.
This version of the method forwards this debug access to the master.
Implements amba_pv::amba_pv_bw_snoop_if.

7.9.4 Field Documentation

7.9.4.1 amba_pv_s

```
template<unsigned int BUSWIDTH = 64>
amba_pv_ace_slave_socket<BUSWIDTH> amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >::amba_↩
pv_s
```

Slave socket.

7.9.4.2 amba_pv_m

```
template<unsigned int BUSWIDTH = 64>
amba_pv_ace_master_socket<BUSWIDTH> amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >::amba↩
_pv_m
```

Master socket.

7.10 amba_pv::amba_pv_ace_simple_probe< BUSWIDTH > Class Template Reference

AMBA-PV ACE simple probe model.
`#include <models/amba_pv_ace_simple_probe.h>`
Inherits amba_pv::amba_pv_fw_transport_if, amba_pv::amba_pv_ace_bw_transport_if, and amba_pv::amba_pv_simple_probe_base

Public Member Functions

- amba_pv_ace_simple_probe (const sc_core::sc_module_name &, bool=true)

  Constructor.
- virtual ~amba_pv_ace_simple_probe ()

  Destructor.
- virtual const char * kind () const

  Returns the kind string of this probe.

Data Fields

- amba_pv_ace_slave_socket< BUSWIDTH > amba_pv_s

  Slave socket.
- amba_pv_ace_master_socket< BUSWIDTH > amba_pv_m

  Master socket.

Protected Member Functions

- virtual void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking transport.
- virtual unsigned int transport_dbg (int, amba_pv_transaction &)

  Debug access to a target.
- virtual bool get_direct_mem_ptr (int, amba_pv_transaction &, tlm::tlm_dmi &)

  Requests a DMI access based on the specified transaction.
- virtual void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)

  Invalidates DMI pointers previously established for the specified DMI region.
- virtual void b_snoop (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking snoop.
- virtual unsigned int snoop_dbg (int, amba_pv_transaction &)

  Debug access to a master.

7.10.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >

AMBA-PV ACE simple probe model.
The amba_pv_ace_simple_probe model prints the contents of transaction between a master and a slave to `std::cout`, a file, or a stream.

Note

If configured for printing transactions, the amba_pv_ace_simple_probe model might have an effect on performance.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 amba_pv_ace_simple_probe()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >::amba_pv_ace_simple_probe (
            const sc_core::sc_module_name & name,
            bool trans_verbose = true )  [inline], [explicit]
```

Constructor.

Parameters

name	probe name.
trans_verbose	`true` to print transactions (default), `false` otherwise.

See also

set_trans_verbose()

7.10.2.2 ~amba_pv_ace_simple_probe()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >::~amba_pv_ace_simple_probe  [inline], [virtual]
```

Destructor.

7.10.3 Member Function Documentation

7.10.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string of this probe.

Reimplemented from amba_pv::amba_pv_simple_probe_base< 64 >.

7.10.3.2 b_transport()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >::b_transport (
            int socket_id,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Blocking transport.

This version of the method completes the transaction and prints its contents.

Implements amba_pv::amba_pv_fw_transport_if.

7.10.3.3 transport_dbg()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >::transport_dbg (
            int socket_id,
            amba_pv_transaction & trans )  [inline], [protected], [virtual]
```

Debug access to a target.

This version of the method forwards this debug access to the slave and prints its contents.

Implements amba_pv::amba_pv_fw_transport_if.

7.10.3.4 get_direct_mem_ptr()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >::get_direct_mem_ptr (
            int socket_id,
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [inline], [protected], [virtual]
```

Requests a DMI access based on the specified transaction.

This version of the method forwards this DMI access request to the slave and prints its contents.

Implements amba_pv::amba_pv_fw_transport_if.

7.10.3.5 invalidate_direct_mem_ptr()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >::invalidate_direct_mem_ptr (
            int socket_id,
            sc_dt::uint64 start_range,
            sc_dt::uint64 end_range )  [inline], [protected], [virtual]
```

Invalidates DMI pointers previously established for the specified DMI region.
This version of the method forwards this DMI call to the master after printing its arguments.
Implements amba_pv::amba_pv_bw_transport_if.

7.10.3.6 b_snoop()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >::b_snoop (
             int socket_id,
             amba_pv_transaction & trans,
             sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Blocking snoop.
This version of the method completes the snoop transaction and prints its contents.
Implements amba_pv::amba_pv_bw_snoop_if.

7.10.3.7 snoop_dbg()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >::snoop_dbg (
             int socket_id,
             amba_pv_transaction & trans )  [inline], [protected], [virtual]
```

Debug access to a master.
This version of the method forwards this debug access to the master.
Implements amba_pv::amba_pv_bw_snoop_if.

7.10.4 Field Documentation

7.10.4.1 amba_pv_s

```
template<unsigned int BUSWIDTH = 64>
amba_pv_ace_slave_socket<BUSWIDTH> amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >::amba_pv_s
```

Slave socket.

7.10.4.2 amba_pv_m

```
template<unsigned int BUSWIDTH = 64>
amba_pv_ace_master_socket<BUSWIDTH> amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >::amba_pv_m
```

Master socket.

7.11 amba_pv::ext::amba_pv_ace_slave_base Class Reference

Base class for all AMBA-PV ACE slave modules.
`#include <user/amba_pv_ext_ace_slave_base.h>`
Inherits amba_pv::ext::amba_pv_fw_transport_if.

Public Member Functions

- amba_pv_ace_slave_base (const std::string &)

  Constructor.
- std::string get_name () const

  Returns the name of this slave.

Protected Member Functions

- virtual void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking transport.
- virtual unsigned int transport_dbg (int, amba_pv_transaction &)

  Debug access to a target.
- virtual bool get_direct_mem_ptr (int, amba_pv_transaction &, tlm::tlm_dmi &)

  Requests a DMI access based on the specified transaction.

7.11.1 Detailed Description

Base class for all AMBA-PV ACE slave modules.
amba_pv_ace_slave_base is intended to be bound to one or more ∗ amba_pv_ace_slave_socket.

Note

amba_pv_ace_slave_base is not an `sc_module`.

amba_pv_slave_base can also be used for AMBA-PV ACE slave modules. It is provided as an alternative, especially for AMBA-PV ACE interconnect components.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 amba_pv_ace_slave_base()

```
amba_pv::ext::amba_pv_ace_slave_base::amba_pv_ace_slave_base (
            const std::string & name )  [inline], [explicit]
```

Constructor.

Parameters

name	slave name.

7.11.3 Member Function Documentation

7.11.3.1 get_name()

```
std::string amba_pv::ext::amba_pv_ace_slave_base::get_name ( ) const  [inline]
```

Returns the name of this slave.

7.11.3.2 b_transport()

```
void amba_pv::ext::amba_pv_ace_slave_base::b_transport (
            int ,
            amba_pv_transaction & ,
            sc_core::sc_time &  )  [inline], [protected], [virtual]
```

Blocking transport.
This version of the method does nothing.
Implements amba_pv::ext::amba_pv_fw_transport_if.

7.11.3.3 transport_dbg()

```
unsigned int amba_pv::ext::amba_pv_ace_slave_base::transport_dbg (
```

```
            int ,
            amba_pv_transaction &  )  [inline], [protected], [virtual]
```

Debug access to a target.
This version of the method returns `0`.
Implements amba_pv::ext::amba_pv_fw_transport_if.

7.11.3.4 get_direct_mem_ptr()

```
bool amba_pv::ext::amba_pv_ace_slave_base::get_direct_mem_ptr (
            int ,
            amba_pv_transaction & ,
            tlm::tlm_dmi & dmi_data )  [inline], [protected], [virtual]
```

Requests a DMI access based on the specified transaction.
This version of the method returns `false` and denies DMI access to the entire memory region.
Implements amba_pv::ext::amba_pv_fw_transport_if.

7.12 amba_pv::amba_pv_ace_slave_socket< BUSWIDTH > Class Template Reference

AMBA-PV ACE socket to be instantiated on the slave side.
`#include <sockets/amba_pv_ace_slave_socket.h>`
Inherits amba_pv::amba_pv_slave_socket< 64 >.

Public Member Functions

- amba_pv_ace_slave_socket ()
  Default constructor.
- amba_pv_ace_slave_socket (const char *, int=0)
  Constructor.
- virtual const char * kind () const
  Returns the kind string of this socket.
- void bind (amba_pv_snoop_socket< BUSWIDTH > &)
  Binds the specified snoop slave socket to the snoop master socket.
- void b_snoop (int, amba_pv_transaction &, sc_core::sc_time &)
  Blocking upstream transport.
- void b_snoop (amba_pv_transaction &, sc_core::sc_time &)
  Blocking upstream transport.
- unsigned int snoop_dbg (int, amba_pv_transaction &)
  Upstream debug transport.
- unsigned int snoop_dbg (amba_pv_transaction &)
  Upstream debug transport.

7.12.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_ace_slave_socket< BUSWIDTH >

AMBA-PV ACE socket to be instantiated on the slave side.
This socket is for use as an AMBA ACE slave socket that is bound by an AMBA ACE master socket amba_pv_ace_master_socket. The amba_pv_ace_slave_socket directly inherits from the amba_pv_slave_socket class, but in addition includes an extra upstream TLM interface as well as the downstream TLM interface. The upstream TLM interface is used to model the snoop channels that the AMBA ACE bus architecture requires.
The upstream path is implemented using an additional master/slave socket pair that are private data members of amba_pv_ace_slave_socket and amba_pv_ace_master_socket respectively. This extra upstream socket pair are automatically bound when the downstream master to slave sockets are bound.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 amba_pv_ace_slave_socket() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_ace_slave_socket< BUSWIDTH >::amba_pv_ace_slave_socket  [inline]
```

Default constructor.

7.12.2.2 amba_pv_ace_slave_socket() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_ace_slave_socket< BUSWIDTH >::amba_pv_ace_slave_socket (
            const char * name,
            int socket_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.
socket↩ _id	socket identifier (defaults to 0).

7.12.3 Member Function Documentation

7.12.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_ace_slave_socket< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string of this socket.

Reimplemented from amba_pv::amba_pv_slave_socket< 64 >.

7.12.3.2 bind()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_ace_slave_socket< BUSWIDTH >::bind (
            amba_pv_snoop_socket< BUSWIDTH > & upstream )  [inline]
```

Binds the specified snoop slave socket to the snoop master socket.

Parameters

upstream	amba_pv_ace_slave_socket to bind to the snoop master socket.

7.12.3.3 b_snoop() [1/2]

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_ace_slave_socket< BUSWIDTH >::b_snoop (
            int socket_id,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline]
```

Blocking upstream transport.
This version of the method forwards the b_transport() call to the ACE master socket bound to this ACE slave socket using the upstream snoop pathway.

Parameters

socket_id	socket identifier (ignored on the slave side).
trans	transaction.
t	timing annotation.

7.12.3.4 b_snoop() [2/2]

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_ace_slave_socket< BUSWIDTH >::b_snoop (
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline]
```

Blocking upstream transport.

Parameters

trans	transaction.
t	timing annotation.

7.12.3.5 snoop_dbg() [1/2]

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_ace_slave_socket< BUSWIDTH >::snoop_dbg (
            int socket_id,
            amba_pv_transaction & trans )  [inline]
```

Upstream debug transport.
This version of the method forwards the transport_dbg() call to the ACE master socket bound to this ACE slave socket using the upstream snoop pathway.

Parameters

socket_id	socket identifier (ignored on the slave side).
trans	transaction.

7.12.3.6 snoop_dbg() [2/2]

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_ace_slave_socket< BUSWIDTH >::snoop_dbg (
            amba_pv_transaction & trans )  [inline]
```

Upstream debug transport.

Parameters

trans	transaction.

7.13 amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL > Class Template Reference

AMBA-PV ACE socket to be instantiated on the slave side.
`#include <sockets/amba_pv_ext_ace_slave_socket.h>`
Inherits amba_pv::ext::amba_pv_ace_base_slave_socket< 64, 1, sc_core::SC_ONE_OR_MORE_BOUND >.

Public Member Functions

- amba_pv_ace_slave_socket ()
  Default constructor.
- amba_pv_ace_slave_socket (const char *, int=0)
  Constructor.
- virtual const char * kind () const
  Returns the kind string of this socket.
- void b_snoop (int, amba_pv_transaction &, sc_core::sc_time &)
  Blocking snoop.
- void b_snoop (amba_pv_transaction &, sc_core::sc_time &)
  Blocking snoop.
- unsigned int snoop_dbg (int, amba_pv_transaction &)
  Debug access to a master.
- unsigned int snoop_dbg (amba_pv_transaction &)
  Debug access to a master.
- void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)
  Invalidates DMI pointers previously established for the specified DMI region.
- void invalidate_direct_mem_ptr (sc_dt::uint64, sc_dt::uint64)
  Invalidates DMI pointers previously established for the specified DMI region.

7.13.1 Detailed Description

template<unsigned int BUSWIDTH = 64, int N = 1, sc_core::sc_port_policy POL = sc_core::SC_ONE_OR_MORE_BOUND>
class amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL >

AMBA-PV ACE socket to be instantiated on the slave side.
This socket is for use as an AMBA-PV ACE slave socket bound to one or more AMBA-PV ACE master sockets.
To use this class, you must define the `AMBA_PV_INCLUDE_HIERARCHICAL_BINDING` macro at compile time.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.
N	number of bindings. Defaults to 1.
POL	port binding policy. Defaults to `sc_core::SC_ONE_OR_MORE_BOUND`.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 amba_pv_ace_slave_socket() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL >::amba_pv_ace_slave_socket  [inline]
```

Default constructor.

7.13.2.2 amba_pv_ace_slave_socket() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL >::amba_pv_ace_slave_socket (
            const char * name,
            int socket_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.
socket_id	socket identifier (defaults to 0).

7.13.3 Member Function Documentation

7.13.3.1 kind()

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
const char * amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL >::kind  [inline], [virtual]
```

Returns the kind string of this socket.

Reimplemented from amba_pv::ext::amba_pv_ace_base_slave_socket< 64, 1, sc_core::SC_ONE_OR_MORE_BOUND >.

7.13.3.2 b_snoop() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL >::b_snoop (
            int index,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline]
```

Blocking snoop.

Parameters

index	interface index (for sockets bound more than once).
trans	snoop transaction
t	timing annotation.

This version of the method forwards the b_snoop() call to the *index'ed* master socket this slave socket is bound to.

7.13.3.3 b_snoop() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL >::b_snoop (
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline]
```

Blocking snoop.

Parameters

trans	snoop transaction
t	timing annotation.

This version of the method forwards the b_snoop() call to the master socket this slave socket is bound to.

7.13.3.4 snoop_dbg() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
unsigned int amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL >::snoop_dbg (
            int index,
            amba_pv_transaction & trans )  [inline]
```

Debug access to a master.

Parameters

index	interface index (for sockets bound more than once).
trans	debug transaction.

This version of the method forwards the snoop_dbg() call to the *index'ed* master socket this slave socket is bound to.

7.13.3.5 snoop_dbg() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
unsigned int amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL >::snoop_dbg (
            amba_pv_transaction & trans )  [inline]
```

Debug access to a master.

Parameters

trans	debug transaction.

This version of the method forwards the snoop_dbg() call to the master socket this slave socket is bound to.

7.13.3.6 invalidate_direct_mem_ptr() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL >::invalidate_direct_mem_ptr (
            int index,
            sc_dt::uint64 start_range,
            sc_dt::uint64 end_range )  [inline]
```

Invalidates DMI pointers previously established for the specified DMI region.

Parameters

index	interface index (for sockets bound more than once).
start_range	DMI region start address.
end_range	DMI region end address.

This version of the method forwards the invalidate_direct_mem_ptr() call to the *index'ed* master socket this slave socket is bound to.

7.13.3.7 invalidate_direct_mem_ptr() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
```

```
void amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL >::invalidate_direct_mem_ptr (
            sc_dt::uint64 start_range,
            sc_dt::uint64 end_range )  [inline]
```

Invalidates DMI pointers previously established for the specified DMI region.

Parameters

start_range	DMI region start address.
end_range	DMI region end address.

This version of the method forwards the invalidate_direct_mem_ptr() call to the master socket this slave socket is bound to.

7.14 amba_pv::amba_pv_address_map Class Reference

AMBA-PV address mapping information structure.

`#include <models/amba_pv_address_map.h>`

Public Member Functions

- amba_pv_address_map ()

  Default constuctor.
- virtual ~amba_pv_address_map ()

  Destuctor.
- void add_region (const sc_dt::uint64 &, const sc_dt::uint64 &, const std::string &, int=-1)

  Add a memory region to this address map.
- bool decode (const sc_dt::uint64 &, amba_pv_address_region *&) const

  Returns whether the specified address a is decoded by this address map.
- iterator begin ()

  Returns an iterator that can be used to begin traversing this map.
- const_iterator begin () const

  Returns an iterator that can be used to begin traversing this map.
- iterator end ()

  Returns an iterator that can be used in a comparison for ending traversal through this map.
- const_iterator end () const

  Returns an iterator that can be used in a comparison for ending traversal through this map.
- size_type size () const

  Returns the number of address regions currently stored in this map.
- reference operator[ ] (size_type)

  Returns the address region of specified index n from the beginning of this map in constant time.
- const_reference operator[ ] (size_type) const

  Returns the address region of specified index n from the beginning of this map in constant time.
- reference at (size_type)

  Returns the address region of specified index n from the beginning of this map in constant time.
- const_reference at (size_type) const

  Returns the address region of specified index n from the beginning of this map in constant time.
- amba_pv_address_map (const amba_pv_address_map &)

  Copy constuctor.
- amba_pv_address_map & operator= (const amba_pv_address_map &)

  Copy assignement.

7.14.1 Detailed Description

AMBA-PV address mapping information structure.
amba_pv_address_map is typically used to implement address decoding, as in amba_pv_decoder.

Note

The implementation of this class does not make any attempt at ensuring that AXI or AHB recommendations on minimal address space allocated to a single slave are met.

See also

amba_pv_decoder

7.14.2 Constructor & Destructor Documentation

7.14.2.1 amba_pv_address_map() [1/2]

```
amba_pv::amba_pv_address_map::amba_pv_address_map ( )  [inline]
```

Default constuctor.

7.14.2.2 ~amba_pv_address_map()

```
amba_pv::amba_pv_address_map::~amba_pv_address_map ( )  [inline], [virtual]
```

Destuctor.

7.14.2.3 amba_pv_address_map() [2/2]

```
amba_pv::amba_pv_address_map::amba_pv_address_map (
            const amba_pv_address_map & map )  [inline]
```

Copy constuctor.

Parameters

map	reference to another address map.

7.14.3 Member Function Documentation

7.14.3.1 add_region()

```
void amba_pv::amba_pv_address_map::add_region (
            const sc_dt::uint64 & start,
            const sc_dt::uint64 & end,
            const std::string & name,
            int rank = -1 )  [inline]
```

Add a memory region to this address map.

Parameters

start	start address of this memory region.
end	end address of this memory region.
name	name of the associated slave port.
rank	rank of the associated slave port.

7.14.3.2 decode()

```
bool amba_pv::amba_pv_address_map::decode (
            const sc_dt::uint64 & a,
            amba_pv_address_region *& r ) const  [inline]
```

Returns whether the specified address *a* is decoded by this address map.

Parameters

a	address to decode.
r	pointer to the memory address region that decodes the given address *a*.

Returns

`true` if this address map contains a region that decodes the given address *a*, `false` otherwise.

7.14.3.3 begin() [1/2]

```
std::vector< amba_pv_address_region * >::iterator amba_pv::amba_pv_address_map::begin ( )
[inline]
```

Returns an iterator that can be used to begin traversing this map.

7.14.3.4 begin() [2/2]

```
std::vector< amba_pv_address_region * >::const_iterator amba_pv::amba_pv_address_map::begin (
) const  [inline]
```

Returns an iterator that can be used to begin traversing this map.

7.14.3.5 end() [1/2]

```
std::vector< amba_pv_address_region * >::iterator amba_pv::amba_pv_address_map::end ( )  [inline]
```

Returns an iterator that can be used in a comparison for ending traversal through this map.

7.14.3.6 end() [2/2]

```
std::vector< amba_pv_address_region * >::const_iterator amba_pv::amba_pv_address_map::end ( )
const  [inline]
```

Returns an iterator that can be used in a comparison for ending traversal through this map.

7.14.3.7 size()

```
std::vector< amba_pv_address_region * >::size_type amba_pv::amba_pv_address_map::size ( )
const  [inline]
```

Returns the number of address regions currently stored in this map.

7.14.3.8 operator[]() [1/2]

```
std::vector< amba_pv_address_region * >::reference amba_pv::amba_pv_address_map::operator[] (
            size_type n )  [inline]
```

Returns the address region of specified index *n* from the beginning of this map in constant time.

Parameters

n	index of the address region to return (zero-based).

7.14.3.9 operator[]() [2/2]

```
std::vector< amba_pv_address_region * >::const_reference amba_pv::amba_pv_address_map::operator[]
(
            size_type n ) const  [inline]
```

Returns the address region of specified index *n* from the beginning of this map in constant time.

Parameters

n	index of the address region to return (zero-based).

7.14.3.10 at() [1/2]

```
std::vector< amba_pv_address_region * >::reference amba_pv::amba_pv_address_map::at (
            size_type n )  [inline]
```

Returns the address region of specified index *n* from the beginning of this map in constant time.
Bounds checking is performed.

Parameters

n	index of the address region to return (zero-based).

7.14.3.11 at() [2/2]

```
std::vector< amba_pv_address_region * >::const_reference amba_pv::amba_pv_address_map::at (
            size_type n ) const  [inline]
```

Returns the address region of specified index *n* from the beginning of this map in constant time.
Bounds checking is performed.

Parameters

n	index of the address region to return (zero-based).

7.14.3.12 operator=()

```
amba_pv_address_map & amba_pv::amba_pv_address_map::operator= (
            const amba_pv_address_map & map )  [inline]
```

Copy assignement.

Parameters

map	reference to another address map.

7.15 amba_pv::amba_pv_address_region Class Reference

AMBA-PV address region structure.
`#include <models/amba_pv_address_map.h>`

Public Member Functions

- amba_pv_address_region (const sc_dt::uint64 &, const sc_dt::uint64 &, const std::string &, int=-1)

  Constuctor.
- sc_dt::uint64 get_start () const

  Returns the start address of this region.
- void set_start (const sc_dt::uint64 &)

  Sets the start address of this region.
- sc_dt::uint64 get_end () const

  Returns the end address of this region.
- void set_end (const sc_dt::uint64 &)

  Sets the end address of this region.
- std::string get_slave_name () const

  Returns the name of the slave port associated to this address region.
- void set_slave_name (const std::string &)

  Sets the name of the slave port associated to this address region.
- int get_slave_rank () const

  Returns the rank of the slave port associated to this address region.
- void set_slave_rank (int)

  Sets the rank of the slave port associated to this address region.
- bool decode (const sc_dt::uint64 &) const

  Returns whether the specified address a is decoded by this address region.

7.15.1 Detailed Description

AMBA-PV address region structure.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 amba_pv_address_region()

```
amba_pv::amba_pv_address_region::amba_pv_address_region (
            const sc_dt::uint64 & start,
            const sc_dt::uint64 & end,
            const std::string & name,
            int rank = -1 )  [inline]
```

Constuctor.

Parameters

start	start address of this region.
end	end address of this region.
name	name of the associated slave port.
rank	rank of the associated slave port.

7.15.3 Member Function Documentation

7.15.3.1 get_start()

```
sc_dt::uint64 amba_pv::amba_pv_address_region::get_start ( ) const  [inline]
```

Returns the start address of this region.

7.15.3.2 set_start()

```
void amba_pv::amba_pv_address_region::set_start (
            const sc_dt::uint64 & start )  [inline]
```

Sets the start address of this region.

Parameters

start	start address of this region.

7.15.3.3 get_end()

```
sc_dt::uint64 amba_pv::amba_pv_address_region::get_end ( ) const  [inline]
```

Returns the end address of this region.

7.15.3.4 set_end()

```
void amba_pv::amba_pv_address_region::set_end (
            const sc_dt::uint64 & end )  [inline]
```

Sets the end address of this region.

Parameters

end	end address of this region.

7.15.3.5 get_slave_name()

```
std::string amba_pv::amba_pv_address_region::get_slave_name ( ) const  [inline]
```

Returns the name of the slave port associated to this address region.

7.15.3.6 set_slave_name()

```
void amba_pv::amba_pv_address_region::set_slave_name (
            const std::string & name )  [inline]
```

Sets the name of the slave port associated to this address region.

Parameters

name	name of the associated slave port.

7.15.3.7 get_slave_rank()

```
int amba_pv::amba_pv_address_region::get_slave_rank ( ) const  [inline]
```

Returns the rank of the slave port associated to this address region.

7.15.3.8 set_slave_rank()

```
void amba_pv::amba_pv_address_region::set_slave_rank (
            int rank )  [inline]
```

Sets the rank of the slave port associated to this address region.

Parameters

rank	rank of the associated slave port.

7.15.3.9 decode()

```
bool amba_pv::amba_pv_address_region::decode (
            const sc_dt::uint64 & a ) const  [inline]
```

Returns whether the specified address *a* is decoded by this address region.

Parameters

a	address to decode.

Returns

`true` if this region decodes the given address *a*, `false` otherwise.

7.16 amba_pv::amba_pv_atomic Class Reference

Provides atomic transaction information used by AMBA AXI buses.
`#include <bus/amba_pv_atomic.h>`
Inherited by amba_pv::amba_pv_extension.

Public Member Functions

- amba_pv_atomic (amba_pv_atomic_op_t)
  Constructor.
- amba_pv_atomic (amba_pv_atomic_op_t, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t)
  Constructor.
- void set_atomic_op (amba_pv_atomic_op_t)
  Sets the atomic transaction type.
- amba_pv_atomic_op_t get_atomic_op () const
  Returns the atomic transaction type.
- void set_atomic_subop (amba_pv_atomic_subop_t)
  Sets the atomic transaction operation type.
- amba_pv_atomic_subop_t get_atomic_subop () const
  Returns the atomic transaction operation type.
- void set_atomic_endianness (amba_pv_atomic_endianness_t)
  Sets the atomic operation endianness.
- amba_pv_atomic_endianness_t get_atomic_endianness () const

Returns the atomic operation endianness.

- void reset ()

  Resets all members to their default value.

- bool is_atomic_size_supported (size_t) const

  Checks if the transaction size is supported by the current atomic transaction.

- bool is_atomic_endianness_valid () const

  Check if the big endian is supported by the current atomic transaction.

- bool is_atomic_request_valid (size_t) const

  sanity checks the current atomic transaction

- bool is_atomic_subop_valid () const

  check if the sub-operation setting is valid

7.16.1 Detailed Description

Provides atomic transaction information used by AMBA AXI buses.
This class contains the information for representing the AXI AWATOP AMBA5 signals.
This class is used as a base class for the AMBA-PV extension type (amba_pv_extension).

See also

amba_pv_extension

7.16.2 Constructor & Destructor Documentation

7.16.2.1 amba_pv_atomic() [1/2]

```
amba_pv::amba_pv_atomic::amba_pv_atomic (
            amba_pv_atomic_op_t op )  [inline], [explicit]
```

Constructor.

Parameters

op	atomic transaction type.

7.16.2.2 amba_pv_atomic() [2/2]

```
amba_pv::amba_pv_atomic::amba_pv_atomic (
            amba_pv_atomic_op_t op,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness )  [inline]
```

Constructor.

Parameters

op	atomic transaction type.
subop	atomic transaction operation type.
endianness	atomic endianness.

7.16.3 Member Function Documentation

7.16.3.1 set_atomic_op()

```
void amba_pv::amba_pv_atomic::set_atomic_op (
            amba_pv_atomic_op_t op ) [inline]
```

Sets the atomic transaction type.

Parameters

op	atomic transaction type.

See also

get_atomic_op()

7.16.3.2 get_atomic_op()

```
amba_pv_atomic_op_t amba_pv::amba_pv_atomic::get_atomic_op ( ) const [inline]
```

Returns the atomic transaction type.

See also

set_atomic_op()

7.16.3.3 set_atomic_subop()

```
void amba_pv::amba_pv_atomic::set_atomic_subop (
            amba_pv_atomic_subop_t subop ) [inline]
```

Sets the atomic transaction operation type.

Parameters

subop	atomic transaction operation type.

See also

get_atomic_subop()

7.16.3.4 get_atomic_subop()

```
amba_pv_atomic_subop_t amba_pv::amba_pv_atomic::get_atomic_subop ( ) const [inline]
```

Returns the atomic transaction operation type.

See also

set_atomic_subop()

7.16.3.5 set_atomic_endianness()

```
void amba_pv::amba_pv_atomic::set_atomic_endianness (
            amba_pv_atomic_endianness_t endianness ) [inline]
```

Sets the atomic operation endianness.

Parameters

endianness	atomic endianness.

See also

get_atomic_endianness()

7.16.3.6 get_atomic_endianness()

```
amba_pv_atomic_endianness_t amba_pv::amba_pv_atomic::get_atomic_endianness ( ) const  [inline]
```

Returns the atomic operation endianness.

See also

set_atomic_endianness()

7.16.3.7 reset()

```
void amba_pv::amba_pv_atomic::reset ( )  [inline]
```

Resets all members to their default value.

7.16.3.8 is_atomic_size_supported()

```
bool amba_pv::amba_pv_atomic::is_atomic_size_supported (
            size_t size ) const  [inline]
```

Checks if the transaction size is supported by the current atomic transaction.
AtomicStore, AtomicLoad and AtomicSwap supports sizes of 1, 2, 4 or 8 bytes. AtomicCompare supports sizes of 2, 4, 8, 16 or 32 bytes.

Parameters

size	selects the transaction size to be tested.

Returns

Returns true if the size is supported by the current atomic operation, otherwise false.

7.16.3.9 is_atomic_endianness_valid()

```
bool amba_pv::amba_pv_atomic::is_atomic_endianness_valid ( ) const  [inline]
```

Check if the big endian is supported by the current atomic transaction.
Big endian is supported only by AtomicStore and AtomicLoad, with non-bitwise operations.

7.16.3.10 is_atomic_request_valid()

```
bool amba_pv::amba_pv_atomic::is_atomic_request_valid (
            size_t size ) const  [inline]
```

sanity checks the current atomic transaction
Burst size and endianness support are checked.

Parameters

size	selects the transaction size to be tested.

Returns

Returns true if size and endianness checks pass, otherwise false.

See also

is_atomic_size_supported(), is_atomic_endianness_valid

7.16.3.11 is_atomic_subop_valid()

```
bool amba_pv::amba_pv_atomic::is_atomic_subop_valid ( ) const  [inline]
```

check if the sub-operation setting is valid
If the operation is set to either AMBA_PV_ATOMICSTORE or AMBA_PV_ATOMICLOAD, then the sub operation can assume any value and the method always returns true. For any other operation the sub operation must be set to AMBA_PV_ATOMIC_ADD. If that is not the case, then the set up is deemed invalid.

Returns

True is returned if subop is valid, otherwise false.

7.17 amba_pv::amba_pv_atomic_utils Class Reference

An utility class that offers the implementation of executing an atomic transaction.
`#include <bus/amba_pv_atomic_utils.h>`

Static Public Member Functions

- static void atomic_store (unsigned char ∗memory, unsigned char ∗data, size_t size, amba_pv_atomic_subop_t subop, amba_pv_atomic_endianness_t endianness)

  Completes an atomic store transaction.
- static void atomic_load (unsigned char ∗memory, unsigned char ∗data, size_t size, amba_pv_atomic_subop_t subop, amba_pv_atomic_endianness_t endianness)

  Completes an atomic load transaction.
- static void atomic_swap (unsigned char ∗memory, unsigned char ∗data, size_t size)

  Completes an atomic swap transaction.
- static bool atomic_compare (unsigned char ∗memory, unsigned char ∗data, size_t size, bool compare_first)

  Completes an atomic compare transaction.

7.17.1 Detailed Description

An utility class that offers the implementation of executing an atomic transaction.
The class contains the implementation of AtomicStore, AtomicLoad, AtomicSwap and AtomicCompare.

7.17.2 Member Function Documentation

7.17.2.1 atomic_store()

```
void amba_pv::amba_pv_atomic_utils::atomic_store (
            unsigned char * memory,
            unsigned char * data,
            size_t size,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness )  [inline], [static]
```

Completes an atomic store transaction.
Memory value is updated according to the atomic subop and incoming data.

Parameters

memory	data pointer pointing to the target address in the memory.
data	transaction data pointer. It must point to an array of *size* bytes.
size	transaction size in bytes as one of [1, 2, 4, 8]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
subop	operation type of the atomic transaction.
endianness	endianness of the atomic operation. If enabled, memory value and incoming data is interpreted in big endian order.

7.17.2.2 atomic_load()

```
void amba_pv::amba_pv_atomic_utils::atomic_load (
            unsigned char * memory,
            unsigned char * data,
            size_t size,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness )  [inline], [static]
```

Completes an atomic load transaction.
Memory value is updated according to the atomic subop and incoming data, the original memory value is then returned to the data pointer.

Parameters

memory	data pointer pointing to the target address in the memory.
data	transaction data pointer. It must point to an array of *size* bytes. The array should initially contain the sending data, then the original data at the address before the atomic operation is returned to the array.
size	transaction size in bytes as one of [1, 2, 4, 8]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
subop	operation type of the atomic transaction.
endianness	endianness of the atomic operation. If enabled, memory value and incoming data is interpreted in big endian order.

7.17.2.3 atomic_swap()

```
void amba_pv::amba_pv_atomic_utils::atomic_swap (
            unsigned char * memory,
            unsigned char * data,
            size_t size )  [inline], [static]
```

Completes an atomic swap transaction.
Memory value is updated as the incoming data, the original memory value is then returned to the data pointer.

Parameters

memory	data pointer pointing to the target address in the memory.
data	transaction data pointer. It must point to an array of *size* bytes. The array should initially contain the sending data, then the original data at the address before the atomic operation is returned to the array.
size	transaction size in bytes as one of [1, 2, 4, 8]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().

7.17.2.4 atomic_compare()

```
bool amba_pv::amba_pv_atomic_utils::atomic_compare (
            unsigned char * memory,
            unsigned char * data,
            size_t size,
            bool compare_first )  [inline], [static]
```

Completes an atomic compare transaction.
Initiator sends out comparing and swapping data. Memory value is checked against the comparing data, if they are equal, the memory value is updated as the swapping data. The original memory value is then returned to the data pointer.

Parameters

memory	data pointer pointing to the target address in the memory.
data	transaction data pointer. It must point to an array of *size* bytes. The array should initially comprise of comparing and swapping data, after the transaction, the original data at the address before the atomic operation is returned to the array.
size	transaction size in bytes as one of [2, 4, 8, 16, 32]. It accommodates both comparing and swapping data, with each occupying half the size. The transaction size must be less than or equal to the value returned by set_bus_width_bytes().
compare_first	Comparing data is sent before swapping data.

Returns

`true` if memory value is swapped, `false` otherwise.

Note

Transaction data pointer points to an array comprising of comparing and swapping data, with their order determined by the transaction address. Comparing data is sent first if the address points to the lower half of the transaction (i.e. lowest address byte); swapping data is sent first if the address points to the upper half of the transaction (i.e. lowest address plus half the ∗*size*∗).

The original data at the address is returned to *data* pointer starting from the lowest byte. The returning size is half the sending *size*.

7.18 amba_pv::amba_pv_attributes Class Reference

Provides support for additional user-defined attributes.
`#include <bus/amba_pv_attributes.h>`
Inherited by amba_pv::amba_pv_control.

Data Structures

- class attribute_ref

  A reference to a specific attribute in a map of attributes that is not accessed until it is required.
- class const_attribute_ref

  A `const` reference to a specific attribute in a map of attributes that is not accessed until it is required.

Public Member Functions

- amba_pv_attributes ()

  Default constructor.
- virtual ∼amba_pv_attributes ()

Destructor.

- void add_attributes (const map_type &)

  Adds a set of attributes.
- attribute_ref operator[ ] (const std::string &)

  Returns a reference to an attribute.
- const_attribute_ref operator[ ] (const std::string &) const

  Returns a reference to an attribute.
- template<typename T >
  void set_attribute (const std::string &, T)

  Sets the value of a given attribute.
- template<typename T >
  bool get_attribute (const std::string &, T &) const

  Returns the value of a given attribute.
- const_iterator_type attributes_begin () const

  Returns an iterator that can be used to begin traversing through the attributes.
- const_iterator_type attributes_end () const

  Returns an iterator that can be used in comparison for ending traversal through the attributes.
- size_t attributes_size () const

  Returns the number of attributes currently stored.
- void remove_attributes (const map_type &)

  Removes a set of attributes.
- size_t remove_attribute (const std::string &)

  Removes the given attribute.
- void clear ()

  Clears content.

7.18.1 Detailed Description

Provides support for additional user-defined attributes.
The amba_pv_attributes class provides support for *user signals* in the form of additional named attributes (namely a map).
This class is used as a base class for the AMBA-PV protocol additional control information (amba_pv_control) type.
To use this class, you must define the `AMBA_PV_INCLUDE_ATTRIBUTES` macro at compile time.

Note

This class might impact simulation performance.

See also

amba_pv_control

7.18.2 Constructor & Destructor Documentation

7.18.2.1 amba_pv_attributes()

`amba_pv::amba_pv_attributes::amba_pv_attributes ( )  [inline]`
Default constructor.

7.18.2.2 ~amba_pv_attributes()

`amba_pv::amba_pv_attributes::~amba_pv_attributes ( )  [inline], [virtual]`
Destructor.

7.18.3 Member Function Documentation

7.18.3.1 add_attributes()

```
void amba_pv::amba_pv_attributes::add_attributes (
            const map_type & m )  [inline]
```

Adds a set of attributes.

Parameters

m	additional attributes to be added.

7.18.3.2 operator[]() [1/2]

```
amba_pv_attributes::attribute_ref amba_pv::amba_pv_attributes::operator[] (
            const std::string & n )  [inline]
```

Returns a reference to an attribute.
The returned reference can be used later to assign to the attribute.

Parameters

n	the name of the attribute.

Note

This cannot be used on a `const` amba_pv_attributes.

7.18.3.3 operator[]() [2/2]

```
amba_pv_attributes::const_attribute_ref amba_pv::amba_pv_attributes::operator[] (
            const std::string & n ) const  [inline]
```

Returns a reference to an attribute.
The returned reference can be used later to read the attribute.

Parameters

n	the name of the attribute.

7.18.3.4 set_attribute()

```
template<typename T >
void amba_pv::amba_pv_attributes::set_attribute (
            const std::string & n,
            T v )  [inline]
```

Sets the value of a given attribute.

Parameters

n	the name of the attribute.
v	the value of the attribute.

See also

operator[ ]()

7.18.3.5 get_attribute()

```
template<typename T >
bool amba_pv::amba_pv_attributes::get_attribute (
            const std::string & n,
            T & v ) const  [inline]
```

Returns the value of a given attribute.

Parameters

n	the name of the attribute.
v	the value of the attribute.

See also

operator[ ]()

7.18.3.6 attributes_begin()

```
std::unordered_map< std::string, std::string >::const_iterator amba_pv::amba_pv_attributes↩
::attributes_begin ( ) const  [inline]
```

Returns an iterator that can be used to begin traversing through the attributes.

See also

attributes_end(), attributes_size()

7.18.3.7 attributes_end()

```
std::unordered_map< std::string, std::string >::const_iterator amba_pv::amba_pv_attributes↩
::attributes_end ( ) const  [inline]
```

Returns an iterator that can be used in comparison for ending traversal through the attributes.

See also

attributes_begin(), attributes_size()

7.18.3.8 attributes_size()

```
size_t amba_pv::amba_pv_attributes::attributes_size ( ) const  [inline]
```

Returns the number of attributes currently stored.

See also

attributes_begin(), attributes_end()

7.18.3.9 remove_attributes()

```
void amba_pv::amba_pv_attributes::remove_attributes (
            const map_type & m )  [inline]
```

Removes a set of attributes.

Parameters

m	attributes to be removed.

See also

remove_attribute(), clear()

7.18.3.10 remove_attribute()

```
size_t amba_pv::amba_pv_attributes::remove_attribute (
            const std::string & n )  [inline]
```

Removes the given attribute.

Parameters

n	the name of the attribute.

See also

remove_attributes(), clear()

7.18.3.11 clear()

```
void amba_pv::amba_pv_attributes::clear ( )  [inline]
```

Clears content.
All attributes are dropped.

See also

remove_attributes(), remove_attribute()

7.19 amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL > Class Template Reference

AMBA-PV base master socket.
`#include <sockets/amba_pv_ext_master_socket.h>`
Inherits amba_pv::amba_pv_socket_base, and tlm::tlm_initiator_socket< 64, amba_pv_protocol_types, 1, sc_core::SC_ONE_OR_MORE_BOUND >.

Public Member Functions

- amba_pv_base_master_socket ()
  Default constructor.
- amba_pv_base_master_socket (const char *, int=0)
  Constructor.
- virtual const char * kind () const
  Returns the kind string of this socket.
- virtual void bind (typename base_base_type::base_type &)
  Binds this socket to the specified master socket (hierarchical bind).
- void operator() (typename base_base_type::base_type &)
  Binds this socket to the specified master socket (hierarchical bind).
- virtual void bind (typename base_base_type::base_target_socket_type &)

Binds this socket to the specified slave socket.

- void operator() (typename base_base_type::base_target_socket_type &)

  Binds this socket to the specified slave socket.
- virtual void bind (base_master_socket_type &)

  Binds this socket to the specified master socket (hierarchical bind).
- void operator() (base_master_socket_type &)

  Binds this socket to the specified master socket (hierarchical bind).
- virtual void bind (base_slave_socket_type &)

  Binds this socket to the specified slave socket.
- void operator() (base_slave_socket_type &)

  Binds this socket to the specified slave socket.
- virtual void bind (amba_pv_bw_transport_if &)

  Binds the specified interface to this socket.
- void operator() (amba_pv_bw_transport_if &)

  Binds the specified interface to this socket.

7.19.1 Detailed Description

template<unsigned int BUSWIDTH = 64, int N = 1, sc_core::sc_port_policy POL = sc_core::SC_ONE_OR_MORE_BOUND>
class amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >

AMBA-PV base master socket.
This socket inherits from the OSCI TLM 2.0 `tlm::tlm_initiator_socket` class and implements a tagged socket. A tagged socket enables a component to determine through which socket an incoming method call arrived. This is required if there are multiple master sockets such as in, for example, a bus decoder.
To use this class, you must define the `AMBA_PV_INCLUDE_HIERARCHICAL_BINDING` macro at compile time.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.
N	number of bindings. Defaults to 1.
POL	port binding policy. Defaults to `sc_core::SC_ONE_OR_MORE_BOUND`.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 amba_pv_base_master_socket() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >::amba_pv_base_master_socket
[inline]
```

Default constructor.

7.19.2.2 amba_pv_base_master_socket() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >::amba_pv_base_master_socket (
            const char * name,
            int socket_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.

Parameters

socket↩ _id	socket identifier (defaults to 0).

7.19.3 Member Function Documentation

7.19.3.1 kind()

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
const char * amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >::kind  [inline],
[virtual]
```

Returns the kind string of this socket.
Reimplemented in amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >.

7.19.3.2 bind() [1/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >::bind (
            typename base_base_type::base_type & s )  [inline], [virtual]
```

Binds this socket to the specified master socket (hierarchical bind).

Note

When binding master socket to master socket, the socket of the child must be bound to the socket of the parent.

Parameters

s	`tlm::tlm_base_initiator_socket_b` master socket to bind to this socket.

7.19.3.3 operator()() [1/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >::operator() (
            typename base_base_type::base_type & s )  [inline]
```

Binds this socket to the specified master socket (hierarchical bind).

Note

When binding master socket to master socket, the socket of the child must be bound to the socket of the parent.

Parameters

s	`tlm::tlm_base_initiator_socket_b` master socket to bind to this socket.

7.19.3.4 bind() [2/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
```

```
void amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >::bind (
            typename base_base_type::base_target_socket_type & s )  [inline], [virtual]
```

Binds this socket to the specified slave socket.

Parameters

s	tlm::tlm_base_target_socket_b slave socket to bind to this socket.

7.19.3.5 operator()() [2/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >::operator() (
            typename base_base_type::base_target_socket_type & s )  [inline]
```

Binds this socket to the specified slave socket.

Parameters

s	tlm::tlm_base_target_socket_b slave socket to bind to this socket.

7.19.3.6 bind() [3/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >::bind (
            base_master_socket_type & s )  [inline], [virtual]
```

Binds this socket to the specified master socket (hierarchical bind).

Note

When binding master socket to master socket, the socket of the child must be bound to the socket of the parent.

Parameters

s	amba_pv_base_master_socket master socket to bind to this socket.

7.19.3.7 operator()() [3/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >::operator() (
            base_master_socket_type & s )  [inline]
```

Binds this socket to the specified master socket (hierarchical bind).

Note

When binding master socket to master socket, the socket of the child must be bound to the socket of the parent.

Parameters

s	amba_pv_base_master_socket master socket to bind to this socket.

7.19.3.8 bind() [4/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >::bind (
            base_slave_socket_type & s ) [inline], [virtual]
```

Binds this socket to the specified slave socket.

Parameters

s	amba_pv_base_slave_socket slave socket to bind to this socket.

7.19.3.9 operator()() [4/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >::operator() (
            base_slave_socket_type & s ) [inline]
```

Binds this socket to the specified slave socket.

Parameters

s	amba_pv_base_slave_socket slave socket to bind to this socket.

7.19.3.10 bind() [5/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >::bind (
            amba_pv_bw_transport_if & iface ) [inline], [virtual]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_bw_transport_if interface to bind to this socket.

7.19.3.11 operator()() [5/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >::operator() (
            amba_pv_bw_transport_if & iface ) [inline]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_bw_transport_if interface to bind to this socket.

7.20 amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL > Class Template Reference

AMBA-PV base slave socket.

`#include <sockets/amba_pv_ext_slave_socket.h>`

Inherits amba_pv::amba_pv_socket_base, and tlm::tlm_target_socket< 64, amba_pv_protocol_types, 1, sc_core↩ ::SC_ONE_OR_MORE_BOUND >.

Public Member Functions

- amba_pv_base_slave_socket ()

  Default constructor.
- amba_pv_base_slave_socket (const char *, int=0)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this socket.
- virtual void bind (typename base_base_type::base_type &)

  Binds this socket to the specified slave socket (hierarchical bind).
- void operator() (typename base_base_type::base_type &)

  Binds this socket to the specified slave socket (hierarchical bind).
- virtual void bind (typename base_base_type::base_initiator_socket_type &)

  Binds this socket to the specified master socket.
- void operator() (typename base_base_type::base_initiator_socket_type &)

  Binds this socket to the specified master socket.
- virtual void bind (base_slave_socket_type &)

  Binds this socket to the specified slave socket (hierarchical bind).
- void operator() (base_slave_socket_type &)

  Binds this socket to the specified slave socket (hierarchical bind).
- virtual void bind (base_master_socket_type &)

  Binds this socket to the specified master socket.
- void operator() (base_master_socket_type &)

  Binds this socket to the specified master socket.
- virtual void bind (amba_pv_fw_transport_if &)

  Binds the specified interface to this socket.
- void operator() (amba_pv_fw_transport_if &)

  Binds the specified interface to this socket.

7.20.1 Detailed Description

template<unsigned int BUSWIDTH = 64, int N = 1, sc_core::sc_port_policy POL = sc_core::SC_ONE_OR_MORE_BOUND>
class amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >

AMBA-PV base slave socket.
This socket inherits from the OSCI TLM 2.0 `tlm::tlm_target_socket` class and implements a tagged socket. A tagged socket allows a component to determine through which socket an incoming method call arrived. This is required if there are multiple slave sockets such as in, for example, an interconnect or a multi-port memory. To use this class, you must define the `AMBA_PV_INCLUDE_HIERARCHICAL_BINDING` macro at compile time.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.
N	number of bindings. Defaults to 1.
POL	port binding policy. Defaults to `sc_core::SC_ONE_OR_MORE_BOUND`.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 amba_pv_base_slave_socket() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >::amba_pv_base_slave_socket  [inline]
```

Default constructor.

7.20.2.2 amba_pv_base_slave_socket() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >::amba_pv_base_slave_socket (
            const char * name,
            int socket_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.
socket_id	socket identifier (defaults to 0).

7.20.3 Member Function Documentation

7.20.3.1 kind()

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
const char * amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >::kind  [inline],
[virtual]
```

Returns the kind string of this socket.
Reimplemented in amba_pv::ext::amba_pv_slave_socket< BUSWIDTH, N, POL >.

7.20.3.2 bind() [1/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >::bind (
            typename base_base_type::base_type & s )  [inline], [virtual]
```

Binds this socket to the specified slave socket (hierarchical bind).

Note

When binding slave socket to slave socket, the socket of the parent must be bound to the socket of the child.

Parameters

s	`tlm::tlm_base_target_socket_b` slave socket to bind to this socket.

7.20.3.3 operator()() [1/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >::operator() (
            typename base_base_type::base_type & s )  [inline]
```

Binds this socket to the specified slave socket (hierarchical bind).

Note

When binding slave socket to slave socket, the socket of the parent must be bound to the socket of the child.

Parameters

s	tlm::tlm_base_target_socket_b slave socket to bind to this socket.

7.20.3.4 bind() [2/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >::bind (
            typename base_base_type::base_initiator_socket_type & s ) [inline], [virtual]
```

Binds this socket to the specified master socket.

Parameters

s	tlm::tlm_base_initiator_socket_b master socket to bind to this socket.

7.20.3.5 operator()() [2/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >::operator() (
            typename base_base_type::base_initiator_socket_type & s ) [inline]
```

Binds this socket to the specified master socket.

Parameters

s	tlm::tlm_base_initiator_socket_b master socket to bind to this socket.

7.20.3.6 bind() [3/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >::bind (
            base_slave_socket_type & s ) [inline], [virtual]
```

Binds this socket to the specified slave socket (hierarchical bind).

Note

When binding slave socket to slave socket, the socket of the parent must be bound to the socket of the child.

Parameters

s	amba_pv_base_slave_socket slave socket to bind to this socket.

7.20.3.7 operator()() [3/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >::operator() (
            base_slave_socket_type & s ) [inline]
```

Binds this socket to the specified slave socket (hierarchical bind).

Note

When binding slave socket to slave socket, the socket of the parent must be bound to the socket of the child.

Parameters

s	amba_pv_base_slave_socket slave socket to bind to this socket.

7.20.3.8 bind() [4/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >::bind (
            base_master_socket_type & s ) [inline], [virtual]
```

Binds this socket to the specified master socket.

Parameters

s	amba_pv_base_master_socket master socket to bind to this socket.

7.20.3.9 operator()() [4/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >::operator() (
            base_master_socket_type & s ) [inline]
```

Binds this socket to the specified master socket.

Parameters

s	amba_pv_base_master_socket master socket to bind to this socket.

7.20.3.10 bind() [5/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >::bind (
            amba_pv_fw_transport_if & iface ) [inline], [virtual]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_fw_transport_if interface to bind to this socket.

7.20.3.11 operator()() [5/5]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >::operator() (
            amba_pv_fw_transport_if & iface ) [inline]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_fw_transport_if interface to bind to this socket.

7.21 amba_pv::amba_pv_bw_snoop_if Class Reference

AMBA-PV core additional transaction interface for ACE.
`#include <core/amba_pv_core_ifs.h>`
Inherits sc_core::sc_interface.
Inherited by amba_pv::amba_pv_ace_bw_transport_if `[virtual]`.

Public Member Functions

- virtual void b_snoop (int socket_id, amba_pv_transaction &trans, sc_core::sc_time &t)=0

  Blocking snoop transport.
- virtual unsigned int snoop_dbg (int socket_id, amba_pv_transaction &trans)=0

  Debug snoop access to a target.

7.21.1 Detailed Description

AMBA-PV core additional transaction interface for ACE.
This is a tagged variant of the `tlm::tlm_fw_transport_if` interface. This interface is used for the backward snoop path.

Note

this interface is for internal use only. AMBA-PV ACE masters must implement the composite interface `amba_pv_bw_transport_and_snoop_if` interface.

7.21.2 Member Function Documentation

7.21.2.1 b_snoop()

```
virtual void amba_pv::amba_pv_bw_snoop_if::b_snoop (
            int socket_id,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [pure virtual]
```

Blocking snoop transport.

Parameters

socket_id	socket identifier (ignored on the master side).
trans	transaction.
t	timing annotation.

Implemented in amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >, amba_pv::amba_pv_ace_simple_probe< BUSWIDTH and amba_pv::amba_pv_ace_master_base.

7.21.2.2 snoop_dbg()

```
virtual unsigned int amba_pv::amba_pv_bw_snoop_if::snoop_dbg (
            int socket_id,
```

```
amba_pv_transaction & trans )  [pure virtual]
```

Debug snoop access to a target.
This use the same path as the b_snoop() interface. This debug access must be performed without any of the delays, waits, event notifications or side effects associated with a regular transaction. This debug access is, therefore, non-intrusive.

Parameters

socket_id	socket identifier (ignored on the master side).
trans	transaction.

Returns

number of bytes read or written or, if error, 0.

Implemented in amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >, amba_pv::amba_pv_ace_simple_probe< BUSWIDTH and amba_pv::amba_pv_ace_master_base.

7.22 amba_pv::amba_pv_bw_transport_if Class Reference

AMBA-PV core transaction interface.
`#include <core/amba_pv_core_ifs.h>`
Inherits sc_core::sc_interface.
Inherited by amba_pv::amba_pv_ace_bw_transport_if[virtual], amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, N amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >[virtual], amba_pv::amba_pv_from_tlm_bridge< BUSWIDTH >[virt amba_pv::amba_pv_master_base[virtual], amba_pv::amba_pv_protocol_checker< BUSWIDTH >[virtual], and amba_pv::amba_pv_simple_probe< BUSWIDTH >[virtual].

Public Member Functions

- virtual void invalidate_direct_mem_ptr (int socket_id, sc_dt::uint64 start_range, sc_dt::uint64 end_range)=0

  Invalidates DMI pointers previously established for the specified DMI region.

7.22.1 Detailed Description

AMBA-PV core transaction interface.
This is a tagged variant of the `tlm::tlm_bw_transport_if` interface. This interface is used for the backward path.

Note

AMBA-PV masters must implement the amba_pv_bw_transport_if interface.

7.22.2 Member Function Documentation

7.22.2.1 invalidate_direct_mem_ptr()

```
virtual void amba_pv::amba_pv_bw_transport_if::invalidate_direct_mem_ptr (
            int socket_id,
            sc_dt::uint64 start_range,
            sc_dt::uint64 end_range )  [pure virtual]
```

Invalidates DMI pointers previously established for the specified DMI region.

Parameters

socket_id	socket identifier (ignored on the slave side).

Parameters

start_range	DMI region start address.
end_range	DMI region end address.

Implemented in amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >, amba_pv::amba_pv_ace_simple_probe< BUSWIDTH amba_pv::amba_pv_from_tlm_bridge< BUSWIDTH >, amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >, amba_pv::amba_pv_protocol_checker< BUSWIDTH >, amba_pv::amba_pv_simple_probe< BUSWIDTH >, amba_pv::amba_pv_ace_master_base, and amba_pv::amba_pv_master_base.

7.23 amba_pv::ext::amba_pv_bw_transport_if Class Reference

AMBA-PV core transaction interface.
`#include <core/amba_pv_ext_core_ifs.h>`
Inherits sc_core::sc_interface.
Inherited by amba_pv::ext::amba_pv_master_base `[virtual]`.

Public Member Functions

- virtual void invalidate_direct_mem_ptr (int socket_id, sc_dt::uint64 start_range, sc_dt::uint64 end_range)=0

  Invalidates DMI pointers previously established for the specified DMI region.

7.23.1 Detailed Description

AMBA-PV core transaction interface.
This is a tagged variant of the `tlm::tlm_bw_transport_if` interface. This interface is used for the backward path.

Note

AMBA-PV masters must implement the amba_pv_bw_transport_if interface.

7.23.2 Member Function Documentation

7.23.2.1 invalidate_direct_mem_ptr()

```
virtual void amba_pv::ext::amba_pv_bw_transport_if::invalidate_direct_mem_ptr (
             int socket_id,
             sc_dt::uint64 start_range,
             sc_dt::uint64 end_range )  [pure virtual]
```

Invalidates DMI pointers previously established for the specified DMI region.

Parameters

socket_id	socket identifier (index into bound interfaces on the slave side).
start_range	DMI region start address.
end_range	DMI region end address.

Implemented in amba_pv::ext::amba_pv_master_base.

7.24 amba_pv::amba_pv_control Class Reference

Provides support for additional control information used by the AMBA buses.
`#include <bus/amba_pv_control.h>`

Inherits amba_pv::amba_pv_attributes.
Inherited by amba_pv::amba_pv_extension.

Public Member Functions

- amba_pv_control ()

  Default constructor.
- void set_id (unsigned int)

  Sets the ID for this transaction.
- unsigned int get_id () const

  Returns the ID for this transaction.
- void set_extended_id (uint64_t)

  Sets the ExtendedID for this transaction.
- uint64_t get_extended_id () const

  Returns the ExtendedID for this transaction.
- void set_privileged (bool=true)

  Sets whether this transaction is privileged or not.
- bool is_privileged () const

  Indicates whether or not this transaction is privileged.
- void set_non_secure (bool=true)

  Sets this transaction as secure or non-secure.
- bool is_non_secure () const

  Indicates whether or not this transaction is non-secure.
- void set_physical_address_space (amba_pv_physical_address_space_t)

  Sets Physical Address Space for this transaction.
- amba_pv_physical_address_space_t get_physical_address_space () const

  Returns Physical Address Space of this transaction, which is one of root, realm, secure or non-secure.
- void set_instruction (bool=true)

  Sets whether this transaction is an instruction or a data transaction.
- bool is_instruction () const

  Indicates whether this transaction is an instruction or a data transaction.
- void set_exclusive (bool=true)

  Sets whether or not this transaction is an exclusive transaction.
- bool is_exclusive () const

  Indicates whether or not this transaction is an exclusive transaction.
- void set_locked (bool=true)

  Sets whether or not this transaction is locked.
- bool is_locked () const

  Indicates whether or not this transaction is locked.
- void set_service_req_number (amba_pv_service_req_t)

  Sets the Service Request Number for this transaction.
- amba_pv_service_req_t get_service_req_number () const

  Returns the Service Request Number for this transaction.
- void set_address_based_routed (bool=true)

  Sets the whether this transaction is routed by address or not.
- bool is_address_based_routed () const

  Indicates whether the transaction is routed by address.
- void set_bufferable (bool=true)

  Sets whether or not this transaction is bufferable.
- bool is_bufferable () const

  Indicates whether or not this transaction is bufferable.

- void set_cacheable (bool=true)

  Sets whether or not this transaction is cacheable.
- bool is_cacheable () const

  Indicates whether or not this transaction is cacheable.
- void set_read_allocate (bool=true)

  Sets the Allocate signal for read transactions.
- bool is_read_allocate () const

  Gets the status of the Allocate signal for read transactions.
- void set_write_allocate (bool=true)

  Sets the Allocate signal for write transactions.
- bool is_write_allocate () const

  Gets the status of the Allocate signal for write transactions.
- void set_modifiable (bool=true)

  Sets whether or not this transaction is modifiable (AXI4).
- bool is_modifiable () const

  Indicates whether or not this transaction is modifiable (AXI4).
- void set_read_other_allocate (bool=true)

  Sets the Other Allocate signal for read transactions.
- bool is_read_other_allocate () const

  Gets the status of the Other Allocate signal for read transactions.
- void set_write_other_allocate (bool=true)

  Sets the Other Allocate signal for write transactions.
- bool is_write_other_allocate () const

  Gets the status of the Other Allocate signal for write transactions.
- void set_gathering (bool=true)

  Sets whether or not this transaction is of type gathering.
- bool is_gathering () const

  Indicates whether or not this transaction is gathering.
- void set_reordering (bool=true)

  Sets whether or not this transaction is reordering.
- bool is_reordering () const

  Indicates whether or not this transaction is reordering.
- void set_transient (bool=true)

  Sets whether or not this transaction is transient.
- bool is_transient () const

  Indicates whether or not this transaction is transient.
- void set_translated_access (bool)

  Sets the TranslatedAccess attribute.
- bool is_translated_access () const

  Gets the TranslatedAccess attribute.
- void set_mmu_flow_type (amba_pv_mmuflow_t)

  Sets the MMU flow type for untranslated addresses.
- amba_pv_mmuflow_t get_mmu_flow_type () const

  Gets the MMU flow type for untranslated addresses.
- void set_qos (unsigned int)

  Sets the QOS bits for this transaction.
- unsigned int get_qos () const

  Returns the QOS bits for this transaction.
- void set_region (unsigned int)

  Sets the REGION bits for this transaction.
- unsigned int get_region () const

Returns the REGION bits for this transaction.

- void set_user (unsigned int)

  Sets the USER bits for this transaction.
- unsigned int get_user () const

  Returns the USER bits for this transaction.
- void set_snoop (amba_pv_snoop_t)

  Sets the snoop type for shareable transactions.
- amba_pv_snoop_t get_snoop () const

  Returns the snoop type for shareable transactions.
- void set_domain (amba_pv_domain_t)

  Sets the shareablility domain for this transactions.
- amba_pv_domain_t get_domain () const

  Returns the shareablility domain for this transactions.
- void set_bar (amba_pv_bar_t)

  Sets the barrier type for this transaction.
- amba_pv_bar_t get_bar () const

  Returns the barrier type for this transaction.
- void reset ()

  Resets all members to their default value.

7.24.1 Detailed Description

Provides support for additional control information used by the AMBA buses.
The additional control information provided by the AMBA buses includes:

- system-level caching and buffering control
- secure and privileged access
- atomic operations, using exclusive or locked accesses
- ARMv8 extentions
- quality of service indication
- multiple region support
- coherency support
- barrier transactions.

This class is used as a base class for the AMBA-PV extension type (amba_pv_extension).

See also

amba_pv_extension

7.24.2 Constructor & Destructor Documentation

7.24.2.1 amba_pv_control()

```
amba_pv::amba_pv_control::amba_pv_control ( )  [inline]
```

Default constructor.

7.24.3 Member Function Documentation

7.24.3.1 set_id()

```
void amba_pv::amba_pv_control::set_id (
            unsigned int id )  [inline]
```

Sets the ID for this transaction.
This is the identification tag for this transaction. It is mainly used for:

- exclusive access
- out-of-order transaction processing (not relevant at PV level)

This ID is set by the master originating the transaction. The interconnect must modify the transfer ID to ensure its uniqueness accross all its masters before passing the transaction to the addressed slave.

Parameters

id	transfer ID.

See also

get_id()

7.24.3.2 get_id()

```
unsigned int amba_pv::amba_pv_control::get_id ( ) const  [inline]
```

Returns the ID for this transaction.

See also

set_id()

7.24.3.3 set_extended_id()

```
void amba_pv::amba_pv_control::set_extended_id (
            uint64_t id )  [inline]
```

Sets the ExtendedID for this transaction.

Parameters

id	ExtendedID.

See also

get_extended_id()

7.24.3.4 get_extended_id()

```
uint64_t amba_pv::amba_pv_control::get_extended_id ( ) const  [inline]
```

Returns the ExtendedID for this transaction.

See also

set_extended_id()

7.24.3.5 set_privileged()

```
void amba_pv::amba_pv_control::set_privileged (
            bool privileged = true )  [inline]
```

Sets whether this transaction is privileged or not.
This enables masters to indicate their processing mode. A privileged transaction typically has a greater level of access within the system.

Parameters

privileged	`true` for a privileged transaction, `false` otherwise.

See also

is_privileged()

7.24.3.6 is_privileged()

```
bool amba_pv::amba_pv_control::is_privileged ( ) const  [inline]
```

Indicates whether or not this transaction is privileged.

Returns

`true` for a privileged transaction, `false` otherwise.

See also

set_privileged()

7.24.3.7 set_non_secure()

```
void amba_pv::amba_pv_control::set_non_secure (
            bool non_secure = true )  [inline]
```

Sets this transaction as secure or non-secure.
This enables differentiating between secure and non-secure transactions.

Parameters

non_secure	`true` for a non-secure transaction, `false` otherwise.

Note

This flag is initialized to `false`, so that the transaction is considered as secure by default.

See also

is_non_secure()

7.24.3.8 is_non_secure()

```
bool amba_pv::amba_pv_control::is_non_secure ( ) const  [inline]
```

Indicates whether or not this transaction is non-secure.

Returns

`true` for a non-secure transaction, `false` otherwise.

See also

set_non_secure()

7.24.3.9 set_physical_address_space()

```
void amba_pv::amba_pv_control::set_physical_address_space (
            amba_pv_physical_address_space_t physical_address_space ) [inline]
```

Sets Physical Address Space for this transaction.
This enables differentiating between root, realm, secure and non-secure transactions.

Parameters

physical_address_space	which is one of root, realm, secure or non-secure.

See also

get_physical_address_space()

7.24.3.10 get_physical_address_space()

```
amba_pv_physical_address_space_t amba_pv::amba_pv_control::get_physical_address_space ( )
const [inline]
```

Returns Physical Address Space of this transaction, which is one of root, realm, secure or non-secure.

See also

set_physical_address_space()

7.24.3.11 set_instruction()

```
void amba_pv::amba_pv_control::set_instruction (
            bool instruction = true ) [inline]
```

Sets whether this transaction is an instruction or a data transaction.

Parameters

instruction	`true` for an instruction transaction, `false` otherwise.

Note

This flag is initialized to `false`, so that the transaction is marked as a data transaction unless it is specifically known to be an instruction transaction.

See also

is_instruction()

7.24.3.12 is_instruction()

```
bool amba_pv::amba_pv_control::is_instruction ( ) const [inline]
```

Indicates whether this transaction is an instruction or a data transaction.

Returns

`true` for an instruction transaction, `false` otherwise.

See also

set_instruction()

7.24.3.13 set_exclusive()

```
void amba_pv::amba_pv_control::set_exclusive (
            bool exclusive = true )  [inline]
```

Sets whether or not this transaction is an exclusive transaction.

Parameters

exclusive	`true` for an exclusive transaction, `false` otherwise.

Note

This flag must not be `true` if is_locked() returned `true`.

See also

is_exclusive(), is_locked()

7.24.3.14 is_exclusive()

```
bool amba_pv::amba_pv_control::is_exclusive ( ) const  [inline]
```

Indicates whether or not this transaction is an exclusive transaction.

Returns

`true` for an exclusive transaction, `false` otherwise.

See also

set_exclusive()

7.24.3.15 set_locked()

```
void amba_pv::amba_pv_control::set_locked (
            bool locked = true )  [inline]
```

Sets whether or not this transaction is locked.

Parameters

locked	`true` for a locked transaction, `false` otherwise.

Note

Locked transactions require that the interconnect prevents any other transactions occurring while the locked sequence is in progress and can therefore have an impact on the interconnect performance. It is recommended that locked accesses are only used to support legacy devices.

This flag must not be `true` if is_exclusive() returned `true`.

See also

is_locked(), is_exclusive()

7.24.3.16 is_locked()

```
bool amba_pv::amba_pv_control::is_locked ( ) const  [inline]
```

Indicates whether or not this transaction is locked.

Returns

`true` for a locked transaction, `false` otherwise.

See also

set_locked()

7.24.3.17 set_service_req_number()

```
void amba_pv::amba_pv_control::set_service_req_number (
            amba_pv_service_req_t service_req )  [inline]
```

Sets the Service Request Number for this transaction.

Parameters

service	request number.

See also

get_service_req_number()

7.24.3.18 get_service_req_number()

```
amba_pv_service_req_t amba_pv::amba_pv_control::get_service_req_number ( ) const  [inline]
```

Returns the Service Request Number for this transaction.

See also

set_service_req_number()

7.24.3.19 set_address_based_routed()

```
void amba_pv::amba_pv_control::set_address_based_routed (
            bool address_based_routed = true )  [inline]
```

Sets the whether this transaction is routed by address or not.

Parameters

address_based_routed	`true` if the device access is to be routed based on address, `false` otherwise.

Note

This is a model only feature and does not map to anything in the AMBA specification.

See also

is_address_based_routed()

7.24.3.20 is_address_based_routed()

```
bool amba_pv::amba_pv_control::is_address_based_routed ( ) const  [inline]
```

Indicates whether the transaction is routed by address.

Returns

`true` for address based routed, `false` otherwise.

See also

set_address_based_routed()

7.24.3.21 set_bufferable()

```
void amba_pv::amba_pv_control::set_bufferable (
            bool bufferable = true )  [inline]
```

Sets whether or not this transaction is bufferable.
A bufferable transaction can be delayed in reaching its final destination. This is usually only relevant to writes. For ARMv8 architectures this is the 'early write acknowledge' attribute.

Parameters

bufferable	`true` for a bufferable transaction, `false` otherwise.

See also

is_bufferable()

7.24.3.22 is_bufferable()

```
bool amba_pv::amba_pv_control::is_bufferable ( ) const  [inline]
```

Indicates whether or not this transaction is bufferable.

Returns

`true` for a bufferable transaction, `false` otherwise.

See also

set_bufferable()

7.24.3.23 set_cacheable()

```
void amba_pv::amba_pv_control::set_cacheable (
            bool cacheable = true )  [inline]
```

Sets whether or not this transaction is cacheable.

For writes, a number of different writes can be merged together. For reads, a location can be pre-fetched or can be fetched just once for multiple read transactions. To determine if a transaction must be cached, use this flag with the set_read_allocate() and set_write_allocate() flags.

Parameters

cacheable	`true` for a cacheable transaction, `false` otherwise.

See also

is_cacheable(), set_read_allocate(), set_write_allocate(), set_modifiable()

7.24.3.24 is_cacheable()

```
bool amba_pv::amba_pv_control::is_cacheable ( ) const  [inline]
```

Indicates whether or not this transaction is cacheable.

Returns

`true` for a cacheable transaction, `false` otherwise.

See also

set_cacheable(), is_modifiable()

7.24.3.25 set_read_allocate()

```
void amba_pv::amba_pv_control::set_read_allocate (
            bool allocate = true )  [inline]
```

Sets the Allocate signal for read transactions.
When asserted the transaction must be looked up in a cache because it could have been previously allocated. If asserted it is also recommended that the transaction is allocated in the cache for performance reasons.

Parameters

allocate	`true` to assert, `false` to deassert

Note

This flag must not be `true` if is_modifiable() returned `false`.

See also

is_read_allocate(), set_read_other_allocate()

7.24.3.26 is_read_allocate()

```
bool amba_pv::amba_pv_control::is_read_allocate ( ) const  [inline]
```

Gets the status of the Allocate signal for read transactions.
When asserted the transaction must be looked up in a cache because it could have been previously allocated. If asserted it is also recommended that the transaction is allocated in the cache for performance reasons.

Returns

`true` the signal is asserted, `false` deasserted

See also

set_read_allocate(), is_read_other_allocate()

7.24.3.27 set_write_allocate()

```
void amba_pv::amba_pv_control::set_write_allocate (
            bool allocate = true )  [inline]
```

Sets the Allocate signal for write transactions.
When asserted the transaction must be looked up in a cache because it could have been previously allocated. If asserted it is also recommended that the transaction is allocated in the cache for performance reasons.

Parameters

allocate	`true` to assert, `false` to deassert

Note

This flag must not be `true` if is_modifiable() returned `false`.

See also

is_write_allocate(), set_write_other_allocate()

7.24.3.28 is_write_allocate()

```
bool amba_pv::amba_pv_control::is_write_allocate ( ) const  [inline]
```

Gets the status of the Allocate signal for write transactions.
When asserted the transaction must be looked up in a cache because it could have been previously allocated. If asserted it is also recommended that the transaction is allocated in the cache for performance reasons.

Returns

`true` the signal is asserted, `false` deasserted

See also

set_write_allocate(), is_write_other_allocate()

7.24.3.29 set_modifiable()

```
void amba_pv::amba_pv_control::set_modifiable (
            bool modifiable = true )  [inline]
```

Sets whether or not this transaction is modifiable (AXI4).
If a transaction is modifiable it can be broken into multiple transactions, and multiple transactions can be merged into a single transaction. A read transaction can fetch more data than required. But the exclusivity and protection attributes cannot be modified.

Note

The modifiable flag is the same as the cacheable flag but has been renamed for AXI4 to better describe the required functionality.

Parameters

modifiable	`true` for a cacheable transaction, `false` otherwise.

See also

is_modifiable(), set_read_allocate(), set_write_allocate(), set_cacheable()

7.24.3.30 is_modifiable()

```
bool amba_pv::amba_pv_control::is_modifiable ( ) const  [inline]
```

Indicates whether or not this transaction is modifiable (AXI4).

Note

The modifiable flag is the same as the cacheable flag but has been renamed for AXI4 to better describe the required functionality.

Returns

`true` for a modifiable transaction, `false` otherwise.

See also

set_modifiable(), is_cacheable()

7.24.3.31 set_read_other_allocate()

```
void amba_pv::amba_pv_control::set_read_other_allocate (
            bool other_allocate = true )  [inline]
```

Sets the Other Allocate signal for read transactions.
When asserted the transaction must be looked up in a cache because it could have been previously allocated by a write transaction or a transaction from another master.

Parameters

other_allocate	`true` to assert, `false` to deassert

Note

This flag must not be `true` if is_modifiable() returned `false`.

See also

set_read_allocate(), is_read_other_allocate()

7.24.3.32 is_read_other_allocate()

```
bool amba_pv::amba_pv_control::is_read_other_allocate ( ) const  [inline]
```

Gets the status of the Other Allocate signal for read transactions.
When asserted the transaction must be looked up in a cache because it could have been previously allocated by a write transaction or or a transaction from another master.

Returns

`true` the signal is asserted, `false` deasserted

See also

set_read_other_allocate(), is_read_allocate()

7.24.3.33 set_write_other_allocate()

```
void amba_pv::amba_pv_control::set_write_other_allocate (
            bool other_allocate = true )  [inline]
```

Sets the Other Allocate signal for write transactions.
When asserted the transaction must be looked up in a cache because it could have been previously allocated by a read transaction or or a transaction from another master.

Parameters

other_allocate	`true` to assert, `false` to deassert

Note

This flag must not be `true` if is_modifiable() returned `false`.

See also

is_write_other_allocate(), set_write_allocate()

7.24.3.34 is_write_other_allocate()

```
bool amba_pv::amba_pv_control::is_write_other_allocate ( ) const  [inline]
```

Gets the status of the Other Allocate signal for write transactions.
When asserted the transaction must be looked up in a cache because it could have been previously allocated by a read transaction or or a transaction from another master.

Returns

`true` the signal is asserted, `false` deasserted

See also

set_write_other_allocate(), is_write_allocate()

7.24.3.35 set_gathering()

```
void amba_pv::amba_pv_control::set_gathering (
            bool gathering = true )  [inline]
```

Sets whether or not this transaction is of type gathering.
A gathering transaction may be merged with other transactions of the same read/write type to a single transaction.

Parameters

gathering	`true` for a gathering transaction, `false` otherwise.

See also

is_gathering()

7.24.3.36 is_gathering()

```
bool amba_pv::amba_pv_control::is_gathering ( ) const  [inline]
```

Indicates whether or not this transaction is gathering.

Returns

`true` for a gathering transaction, `false` otherwise.

See also

set_gathering()

7.24.3.37 set_reordering()

```
void amba_pv::amba_pv_control::set_reordering (
            bool reordering = true )  [inline]
```

Sets whether or not this transaction is reordering.
A reordering transaction may be reordered with respect to other reordering transactions.

Parameters

reordering	`true` for a reordering transaction, `false` otherwise.

See also

is_reordering()

7.24.3.38 is_reordering()

```
bool amba_pv::amba_pv_control::is_reordering ( ) const  [inline]
```

Indicates whether or not this transaction is reordering.

Returns

`true` for a reordering transaction, `false` otherwise.

See also

set_reordering()

7.24.3.39 set_transient()

```
void amba_pv::amba_pv_control::set_transient (
            bool transient = true )  [inline]
```

Sets whether or not this transaction is transient.
A transient transaction can be delayed in reaching its final destination. This is usually only relevant to writes.

Parameters

transient	`true` for a transient transaction, `false` otherwise.

See also

is_transient()

7.24.3.40 is_transient()

```
bool amba_pv::amba_pv_control::is_transient ( ) const  [inline]
```

Indicates whether or not this transaction is transient.

Returns

`true` for a transient transaction, `false` otherwise.

See also

set_transient()

7.24.3.41 set_translated_access()

```
void amba_pv::amba_pv_control::set_translated_access (
            bool translated_access ) [inline]
```

Sets the TranslatedAccess attribute.

Parameters

translated_access	`true` if the device access has been already translated by a local TLB, `false` otherwise.

See also

is_translated_access()

7.24.3.42 is_translated_access()

```
bool amba_pv::amba_pv_control::is_translated_access ( ) const [inline]
```

Gets the TranslatedAccess attribute.

Returns

`true` for TranslatedAccess, `false` otherwise.

See also

set_translated_access()

7.24.3.43 set_mmu_flow_type()

```
void amba_pv::amba_pv_control::set_mmu_flow_type (
            amba_pv_mmuflow_t mmu_flow_type ) [inline]
```

Sets the MMU flow type for untranslated addresses.

Parameters

mmu_flow_type	as defined for AxMMUFLOW encodings (refer AMBA AXI specification)

See also

get_mmu_flow_type()

7.24.3.44 get_mmu_flow_type()

```
amba_pv_mmuflow_t amba_pv::amba_pv_control::get_mmu_flow_type ( ) const [inline]
```

Gets the MMU flow type for untranslated addresses.

Returns

mmu_flow_type as defined for AxMMUFLOW encodings (refer AMBA AXI specification)

See also

set_mmu_flow_type()

7.24.3.45 set_qos()

```
void amba_pv::amba_pv_control::set_qos (
            unsigned int qos )  [inline]
```

Sets the QOS bits for this transaction.

Parameters

qos	QOS bits

See also

get_qos()

7.24.3.46 get_qos()

```
unsigned int amba_pv::amba_pv_control::get_qos ( ) const  [inline]
```

Returns the QOS bits for this transaction.

See also

set_qos()

7.24.3.47 set_region()

```
void amba_pv::amba_pv_control::set_region (
            unsigned int region )  [inline]
```

Sets the REGION bits for this transaction.

Parameters

region	REGION bits

See also

get_region()

7.24.3.48 get_region()

```
unsigned int amba_pv::amba_pv_control::get_region ( ) const  [inline]
```

Returns the REGION bits for this transaction.

See also

set_region()

7.24.3.49 set_user()

```
void amba_pv::amba_pv_control::set_user (
            unsigned int user )  [inline]
```

Sets the USER bits for this transaction.

Parameters

user	USER bits

See also

get_user()

7.24.3.50 get_user()

```
unsigned int amba_pv::amba_pv_control::get_user ( ) const  [inline]
```

Returns the USER bits for this transaction.

See also

set_user()

7.24.3.51 set_snoop()

```
void amba_pv::amba_pv_control::set_snoop (
            amba_pv_snoop_t snoop )  [inline]
```

Sets the snoop type for shareable transactions.

Parameters

snoop	transaction type for shareable transactions.

See also

get_snoop()

7.24.3.52 get_snoop()

```
amba_pv_snoop_t amba_pv::amba_pv_control::get_snoop ( ) const  [inline]
```

Returns the snoop type for shareable transactions.

See also

set_snoop()

7.24.3.53 set_domain()

```
void amba_pv::amba_pv_control::set_domain (
            amba_pv_domain_t domain )  [inline]
```

Sets the shareablility domain for this transactions.

Parameters

domain	shareability domain.

See also

get_domain()

7.24.3.54 get_domain()

```
amba_pv_domain_t amba_pv::amba_pv_control::get_domain ( ) const  [inline]
```

Returns the shareablility domain for this transactions.

See also

set_domain()

7.24.3.55 set_bar()

```
void amba_pv::amba_pv_control::set_bar (
            amba_pv_bar_t bar )  [inline]
```

Sets the barrier type for this transaction.

Parameters

bar	barrier type.

See also

get_bar()

7.24.3.56 get_bar()

```
amba_pv_bar_t amba_pv::amba_pv_control::get_bar ( ) const  [inline]
```

Returns the barrier type for this transaction.

See also

set_bar()

7.24.3.57 reset()

```
void amba_pv::amba_pv_control::reset ( )  [inline]
```

Resets all members to their default value.

7.25 amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES > Class Template Reference

AMBA-PV bus decoder model.

`#include <models/amba_pv_decoder.h>`

Inherits amba_pv::amba_pv_fw_transport_if, amba_pv::amba_pv_bw_transport_if, and sc_core::sc_module.

Public Member Functions

- amba_pv_decoder (const sc_core::sc_module_name &)

  Constructor.
- amba_pv_decoder (const sc_core::sc_module_name &, const std::string &)

  Parameterized constructor.
- virtual const char * kind () const

  Returns the kind string of this decoder.
- void bind (int, base_slave_socket_type &, const sc_dt::uint64 &, const sc_dt::uint64 &, bool=false)

  Binds the specified slave socket to the master socket of this decoder at the specified index.
- void operator() (int, base_slave_socket_type &, const sc_dt::uint64 &, const sc_dt::uint64 &, bool=false)

  Binds the specified slave socket to the master socket of this decoder at the specified index.
- unsigned int get_id_shift () const

  Returns the transaction ID shift value.
- void set_id_shift (unsigned int)

  Sets the transaction ID shift value.
- std::string get_map_file () const

  Returns the address map file.
- void set_map_file (const std::string &)

  Sets the address map file.
- amba_pv_address_map get_address_map () const

  Returns the address map of this decoder.
- void set_address_map (const amba_pv_address_map &)

  Sets the address map of this decoder.
- amba_pv_address_map get_default_address_map () const

  Returns the default address map of this decoder.
- void set_default_address_map (const amba_pv_address_map &)

  Sets the default address map of this decoder.
- void load_address_map (const std::string &)

  Loads the address map of this decoder from the specified file.
- void load_address_map (std::istream &)

  Loads the address map of this decoder from the specified stream.
- void print_address_map (const std::string &) const

  Prints the address map of this decoder.
- void print_address_map (std::ostream &) const

  Prints the address map of this decoder.
- void set_verbose (bool=true)

  Sets the verbosity of this decoder.

Data Fields

- amba_pv_socket_array< slave_socket_type > amba_pv_s

  Slaves socket array.
- amba_pv_socket_array< master_socket_type > amba_pv_m

  Masters socket array.

Protected Member Functions

- virtual void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking transport.
- virtual unsigned int transport_dbg (int, amba_pv_transaction &)

  Debug access to a target.
- virtual bool get_direct_mem_ptr (int, amba_pv_transaction &, tlm::tlm_dmi &)

  Requests a DMI access based on the specified transaction.
- virtual void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)

  Invalidates DMI pointers previously established for the specified DMI region.

7.25.1 Detailed Description

template<unsigned int BUSWIDTH = 64, int NUMMASTERS = 1, int NUMSLAVES = 1>
class amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >

AMBA-PV bus decoder model.
Each master is bound to an *amba_pv_s*[...] slave socket of the decoder, and each of the *amba_pv_m*[...] master sockets belonging to the decoder is bound to a socket belonging to a different slave. Each master-to-slave socket connection is point-to-point. The numbers of slave and master sockets of the decoder are specified using template arguments.
This decoder routes transactions through to the appropriate slave depending on the transaction address, translating the address to local address for each slave as it does so. The same address translation applies also to DMI and debug transactions.

Note

amba_pv_decoder does not currently support locked transactions (see amba_pv_control::set_locked()). Any locked transaction will be handled as if it were not locked.

The names of the master sockets follows the scheme `"amba_pv_m%d"`, and the name for the slave sockets follows the scheme `"amba_pv_s%d"`, where `d` is the socket index.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.
NUMMASTERS	number of masters connected to this decoder. Defaults to 1.
NUMSLAVES	number of slaves connected to this decoder. Defaults to 1.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 amba_pv_decoder() [1/2]

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::amba_pv_decoder (
            const sc_core::sc_module_name & name )  [inline], [explicit]
```

Constructor.

Parameters

name	module name.

7.25.2.2 amba_pv_decoder() [2/2]

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::amba_pv_decoder (
            const sc_core::sc_module_name & name,
            const std::string & file )  [inline]
```

Parameterized constructor.

Parameters

name	module name.
file	file from which the address map of this decoder is loaded.

Note

The use of an address map file relies on the `sc_core::sc_find_object()` method to find the slave sockets bound to the master sockets of this decoder. OSCI TLM 2.0, 9 Jun 2008, contains bugs in the convenience sockets, as their names are computed by `sc_core::sc_gen_unique_name()`. It is recommended to use the bind() methods rather than such map file when binding to OSCI TLM 2.0 convenience sockets. The OSCI TLM 2.0.1 release, 15 Jul 2009, fixes this.

See also

bind()

7.25.3 Member Function Documentation

7.25.3.1 kind()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
const char * amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::kind  [inline],
[virtual]
```

Returns the kind string of this decoder.

7.25.3.2 bind()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
void amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::bind (
            int index,
            base_slave_socket_type & s,
            const sc_dt::uint64 & start,
            const sc_dt::uint64 & end,
            bool default_map = false )  [inline]
```

Binds the specified slave socket to the master socket of this decoder at the specified *index*.

Parameters

index	master socket index.
s	slave socket to bind to the master socket.
start	start address of the memory region associated to the slave socket *s*.
end	end address of this region.
default_map	`true` to add the memory region to the default address map instead; default to `false`.

See also

operator()()

7.25.3.3 operator()()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
void amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::operator() (
            int index,
            base_slave_socket_type & s,
            const sc_dt::uint64 & start,
            const sc_dt::uint64 & end,
            bool default_map = false )  [inline]
```

Binds the specified slave socket to the master socket of this decoder at the specified *index*.

Parameters

index	master socket index.
s	slave socket to bind to the master socket.
start	start address of the memory region associated to the slave socket *s*.
end	end address of this region.
default_map	`true` to add the memory region to the default address map instead; default to `false`.

See also

bind()

7.25.3.4 get_id_shift()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
unsigned int amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::get_id_shift  [inline]
```

Returns the transaction ID shift value.

This method returns the shift value used in updating the transaction ID, to ensure its uniqueness, before forwarding the transaction throught to the addressed slave. The transaction ID shift value is initialised by default to the following:

```
log(NUMMASTERS) / log(2.0)
```

See also

set_id_shift(), amba_pv_control

7.25.3.5 set_id_shift()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
void amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::set_id_shift (
            unsigned int id_shift )  [inline]
```

Sets the transaction ID shift value.

This method sets the shift value used in updating the transaction ID, to ensure its uniqueness, before forwarding the transaction throught to the addressed slave. When the shift value is set to zero, the transaction ID will not be updated.

See also

get_id_shift(), amba_pv_control

7.25.3.6 get_map_file()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
std::string amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::get_map_file  [inline]
```

Returns the address map file.

See also

set_map_file()

7.25.3.7 set_map_file()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
void amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::set_map_file (
            const std::string & file )  [inline]
```

Sets the address map file.

Parameters

file	file from which the address map of this decoder is loaded.

Note

The use of an address map file relies on the `sc_core::sc_find_object()` method to find the slave sockets bound to the master sockets of this decoder. OSCI TLM 2.0, 9 Jun 2008, contains bugs in the convenience sockets, as their names are computed by `sc_core::sc_gen_unique_name()`. It is recommended to use the bind() methods rather than such map file when binding to OSCI TLM 2.0 convenience sockets. The OSCI TLM 2.0.1 release, 15 Jul 2009, fixes this.

See also

get_map_file()

7.25.3.8 get_address_map()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
amba_pv_address_map amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::get_address↩
_map  [inline]
```

Returns the address map of this decoder.

See also

set_address_map()

7.25.3.9 set_address_map()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
void amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::set_address_map (
            const amba_pv_address_map & map )  [inline]
```

Sets the address map of this decoder.

Note

Setting the address map is not possible if the simulation is running.

Parameters

map	new address map.

See also

get_address_map()

7.25.3.10 get_default_address_map()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
amba_pv_address_map amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::get_default↩
_address_map  [inline]
```

Returns the default address map of this decoder.

See also

set_default_address_map()

7.25.3.11 set_default_address_map()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
void amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::set_default_address_map (
            const amba_pv_address_map & map )  [inline]
```

Sets the default address map of this decoder.

Note

Setting the default address map is not possible if the simulation is running.

Parameters

map	new default address map.

See also

get_default_address_map()

7.25.3.12 load_address_map() [1/2]

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
void amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::load_address_map (
            const std::string & file )  [inline]
```

Loads the address map of this decoder from the specified file.

Note

Loading the address map is not possible if the simulation is running.

Parameters

file	file from which the address map of this decoder is loaded.

7.25.3.13 load_address_map() [2/2]

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
void amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::load_address_map (
            std::istream & is )  [inline]
```

Loads the address map of this decoder from the specified stream.

Note

Loading the address map is not possible if the simulation is running.

Parameters

is	stream from which load the address map of this decoder is loaded.

7.25.3.14 print_address_map() [1/2]

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
void amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::print_address_map (
            const std::string & file ) const  [inline]
```

Prints the address map of this decoder.

Parameters

file	file to which the address map of this decoder is printed.

7.25.3.15 print_address_map() [2/2]

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
void amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::print_address_map (
            std::ostream & os ) const  [inline]
```

Prints the address map of this decoder.

Parameters

os	stream to which the address map of this decoder is printed.

7.25.3.16 set_verbose()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
void amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::set_verbose (
            bool verbose = true )  [inline]
```

Sets the verbosity of this decoder.
The set_verbose() method turns on or off the display of "`decode error at address...`" warning messages.

Parameters

verbose	`true` (default) to turn on the display of warning messages, `false` otherwise.

7.25.3.17 b_transport()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
void amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::b_transport (
            int socket_id,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Blocking transport.
This version of the method decodes the address of the specified transaction and forwards it through the corresponding master socket to the addressed slave.
Implements amba_pv::amba_pv_fw_transport_if.

7.25.3.18 transport_dbg()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
unsigned int amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::transport_dbg (
            int socket_id,
            amba_pv_transaction & trans )  [inline], [protected], [virtual]
```

Debug access to a target.
This version of the method decodes the address of the specified transaction and forwards it through the corresponding master socket to the addressed slave.
Implements amba_pv::amba_pv_fw_transport_if.

7.25.3.19 get_direct_mem_ptr()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
bool amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::get_direct_mem_ptr (
            int socket_id,
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [inline], [protected], [virtual]
```

Requests a DMI access based on the specified transaction.
This version of the method decodes the address of the specified transaction and forwards it through the corresponding master socket to the addressed slave. On return, the address range of the DMI descriptor is adjusted to the slave mapped range.
Implements amba_pv::amba_pv_fw_transport_if.

7.25.3.20 invalidate_direct_mem_ptr()

```
template<unsigned int BUSWIDTH, int NUMMASTERS, int NUMSLAVES>
void amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >::invalidate_direct_mem_ptr (
            int socket_id,
            sc_dt::uint64 start_range,
            sc_dt::uint64 end_range )  [inline], [protected], [virtual]
```

Invalidates DMI pointers previously established for the specified DMI region.
This version of the method adjusts the address range of the DMI descriptor to the slave mapped range and broadcasts the invalidate_direct_mem_ptr() call through the slave sockets to all masters.
Implements amba_pv::amba_pv_bw_transport_if.

7.25.4 Field Documentation

7.25.4.1 amba_pv_s

```
template<unsigned int BUSWIDTH = 64, int NUMMASTERS = 1, int NUMSLAVES = 1>
amba_pv_socket_array<slave_socket_type> amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES
>::amba_pv_s
```

Slaves socket array.

7.25.4.2 amba_pv_m

```
template<unsigned int BUSWIDTH = 64, int NUMMASTERS = 1, int NUMSLAVES = 1>
amba_pv_socket_array<master_socket_type> amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS,
NUMSLAVES >::amba_pv_m
```

Masters socket array.

7.26 amba_pv::amba_pv_dvm Class Reference

Provides DVM message information used by the AMBA ACE buses.

`#include <bus/amba_pv_dvm.h>`

Inherited by amba_pv::amba_pv_extension.

Public Member Functions

- amba_pv_dvm ()

  Default constructor.
- void set_dvm_encoded_transaction (sc_dt::uint64)

  Set the encoded DVM transaction.
- sc_dt::uint64 get_dvm_encoded_transaction () const

  Return the encoded DVM transaction.
- void set_dvm_encoded_additional_transaction (sc_dt::uint64)

  Set the encoded additional DVM transaction.
- sc_dt::uint64 get_dvm_encoded_additional_transaction () const

  Return the encoded additional DVM transaction.
- bool has_dvm_additional_transaction () const

  Indicate whether there is an additional transaction for this DVM message.
- void set_dvm_address (sc_dt::uint64)

  Set the DVM address.
- sc_dt::uint64 get_dvm_address () const

  Return the DVM address.
- void set_dvm_vmid (unsigned int)

  Set the VMID for this DVM transaction.
- bool is_dvm_vmid_set () const

  Indicate whether there is a VMID set for this DVM transaction.
- unsigned int get_dvm_vmid () const

  Return the VMID for this DVM transaction.
- void set_dvm_asid (unsigned int)

  Set the ASID for this DVM transaction.
- bool is_dvm_asid_set () const

  Indicate whether there is an ASID set for this DVM transaction.
- unsigned int get_dvm_asid () const

  Return the ASID for this DVM transaction.
- void set_dvm_virtual_index (unsigned int)

  Set the Virtual Index for this DVM transaction.

- bool is_dvm_virtual_index_set () const

  Indicate whether there is a virtual index set for this DVM transaction.
- unsigned int get_dvm_virtual_index () const

  Return the virtual index for this DVM transaction.
- void set_dvm_message_type (amba_pv_dvm_message_t)

  Set the message type for this DVM transaction.
- amba_pv_dvm_message_t get_dvm_message_type () const

  Return the message type for this DVM transaction.
- void set_dvm_os (amba_pv_dvm_os_t)

  Set the OS type for this DVM transaction.
- amba_pv_dvm_os_t get_dvm_os () const

  Return the OS type for this DVM transaction.
- void set_dvm_security (amba_pv_dvm_security_t)

  Set the security type for this DVM transaction.
- amba_pv_dvm_security_t get_dvm_security () const

  Return the security for this DVM transaction.
- void set_dvm_tlb_leaf (bool)

  Set Leaf Entry only invalidation for this DVM transaction.
- bool is_dvm_tlb_leaf_set () const

  Indicate whether Leaf Entry only invalidation is set for this DVM transaction.
- void set_dvm_stage (amba_pv_dvm_stage_t)

  Set the Staged Invalidation for this DVM transaction.
- amba_pv_dvm_stage_t get_dvm_stage () const

  Return the Staged Invalidation for this DVM transaction.
- void reset ()

  Reset DVM message to default value.
- void set_dvm_transaction (unsigned int)

  Set the encoded DVM transaction.
- unsigned int get_dvm_transaction () const

  Return the encoded DVM transaction.
- void set_dvm_additional_address (sc_dt::uint64)

  Set the DVM additional address for this transaction.
- bool is_dvm_additional_address_set () const

  Indicate whether there is an additional address for this DVM transaction.
- sc_dt::uint64 get_dvm_additional_address () const

  Return the DVM additional address for this transaction.

7.26.1 Detailed Description

Provides DVM message information used by the AMBA ACE buses.
This class is used as a base class for the AMBA-PV extension type (amba_pv_extension).

See also

amba_pv_extension

7.26.2 Constructor & Destructor Documentation

7.26.2.1 amba_pv_dvm()

```
amba_pv::amba_pv_dvm::amba_pv_dvm ( )  [inline]
```

Default constructor.

7.26.3 Member Function Documentation

7.26.3.1 set_dvm_encoded_transaction()

```
void amba_pv::amba_pv_dvm::set_dvm_encoded_transaction (
            sc_dt::uint64 dvm_transaction ) [inline]
```

Set the encoded DVM transaction.
Set the DVM transaction for this DVM message, as encoded on AxADDR signal.

Parameters

dvm_transaction	DVM transaction, as encoded on AxADDR signal

See also

get_dvm_encoded_transaction(), get_dvm_encoded_additional_transaction(), set_dvm_encoded_additional_transaction(), has_dvm_additional_transaction()

7.26.3.2 get_dvm_encoded_transaction()

```
sc_dt::uint64 amba_pv::amba_pv_dvm::get_dvm_encoded_transaction ( ) const [inline]
```

Return the encoded DVM transaction.
Return the DVM transaction for this DVM message, as encoded on AxADDR signal.

See also

set_dvm_encoded_transaction(), get_dvm_encoded_additional_transaction(), set_dvm_encoded_additional_transaction(), has_dvm_additional_transaction()

7.26.3.3 set_dvm_encoded_additional_transaction()

```
void amba_pv::amba_pv_dvm::set_dvm_encoded_additional_transaction (
            sc_dt::uint64 encoded_additional_transaction ) [inline]
```

Set the encoded additional DVM transaction.
Set the additional DVM transaction for this DVM message, as encoded on AxADDR signal.
Please note that this method does not alter the first DVM transaction for this DVM message: in particular, its LSB (indicating that there is an additional transaction) is not set. set_dvm_encoded_transaction() must be called with the appropriate value, instead.

Parameters

encoded_additional_transaction	additional DVM transaction, as encoded on AxADDR signal

See also

get_dvm_encoded_additional_transaction(), has_dvm_additional_transaction(), get_dvm_encoded_transaction(), set_dvm_encoded_transaction()

7.26.3.4 get_dvm_encoded_additional_transaction()

```
sc_dt::uint64 amba_pv::amba_pv_dvm::get_dvm_encoded_additional_transaction ( ) const [inline]
```

Return the encoded additional DVM transaction.
Return the additional DVM transaction for this DVM message, as encoded on AxADDR signal.

See also

set_dvm_encoded_additional_transaction(), has_dvm_additional_transaction(), get_dvm_encoded_transaction(), set_dvm_encoded_transaction()

7.26.3.5 has_dvm_additional_transaction()

```
bool amba_pv::amba_pv_dvm::has_dvm_additional_transaction ( ) const  [inline]
```

Indicate whether there is an additional transaction for this DVM message.

See also

get_dvm_encoded_transaction(), set_dvm_encoded_transaction(), get_dvm_encoded_additional_transaction(), set_dvm_encoded_additional_transaction()

7.26.3.6 set_dvm_address()

```
void amba_pv::amba_pv_dvm::set_dvm_address (
            sc_dt::uint64 address )  [inline]
```

Set the DVM address.

Parameters

address	DVM address

See also

get_dvm_address()

7.26.3.7 get_dvm_address()

```
sc_dt::uint64 amba_pv::amba_pv_dvm::get_dvm_address ( ) const  [inline]
```

Return the DVM address.

See also

set_dvm_address()

7.26.3.8 set_dvm_vmid()

```
void amba_pv::amba_pv_dvm::set_dvm_vmid (
            unsigned int vmid )  [inline]
```

Set the VMID for this DVM transaction.

Parameters

vmid	Virtual Machine IDentifier (VMID) [0-255]

See also

get_dvm_vmid(), is_dvm_vmid_set()

7.26.3.9 is_dvm_vmid_set()

```
bool amba_pv::amba_pv_dvm::is_dvm_vmid_set ( ) const  [inline]
```

Indicate whether there is a VMID set for this DVM transaction.

See also

get_dvm_vmid(), set_dvm_vmid()

7.26.3.10 get_dvm_vmid()

```
unsigned int amba_pv::amba_pv_dvm::get_dvm_vmid ( ) const  [inline]
```

Return the VMID for this DVM transaction.

See also

set_dvm_vmid(), is_dvm_vmid_set()

7.26.3.11 set_dvm_asid()

```
void amba_pv::amba_pv_dvm::set_dvm_asid (
            unsigned int asid )  [inline]
```

Set the ASID for this DVM transaction.

Parameters

asid	Address Space IDentifier (ASID) [0-65535]

See also

get_dvm_asid(), is_dvm_asid_set()

7.26.3.12 is_dvm_asid_set()

```
bool amba_pv::amba_pv_dvm::is_dvm_asid_set ( ) const  [inline]
```

Indicate whether there is an ASID set for this DVM transaction.

See also

get_dvm_asid(), set_dvm_asid()

7.26.3.13 get_dvm_asid()

```
unsigned int amba_pv::amba_pv_dvm::get_dvm_asid ( ) const  [inline]
```

Return the ASID for this DVM transaction.

See also

set_dvm_asid(), is_dvm_asid_set()

7.26.3.14 set_dvm_virtual_index()

```
void amba_pv::amba_pv_dvm::set_dvm_virtual_index (
            unsigned int virtual_index )  [inline]
```

Set the Virtual Index for this DVM transaction.

Parameters

virtual_index	Virtual index [0-0xFFFF]

See also

get_dvm_virtual_index(), is_dvm_virtual_index_set()

7.26.3.15 is_dvm_virtual_index_set()

```
bool amba_pv::amba_pv_dvm::is_dvm_virtual_index_set ( ) const  [inline]
```

Indicate whether there is a virtual index set for this DVM transaction.

See also

get_dvm_virtual_index(), set_dvm_virtual_index()

7.26.3.16 get_dvm_virtual_index()

```
unsigned int amba_pv::amba_pv_dvm::get_dvm_virtual_index ( ) const  [inline]
```

Return the virtual index for this DVM transaction.

See also

set_dvm_virtual_index(), is_dvm_virtual_index_set()

7.26.3.17 set_dvm_message_type()

```
void amba_pv::amba_pv_dvm::set_dvm_message_type (
            amba_pv_dvm_message_t message_type )  [inline]
```

Set the message type for this DVM transaction.

Parameters

message_type	DVM message type

See also

get_dvm_message_type()

7.26.3.18 get_dvm_message_type()

```
amba_pv_dvm_message_t amba_pv::amba_pv_dvm::get_dvm_message_type ( ) const  [inline]
```

Return the message type for this DVM transaction.

See also

set_dvm_message_type()

7.26.3.19 set_dvm_os()

```
void amba_pv::amba_pv_dvm::set_dvm_os (
            amba_pv_dvm_os_t os ) [inline]
```

Set the OS type for this DVM transaction.

Parameters

os	guest OS or hypervisor type

See also

get_dvm_os()

7.26.3.20 get_dvm_os()

```
amba_pv_dvm_os_t amba_pv::amba_pv_dvm::get_dvm_os ( ) const [inline]
```

Return the OS type for this DVM transaction.

See also

set_dvm_os()

7.26.3.21 set_dvm_security()

```
void amba_pv::amba_pv_dvm::set_dvm_security (
            amba_pv_dvm_security_t security ) [inline]
```

Set the security type for this DVM transaction.

Parameters

security	DVM security type

See also

get_dvm_security()

7.26.3.22 get_dvm_security()

```
amba_pv_dvm_security_t amba_pv::amba_pv_dvm::get_dvm_security ( ) const [inline]
```

Return the security for this DVM transaction.

See also

set_dvm_security()

7.26.3.23 set_dvm_tlb_leaf()

```
void amba_pv::amba_pv_dvm::set_dvm_tlb_leaf (
            bool leaf_entry_only ) [inline]
```

Set Leaf Entry only invalidation for this DVM transaction.

Parameters

leaf_entry_only	Leaf Entry only invalidation

See also

is_dvm_tlb_leaf_set()

7.26.3.24 is_dvm_tlb_leaf_set()

```
bool amba_pv::amba_pv_dvm::is_dvm_tlb_leaf_set ( ) const  [inline]
```

Indicate whether Leaf Entry only invalidation is set for this DVM transaction.

See also

set_dvm_tlb_leaf()

7.26.3.25 set_dvm_stage()

```
void amba_pv::amba_pv_dvm::set_dvm_stage (
            amba_pv_dvm_stage_t stage )  [inline]
```

Set the Staged Invalidation for this DVM transaction.

Parameters

stage	DVM Staged Invalidation

See also

get_dvm_stage()

7.26.3.26 get_dvm_stage()

```
amba_pv_dvm_stage_t amba_pv::amba_pv_dvm::get_dvm_stage ( ) const  [inline]
```

Return the Staged Invalidation for this DVM transaction.

See also

set_dvm_stage()

7.26.3.27 reset()

```
void amba_pv::amba_pv_dvm::reset ( )  [inline]
```

Reset DVM message to default value.

7.26.3.28 set_dvm_transaction()

```
void amba_pv::amba_pv_dvm::set_dvm_transaction (
            unsigned int dvm_transaction )  [inline]
```

Set the encoded DVM transaction.

Parameters

dvm_transaction	DVM transaction, as encoded on AxADDR signal

See also

set_dvm_encoded_transaction()

7.26.3.29 get_dvm_transaction()

```
unsigned int amba_pv::amba_pv_dvm::get_dvm_transaction ( ) const  [inline]
```

Return the encoded DVM transaction.

See also

get_dvm_encoded_transaction()

7.26.3.30 set_dvm_additional_address()

```
void amba_pv::amba_pv_dvm::set_dvm_additional_address (
            sc_dt::uint64 additional_address )  [inline]
```

Set the DVM additional address for this transaction.

See also

set_dvm_encoded_additional_transaction(), set_dvm_address()

7.26.3.31 is_dvm_additional_address_set()

```
bool amba_pv::amba_pv_dvm::is_dvm_additional_address_set ( ) const  [inline]
```

Indicate whether there is an additional address for this DVM transaction.

See also

has_dvm_additional_transaction()

7.26.3.32 get_dvm_additional_address()

```
sc_dt::uint64 amba_pv::amba_pv_dvm::get_dvm_additional_address ( ) const  [inline]
```

Return the DVM additional address for this transaction.

See also

get_dvm_encoded_additional_transaction(), get_dvm_address()

7.27 amba_pv::amba_pv_exclusive_monitor< BUSWIDTH > Class Template Reference

AMBA-PV exclusive monitor model.

`#include <models/amba_pv_exclusive_monitor.h>`

Inherits amba_pv::amba_pv_fw_transport_if, amba_pv::amba_pv_bw_transport_if, and sc_core::sc_module.

Public Member Functions

- amba_pv_exclusive_monitor (const sc_core::sc_module_name &)

  Constructor.
- amba_pv_exclusive_monitor (const sc_core::sc_module_name &, unsigned int, bool=false)

  Parameterized constructor.
- virtual const char * kind () const

  Returns the kind string of this monitor.
- unsigned int get_erg () const

  Returns the ERG of this monitor.
- void set_erg (unsigned int)

  Sets the ERG of this monitor.
- bool is_dmi_enabled () const

  Returns whether or not DMI is enabled for non exclusive regions for this monitor.
- void set_dmi_enabled (bool=true)

  Sets whether or not DMI is enabled for non exclusive regions for this monitor.
- bool non_exclusive_writes_ignored () const

  Returns whether or not a non-exclusive write to a monitored exclusive region by the same master should revoke the monitor.
- void ignore_non_exclusive_writes (bool=true)

  Sets whether or not a non-exclusive write to a monitored exclusive region by the same master should revoke the monitor.
- bool must_exclusive_accesses_match () const

  Returns whether or not exclusive write adress must match the preceeding exclusive read address from the same master.
- void exclusive_accesses_must_match (bool=true)

  Sets whether or not an exclusive write address must match the preceeding exclusive read address from the same master.
- amba_pv_domain_t get_domain () const

  Returns the shareability domain of this monitor.
- void set_domain (amba_pv_domain_t=AMBA_PV_SYSTEM)

  Sets the shareability domain of this monitor.

Data Fields

- amba_pv_slave_socket< BUSWIDTH > amba_pv_s

  Slave socket.
- amba_pv_master_socket< BUSWIDTH > amba_pv_m

  Master socket.
- signal_master_port< bool, 0, sc_core::SC_ZERO_OR_MORE_BOUND > clr_ex_mon_out

  Global exclusive monitor clear event.

Protected Member Functions

- virtual void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking transport.
- virtual unsigned int transport_dbg (int, amba_pv_transaction &)

  Debug access to a target.
- virtual bool get_direct_mem_ptr (int, amba_pv_transaction &, tlm::tlm_dmi &)

  Requests a DMI access based on the specified transaction.
- virtual void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)

  Invalidates DMI pointers previously established for the given DMI region.

7.27.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >

AMBA-PV exclusive monitor model.
The amba_pv_exclusive_monitor model provides exclusive access support, and can be added before any AMBA-PV slave.
Exclusive accesses are *single aligned transfers*, for which amba_pv_control::is_exclusive() returns `true`.
The amba_pv_exclusive_monitor model can be configured to:

- either disable DMI (default behavior) and thus reject all DMI requests
- or enable DMI for non exclusive regions and thus reject DMI requests that intersects with exclusive regions and invalidate DMI pointers for the same, unless non-exclusive writes are ignored
- ignore non-exclusive write by the same master, that is to keep the monitor in exclusive state or not
- to pass on exclusives that are outside of its domain.

Note

When DMI is enabled, the onus is on the master to not use DMI for exclusive accesses. If the master uses DMI for exclusive accesses, wrong behavior might be observed.

When configured to disable DMI, the amba_pv_exclusive_monitor model might have an impact on performance.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 amba_pv_exclusive_monitor() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::amba_pv_exclusive_monitor (
            const sc_core::sc_module_name & name )  [inline], [explicit]
```

Constructor.
The *Exclusives Reservation Granule* (ERG) is set to: min(7u, log2((*BUSWIDTH* + 7) / 8)).

Parameters

name	monitor name.

7.27.2.2 amba_pv_exclusive_monitor() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::amba_pv_exclusive_monitor (
            const sc_core::sc_module_name & name,
            unsigned int erg,
            bool dmi_enabled = false )  [inline]
```

Parameterized constructor.

Parameters

name	monitor name.
erg	the ERG of this monitor.
dmi_enabled	`true` to enable DMI for non exclusive regions Defaults to `false`.

7.27.3 Member Function Documentation

7.27.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string of this monitor.

7.27.3.2 get_erg()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::get_erg  [inline]
```

Returns the ERG of this monitor.

7.27.3.3 set_erg()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::set_erg (
            unsigned int erg )  [inline]
```

Sets the ERG of this monitor.

Note

Setting the ERG is not possible if the simulation is running.

Parameters

erg	the new ERG.

7.27.3.4 is_dmi_enabled()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::is_dmi_enabled  [inline]
```

Returns whether or not DMI is enabled for non exclusive regions for this monitor.

Returns

`true` when DMI is enabled, `false` otherwise.

7.27.3.5 set_dmi_enabled()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::set_dmi_enabled (
            bool dmi_enabled = true )  [inline]
```

Sets whether or not DMI is enabled for non exclusive regions for this monitor.
When DMI is enabled, the monitor rejects DMI requests that intersect with exclusive regions and invalidates DMI pointers for the same.
When DMI is disabled, DMI is not allowed.

Note

When DMI is disabled during simulation, DMI pointers previously acquired are not invalidated.

When DMI is enabled, the onus is on the master to not use DMI for exclusive accesses. If the master uses DMI for exclusive accesses, wrong behavior might be observed.

Parameters

dmi_enabled	`true` to enable DMI, `false` otherwise.

See also

`tlm::tlm_generic_payload::set_dmi_allowed()` in the *TLM 2.0 Language Reference Manual* for more information on DMI allowed.

7.27.3.6 non_exclusive_writes_ignored()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::non_exclusive_writes_ignored  [inline]
```

Returns whether or not a non-exclusive write to a monitored exclusive region by the same master should revoke the monitor.
This behaviour is `ImpDef` in the ARM ARM.

Returns

`true` when non-exclusive writes are ignored, `false` otherwise.

7.27.3.7 ignore_non_exclusive_writes()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::ignore_non_exclusive_writes (
            bool ignore_non_exclusive_writes = true )  [inline]
```

Sets whether or not a non-exclusive write to a monitored exclusive region by the same master should revoke the monitor.
This behaviour is `ImpDef` in the ARM ARM.

Parameters

ignore_non_exclusive_writes	`true` to ignore non-exclusive writes, `false` otherwise.

7.27.3.8 must_exclusive_accesses_match()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::must_exclusive_accesses_match  [inline]
```

Returns whether or not exclusive write adress must match the preceeding exclusive read address from the same master.
This behaviour is `ImpDef` in the ARM ARM.

Returns

`true` when exclusive accesses must match, `false` otherwise.

7.27.3.9 exclusive_accesses_must_match()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::exclusive_accesses_must_match (
            bool must_match = true )  [inline]
```

Sets whether or not an exclusive write address must match the preceeding exclusive read address from the same master.
This behaviour is `ImpDef` in the ARM ARM.

Parameters

must_match	`true` when exclusive access address must match, `false` otherwise.

7.27.3.10 get_domain()

```
template<unsigned int BUSWIDTH>
amba_pv_domain_t amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::get_domain  [inline]
```

Returns the shareability domain of this monitor.

7.27.3.11 set_domain()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::set_domain (
            amba_pv_domain_t domain = AMBA_PV_SYSTEM )  [inline]
```

Sets the shareability domain of this monitor.
The monitor will simply pass on exclusives that are outside of its domain.

Note

Setting the shareability domain is not possible if the simulation is running.

Parameters

domain	the new shareability domain of interest, default `AMBA_PV_SYSTEM`, all exclusive transactions are handled.

7.27.3.12 b_transport()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::b_transport (
            int socket_id,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Blocking transport.
This version of the method completes the specified transaction while processing exclusive access. amba_pv_↩ extension::get_rsp() indicates the successful completion of the exclusive access, or an error occured. The response `AMBA_PV_EXOKAY` is returned to indicate a successful completion of the exclusive access, while `AMBA_PV_↩ OKAY` is returned in case of failure.
Implements amba_pv::amba_pv_fw_transport_if.

7.27.3.13 transport_dbg()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::transport_dbg (
            int socket_id,
            amba_pv_transaction & trans )  [inline], [protected], [virtual]
```

Debug access to a target.
This version of the method forwards this debug access to the slave.
Implements amba_pv::amba_pv_fw_transport_if.

7.27.3.14 get_direct_mem_ptr()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::get_direct_mem_ptr (
            int socket_id,
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi )  [inline], [protected], [virtual]
```

Requests a DMI access based on the specified transaction.
This version of the method returns `false` when DMI is disabled, `false` for exclusive accesses, or when the DMI region intersects with an exclusive region and the DMI region cannot be resized outside of the exclusive region. It returns `true` when DMI access is granted.

Note

Read-only DMI is allowed though as a potential performance optimization.

Implements amba_pv::amba_pv_fw_transport_if.

7.27.3.15 invalidate_direct_mem_ptr()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::invalidate_direct_mem_ptr (
            int socket_id,
            sc_dt::uint64 start_range,
            sc_dt::uint64 end_range )  [inline], [protected], [virtual]
```

Invalidates DMI pointers previously established for the given DMI region.
This version of the method simply forwards the call backward to the master.
Implements amba_pv::amba_pv_bw_transport_if.

7.27.4 Field Documentation

7.27.4.1 amba_pv_s

```
template<unsigned int BUSWIDTH = 64>
amba_pv_slave_socket<BUSWIDTH> amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::amba_pv_s
```

Slave socket.

7.27.4.2 amba_pv_m

```
template<unsigned int BUSWIDTH = 64>
amba_pv_master_socket<BUSWIDTH> amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >::amba_pv_m
```

Master socket.

7.27.4.3 clr_ex_mon_out

```
template<unsigned int BUSWIDTH = 64>
signal_master_port<bool, 0, sc_core::SC_ZERO_OR_MORE_BOUND> amba_pv::amba_pv_exclusive_monitor<
BUSWIDTH >::clr_ex_mon_out
```

Global exclusive monitor clear event.

7.28 amba_pv::amba_pv_extension Class Reference

AMBA-PV extension class.

`#include <bus/amba_pv_extension.h>`

Inherits tlm::tlm_extension< amba_pv_extension >, amba_pv::amba_pv_control, amba_pv::amba_pv_dvm, and amba_pv::amba_pv_atomic.

Public Member Functions

- amba_pv_extension ()

  Default constructor.
- amba_pv_extension (int, const amba_pv_control *)

  Constructor for read and write transactions.
- amba_pv_extension (unsigned int, int, const amba_pv_control *, amba_pv_burst_t)

  Constructor for burst read and write transactions.
- virtual tlm::tlm_extension_base * clone () const

  Returns a copy of this extension.
- virtual void copy_from (tlm::tlm_extension_base const &)

  Copy this extension from a specified one.
- void set_length (unsigned int)

  Sets the number of data transfers that occur within this burst.
- unsigned int get_length () const

  Returns the number of data transfers that occur within this burst.
- void set_size (int)

  Sets the maximum number of data bytes to transfer in each beat, or data transfer, within a burst.
- unsigned int get_size () const

  Returns the maximum number of data bytes to transfer in each beat, or data transfer, within a burst.
- void set_burst (amba_pv_burst_t)

  Specifies the burst type.
- amba_pv_burst_t get_burst () const

  Returns the burst type.
- void set_resp (amba_pv_resp_t)

  Sets this transaction response.
- amba_pv_resp_t get_resp () const

  Returns this transaction response.
- bool is_incomplete () const

  Returns wether or not the response is incomplete.
- void set_incomplete ()

  Sets the response to incomplete.
- bool is_okay () const

  Returns wether or not the `OKAY` response is set.
- void set_okay ()

  Sets the `OKAY` response.
- bool is_exokay () const

  Returns wether or not the response is `EXOKAY`.

- void set_exokay ()

  Sets the `EXOKAY` response.
- bool is_slverr () const

  Returns wether or not the response is `SLVERR`.
- void set_slverr ()

  Sets the `SLVERR` response.
- bool is_decerr () const

  Returns wether or not the response is `DECERR`.
- void set_decerr ()

  Sets the `DECERR` response.
- bool is_pass_dirty () const

  Returns wether or not the `PassDirty` response bit is set.
- void set_pass_dirty (bool=true)

  Sets the `PassDirty` response bit.
- bool is_shared () const

  Returns wether or not the `IsShared` response bit is set.
- void set_shared (bool=true)

  Sets the `IsShared` response bit.
- bool is_snoop_data_transfer () const

  Returns wether or not the `DataTransfer` snoop response bit is set.
- void set_snoop_data_transfer (bool=true)

  Sets the `DataTransfer` snoop response bit.
- bool is_snoop_error () const

  Returns wether or not the `Error` snoop response bit is set.
- void set_snoop_error (bool=true)

  Sets the `Error` snoop response bit.
- bool is_snoop_was_unique () const

  Returns wether or not the `WasUnique` snoop response bit is set.
- void set_snoop_was_unique (bool=true)

  Sets the `WasUnique` snoop response bit.
- amba_pv_response * get_response ()

  Get combined response.
- void set_response_array_ptr (amba_pv_response *)

  Set response array pointer.
- amba_pv_response * get_response_array_ptr ()

  Get response array pointer.
- void set_response_array_complete (bool=true)

  Indicate whether the response array has been completed.
- bool is_response_array_complete () const

  Indicates whether a slave has used the response array to return a response.
- void reset ()

  Resets members of this AMBA-PV extension to their default value.
- void reset (int, const amba_pv_control *)

  Resets members of this AMBA-PV extension for read and write transactions.
- void reset (unsigned int, int, const amba_pv_control *, amba_pv_burst_t)

  Resets members of this AMBA-PV extension for burst read and write transactions.

7.28.1 Detailed Description

AMBA-PV extension class.
The amba_pv_extension class extends `tlm::tlm_extension` by providing support for AMBA 4 buses specific addressing options and additional control information.
The addressing options provided by the AMBA 4 buses include:

- burst length, from 1 to 256 data transfers per burst
- burst transfer size of 8-1024 bits
- wrapping, incrementing, and non-incrementing burst types.

The additional control information provided by the AMBA 4 buses includes:

- system-level caching and buffering control
- secure and privileged access
- atomic operations, using exclusive, locked accesses, or atomic transactions.

The AMBA-PV extension must be used with AMBA-PV sockets, that is, sockets parameterized with the amba_pv_protocol_types traits class. This follows the rules set out in the section "Define a new protocol traits class containg a typedef for tlm_generic_payload" of the *TLM 2.0 Language Reference Manual*. The AMBA-PV extension is a mandatory extension for the modelling of AMBA 4 buses. For more information, see the section "Non-ignorable amd mandatory extensions", *TLM 2.0 Language Reference Manual*.

See also

amba_pv_master_socket, amba_pv_slave_socket

7.28.2 Constructor & Destructor Documentation

7.28.2.1 amba_pv_extension() [1/3]

```
amba_pv::amba_pv_extension::amba_pv_extension ( )  [inline]
```

Default constructor.
By default:

- the burst length is initialized to 1
- the burst size is initialized to 8
- the burst type is initialized to `AMBA_PV_INCR`
- the response is initialized to `AMBA_PV_INCOMPLETE`.

7.28.2.2 amba_pv_extension() [2/3]

```
amba_pv::amba_pv_extension::amba_pv_extension (
            int size,
            const amba_pv_control * ctrl )  [inline]
```

Constructor for read and write transactions.

Parameters

size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. An instance with size 0 is considered to be an invalid extension
ctrl	optional AMBA 4 control information (set to `NULL` if unused).

7.28.2.3 amba_pv_extension() [3/3]

```
amba_pv::amba_pv_extension::amba_pv_extension (
            unsigned int length,
            int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst )  [inline]
```

Constructor for burst read and write transactions.

Parameters

length	transaction burst length as in [1-256].
size	transaction burst size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. An instance with size 0 is considered to be an invalid extension
ctrl	optional AMBA 4 control information (set to NULL if unused).
burst	transaction burst type as one of AMBA_PV_INCR, AMBA_PV_FIXED, AMBA_PV_WRAP.

7.28.3 Member Function Documentation

7.28.3.1 clone()

```
tlm::tlm_extension_base * amba_pv::amba_pv_extension::clone ( ) const  [inline], [virtual]
```

Returns a copy of this extension.

7.28.3.2 copy_from()

```
void amba_pv::amba_pv_extension::copy_from (
            tlm::tlm_extension_base const & ext )  [inline], [virtual]
```

Copy this extension from a specified one.
copy_from() assumes that the specified extension *ext* is of the same type as this one.

Parameters

ext	extention to copy from.

7.28.3.3 set_length()

```
void amba_pv::amba_pv_extension::set_length (
            unsigned int length )  [inline]
```

Sets the number of data transfers that occur within this burst.
Each burst can be between 1 and 256 transfers long.

Parameters

length	number of data transfers as in [1-256].

See also

get_length()

7.28.3.4 get_length()

```
unsigned int amba_pv::amba_pv_extension::get_length ( ) const  [inline]
```

Returns the number of data transfers that occur within this burst.

See also

set_length()

7.28.3.5 set_size()

```
void amba_pv::amba_pv_extension::set_size (
            int size )  [inline]
```

Sets the maximum number of data bytes to transfer in each beat, or data transfer, within a burst.

Parameters

size	bytes in each transfer as one of [1, 2, 4, 8, 16, 32, 64, 128]. An instance with size 0 is considered to be an invalid extension

See also

get_size()

7.28.3.6 get_size()

```
unsigned int amba_pv::amba_pv_extension::get_size ( ) const  [inline]
```

Returns the maximum number of data bytes to transfer in each beat, or data transfer, within a burst.

See also

set_size()

7.28.3.7 set_burst()

```
void amba_pv::amba_pv_extension::set_burst (
            amba_pv_burst_t burst )  [inline]
```

Specifies the burst type.

Parameters

burst	burst type as one of AMBA_PV_FIXED, AMBA_PV_INCR, or AMBA_PV_WRAP.

See also

get_burst()

7.28.3.8 get_burst()

```
amba_pv_burst_t amba_pv::amba_pv_extension::get_burst ( ) const  [inline]
```

Returns the burst type.

See also

set_burst()

7.28.3.9 set_resp()

```
void amba_pv::amba_pv_extension::set_resp (
            amba_pv_resp_t resp ) [inline]
```

Sets this transaction response.

Parameters

resp	transaction response.

See also

get_resp(), set_incomplete(), set_okay(), set_exokay(), set_slverr(), set_decerr(), set_pass_dirty(), set_shared()

7.28.3.10 get_resp()

```
amba_pv_resp_t amba_pv::amba_pv_extension::get_resp ( ) const [inline]
```

Returns this transaction response.

See also

set_resp(), is_incomplete(), is_okay(), is_exokay(), is_slverr(), is_decerr(), is_pass_dirty(), is_shared()

7.28.3.11 is_incomplete()

```
bool amba_pv::amba_pv_extension::is_incomplete ( ) const [inline]
```

Returns wether or not the response is incomplete.
An incomplete reponse indicates that the slave did not attempt to perform the access.

Returns

`true` if the response is incomplete, `false` otherwise.

See also

set_incomplete()

7.28.3.12 set_incomplete()

```
void amba_pv::amba_pv_extension::set_incomplete ( ) [inline]
```

Sets the response to incomplete.

See also

is_incomplete()

7.28.3.13 is_okay()

```
bool amba_pv::amba_pv_extension::is_okay ( ) const  [inline]
```

Returns wether or not the OKAY response is set.
The OKAY reponse indicates if a normal access has been successful. It indicates also an exclusive access failure.

Returns

true if the OKAY response is set, false otherwise.

See also

set_okay()

7.28.3.14 set_okay()

```
void amba_pv::amba_pv_extension::set_okay ( )  [inline]
```

Sets the OKAY response.

See also

is_okay(), set_pass_dirty(), set_shared()

7.28.3.15 is_exokay()

```
bool amba_pv::amba_pv_extension::is_exokay ( ) const  [inline]
```

Returns wether or not the response is EXOKAY.
If true, the EXOKAY response indicates that either the read or write portion of an exclusive access has been successful.

Returns

true if the response is EXOKAY, false otherwise.

See also

set_exokay(), is_pass_dirty(), is_shared()

7.28.3.16 set_exokay()

```
void amba_pv::amba_pv_extension::set_exokay ( )  [inline]
```

Sets the EXOKAY response.
The PassDirty and IsShared response flags will be cleared.

See also

is_exokay(), set_pass_dirty(), set_shared()

7.28.3.17 is_slverr()

```
bool amba_pv::amba_pv_extension::is_slverr ( ) const  [inline]
```

Returns wether or not the response is SLVERR.
The SLVERR response is used if the access has reached the slave successfully, but the slave returned an error condition to the originating master.

Returns

true if the response is SLVERR, false otherwise.

See also

set_slverr()

7.28.3.18 set_slverr()

```
void amba_pv::amba_pv_extension::set_slverr ( )  [inline]
```

Sets the SLVERR response.

See also

is_slverr()

7.28.3.19 is_decerr()

```
bool amba_pv::amba_pv_extension::is_decerr ( ) const  [inline]
```

Returns wether or not the response is DECERR.
The DECERR response is generated typically by an interconnect component to indicate that there is no slave at the transaction address.

Returns

true if the response is DECERR, false otherwise.

See also

set_decerr()

7.28.3.20 set_decerr()

```
void amba_pv::amba_pv_extension::set_decerr ( )  [inline]
```

Sets the DECERR response.

See also

is_decerr()

7.28.3.21 is_pass_dirty()

```
bool amba_pv::amba_pv_extension::is_pass_dirty ( ) const  [inline]
```

Returns wether or not the PassDirty response bit is set.
The PassDirty response bit indicates the cache line is dirty with respect to main memory. For AMBA-PV ACE this bit is a part of both read and snoop responses.

Returns

true if the PassDirty bit is set, false otherwise.

See also

set_pass_dirty(), set_okay(), set_exokay()

7.28.3.22 set_pass_dirty()

```
void amba_pv::amba_pv_extension::set_pass_dirty (
            bool pass_dirty = true )  [inline]
```

Sets the PassDirty response bit.
The PassDirty response bit indicates the cache line is dirty with respect to main memory. For AMBA-PV ACE this bit is a part of both read and snoop responses.

Parameters

pass_dirty	status of `PassDirty` bit

See also

is_pass_dirty(), is_okay(), is_exokay()

7.28.3.23 is_shared()

```
bool amba_pv::amba_pv_extension::is_shared ( ) const  [inline]
```

Returns wether or not the `IsShared` response bit is set.
The `IsShared` response bit hints that another copy of the data might be held in another cache. For AMBA-PV ACE this bit is a part of both read and snoop responses.

Returns

`true` if the `IsShared` bit is set, `false` otherwise.

See also

set_shared(), set_okay(), set_exokay()

7.28.3.24 set_shared()

```
void amba_pv::amba_pv_extension::set_shared (
            bool is_shared = true )  [inline]
```

Sets the `IsShared` response bit.
The `IsShared` response bit hints that another copy of the data might be held in another cache. For AMBA-PV ACE this bit is a part of both read and snoop responses.

Parameters

is_shared	status of `IsShared` bit

See also

is_shared(), is_okay(), is_exokay()

7.28.3.25 is_snoop_data_transfer()

```
bool amba_pv::amba_pv_extension::is_snoop_data_transfer ( ) const  [inline]
```

Returns wether or not the `DataTransfer` snoop response bit is set.
The `DataTransfer` response bit indicates that a full cache line of data will be provided on the snoop data channel for this transaction.

Returns

`true` if the `DataTransfer` bit is set, `false` otherwise.

See also

set_snoop_data_transfer()

7.28.3.26 set_snoop_data_transfer()

```
void amba_pv::amba_pv_extension::set_snoop_data_transfer (
            bool data_transfer = true )  [inline]
```

Sets the `DataTransfer` snoop response bit.
The `DataTransfer` response bit indicates that a full cache line of data will be provided on the snoop data channel for this transaction.

Parameters

data_transfer	status of `DataTransfer` bit

See also

is_snoop_data_transfer()

7.28.3.27 is_snoop_error()

```
bool amba_pv::amba_pv_extension::is_snoop_error ( ) const  [inline]
```

Returns wether or not the `Error` snoop response bit is set.
The `Error` response bit indicates that the snooped cache line is in error.

Returns

`true` if the `Error` bit is set, `false` otherwise.

See also

set_snoop_error()

7.28.3.28 set_snoop_error()

```
void amba_pv::amba_pv_extension::set_snoop_error (
            bool error = true )  [inline]
```

Sets the `Error` snoop response bit.
The `Error` response bit indicates that the snooped cache line is in error.

Parameters

error	status of `Error` bit

See also

is_snoop_error()

7.28.3.29 is_snoop_was_unique()

```
bool amba_pv::amba_pv_extension::is_snoop_was_unique ( ) const  [inline]
```

Returns wether or not the `WasUnique` snoop response bit is set.
The `WasUnique` bit indicates that the cache line was held in a Unique state before the snoop.

Returns

`true` if the `WasUnique` bit is set, `false` otherwise.

See also

set_snoop_was_unique()

7.28.3.30 set_snoop_was_unique()

```
void amba_pv::amba_pv_extension::set_snoop_was_unique (
            bool was_unique = true )  [inline]
```

Sets the WasUnique snoop response bit.
The WasUnique bit indicates that the cache line was held in a Unique state before the snoop.

Parameters

was_unique	status of WasUnique bit

See also

is_snoop_was_unique()

7.28.3.31 get_response()

```
amba_pv_response * amba_pv::amba_pv_extension::get_response ( )  [inline]
```

Get combined response.
Get a pointer to the transaction response object.

See also

get_response_array_ptr()

7.28.3.32 set_response_array_ptr()

```
void amba_pv::amba_pv_extension::set_response_array_ptr (
            amba_pv_response * response_array_ptr )  [inline]
```

Set response array pointer.
Used by masters to supply a response array for burst transaction responses. The size of the array must match the burst length set by set_length(). Use of the response array by a slave is optional.

Parameters

response_array_ptr	response array pointer

See also

get_response_array_ptr(), set_length()

7.28.3.33 get_response_array_ptr()

```
amba_pv_response * amba_pv::amba_pv_extension::get_response_array_ptr ( )  [inline]
```

Get response array pointer.
Get the response array pointer for burst transactions. If the response array pointer is not null, then the size of the array will match the burst length returned by get_length(). Use of the response array by a slave is optional, but if a slave does use the response array then the slave should set_response_array_complete() when all the array elements have been assigned.

See also

set_response_array_ptr(), get_length(), set_response_array_complete()

7.28.3.34 set_response_array_complete()

```
void amba_pv::amba_pv_extension::set_response_array_complete (
            bool complete = true )  [inline]
```

Indicate whether the response array has been completed.
A slave should use this method when all the response array elements have been assigned.

Parameters

complete	`true` response array has been completed

See also

get_response_array_complete(), get_response_array_ptr()

7.28.3.35 is_response_array_complete()

```
bool amba_pv::amba_pv_extension::is_response_array_complete ( ) const  [inline]
```

Indicates whether a slave has used the response array to return a response.
A master should use this method to determine the correct source for a transaction response. If this method returns `true` then the get_resp() method should not be used to determine the transaction response and the responses stored in the response array should be used instead.

See also

set_response_array_complete(), get_response_array_ptr()

7.28.3.36 reset() [1/3]

```
void amba_pv::amba_pv_extension::reset ( )  [inline]
```

Resets members of this AMBA-PV extension to their default value.

7.28.3.37 reset() [2/3]

```
void amba_pv::amba_pv_extension::reset (
            int size,
            const amba_pv_control * ctrl )  [inline]
```

Resets members of this AMBA-PV extension for read and write transactions.

Parameters

size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. An instance with size 0 is considered to be an invalid extension
ctrl	optional AMBA 4 control information (set to `NULL` if unused).

7.28.3.38 reset() [3/3]

```
void amba_pv::amba_pv_extension::reset (
```

```
unsigned int length,
int size,
const amba_pv_control * ctrl,
amba_pv_burst_t burst )  [inline]
```

Resets members of this AMBA-PV extension for burst read and write transactions.

Parameters

length	transaction burst length as in [1-256].
size	transaction burst size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128].
ctrl	optional AMBA 4 control information (set to `NULL` if unused).
burst	transaction burst type as one of `AMBA_PV_INCR`, `AMBA_PV_FIXED`, `AMBA_PV_WRAP`.

7.29 amba_pv::amba_pv_from_tlm_bridge< BUSWIDTH > Class Template Reference

TLM 2.0 BP to AMBA-PV bridge module.
`#include <models/amba_pv_bridges.h>`
Inherits amba_pv::amba_pv_bw_transport_if, and sc_core::sc_module.

Public Member Functions

- amba_pv_from_tlm_bridge (const sc_core::sc_module_name &)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this bridge.

Data Fields

- tlm_utils::simple_target_socket< amba_pv_from_tlm_bridge, BUSWIDTH, tlm::tlm_base_protocol_types > tlm_s

  Slave socket from TLM 2.0 BP.
- amba_pv_master_socket< BUSWIDTH > amba_pv_m

  Master socket to AMBA-PV.

Protected Member Functions

- virtual void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)

  Invalidates DMI pointers previously established for the specified DMI region.

7.29.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_from_tlm_bridge< BUSWIDTH >

TLM 2.0 BP to AMBA-PV bridge module.
The amba_pv_from_tlm_bridge class translates TLM 2.0 BP transactions into AMBA-PV specific transactions. This consists mainly in adding the AMBA-PV extension (class amba_pv_extension) to the TLM 2.0 GP. In addition, the bridge checks the following:

- The address attribute must be aligned to the bus length for burst transactions and to the data length for single transactions. If not, an error response of `tlm::TLM_ADDRESS_ERROR_RESPONSE` is returned.
- The data length attribute must be a multiple of the bus length for burst transactions. If not, an error response of `tlm::TLM_BURST_ERROR_REPONSE` is returned.

- The streaming width attribute must be equal to the bus length for fixed burst transactions. If not, an error response of `tlm::TLM_BURST_ERROR_REPONSE` is returned.
- The byte enable pointer attribute must be `NULL` on read transactions. If not, an error response of `tlm::TLM_BYTE_ENABLE_ERROR_REPONSE` is returned.
- The byte enable length attribute must be equal to the data length for single write transactions and a multiple of the bus length for burst write transactions. If not, an error response of `tlm::TLM_BYTE_ENABLE_ERROR_REPONSE` is returned.

Note

The bus length is defined as (*BUSWIDTH* + 7) / 8.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

See also

amba_pv_extension

7.29.2 Constructor & Destructor Documentation

7.29.2.1 amba_pv_from_tlm_bridge()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_from_tlm_bridge< BUSWIDTH >::amba_pv_from_tlm_bridge (
            const sc_core::sc_module_name & name )  [inline], [explicit]
```

Constructor.

Parameters

name	bridge name.

7.29.3 Member Function Documentation

7.29.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_from_tlm_bridge< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string of this bridge.

7.29.3.2 invalidate_direct_mem_ptr()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_from_tlm_bridge< BUSWIDTH >::invalidate_direct_mem_ptr (
            int ,
            sc_dt::uint64 start_range,
            sc_dt::uint64 end_range )  [inline], [protected], [virtual]
```

Invalidates DMI pointers previously established for the specified DMI region.
This version of the method converts DMI invalidate from AMBA-PV backward to TLM 2.0 BP.
Implements amba_pv::amba_pv_bw_transport_if.

7.29.4 Field Documentation

7.29.4.1 tlm_s

```
template<unsigned int BUSWIDTH = 64>
tlm_utils::simple_target_socket<amba_pv_from_tlm_bridge, BUSWIDTH, tlm::tlm_base_protocol_↩
types> amba_pv::amba_pv_from_tlm_bridge< BUSWIDTH >::tlm_s
```

Slave socket from TLM 2.0 BP.

7.29.4.2 amba_pv_m

```
template<unsigned int BUSWIDTH = 64>
amba_pv_master_socket<BUSWIDTH> amba_pv::amba_pv_from_tlm_bridge< BUSWIDTH >::amba_pv_m
```

Master socket to AMBA-PV.

7.30 amba_pv::amba_pv_fw_transport_if Class Reference

AMBA-PV core transaction interface.

`#include <core/amba_pv_core_ifs.h>`

Inherits sc_core::sc_interface.

Inherited by amba_pv::amba_pv_slave_base< BUSWIDTH >[virtual], amba_pv::amba_pv_slave_base< 64 >[virtual], amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >[virtual], amba_pv::amba_pv_ace_simple_probe< BUSWIDTH > amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >[virtual], amba_pv::amba_pv_exclusive_monitor amba_pv::amba_pv_protocol_checker< BUSWIDTH >[virtual], amba_pv::amba_pv_simple_probe< BUSWIDTH >[virtua amba_pv::amba_pv_slave_base< BUSWIDTH >[virtual], and amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >[virtual

Public Member Functions

- virtual void b_transport (int socket_id, amba_pv_transaction &trans, sc_core::sc_time &t)=0
  Blocking transport.
- virtual unsigned int transport_dbg (int socket_id, amba_pv_transaction &trans)=0
  Debug access to a target.
- virtual bool get_direct_mem_ptr (int socket_id, amba_pv_transaction &trans, tlm::tlm_dmi &dmi_data)=0
  Requests a DMI access based on the specified transaction.

7.30.1 Detailed Description

AMBA-PV core transaction interface.

This is a tagged variant of the `tlm::tlm_fw_transport_if` interface. This interface is used for the forward path.

Note

AMBA-PV slaves must implement the amba_pv_fw_transport_if interface.

7.30.2 Member Function Documentation

7.30.2.1 b_transport()

```
virtual void amba_pv::amba_pv_fw_transport_if::b_transport (
            int socket_id,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [pure virtual]
```

Blocking transport.

Parameters

socket↩ _id	socket identifier (ignored on the master side).
trans	transaction.
t	timing annotation.

Implemented in amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >, amba_pv::amba_pv_ace_simple_probe< BUSWIDTH amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >, amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >, amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >, amba_pv::amba_pv_memory_base< BUSWIDTH >, amba_pv::amba_pv_memory_base< 64 >, amba_pv::amba_pv_protocol_checker< BUSWIDTH >, amba_pv::amba_pv_simple_pro amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, and amba_pv::amba_pv_slave_base< 64 >.

7.30.2.2 transport_dbg()

```
virtual unsigned int amba_pv::amba_pv_fw_transport_if::transport_dbg (
            int socket_id,
            amba_pv_transaction & trans )  [pure virtual]
```

Debug access to a target.

This use the same path as the b_transport() interface. This debug access must be performed without any of the delays, waits, event notifications or side effects associated with a regular transaction. This debug access is, therefore, non-intrusive.

Parameters

socket↩ _id	socket identifier (ignored on the master side).
trans	transaction.

Returns

number of bytes read or written or, if error, 0.

Implemented in amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >, amba_pv::amba_pv_ace_simple_probe< BUSWIDTH amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >, amba_pv::amba_pv_exclusive_monitor< BUSWIDT amba_pv::amba_pv_protocol_checker< BUSWIDTH >, amba_pv::amba_pv_simple_probe< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_bas and amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >.

7.30.2.3 get_direct_mem_ptr()

```
virtual bool amba_pv::amba_pv_fw_transport_if::get_direct_mem_ptr (
            int socket_id,
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [pure virtual]
```

Requests a DMI access based on the specified transaction.

Returns a reference to a DMI descriptor of type `tlm_dmi` that contains the bounds of the DMI region.

Parameters

socket↩ _id	socket identifier (ignored on the master side).
trans	transaction.
dmi_data	DMI Descriptor.

Returns

`true` if DMI access is granted, `false` otherwise.

Implemented in amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >, amba_pv::amba_pv_ace_simple_probe< BUSWIDTH
amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >, amba_pv::amba_pv_exclusive_monitor< BUSWIDT
amba_pv::amba_pv_protocol_checker< BUSWIDTH >, amba_pv::amba_pv_simple_probe< BUSWIDTH >,
amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_bas
and amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >.

7.31 amba_pv::ext::amba_pv_fw_transport_if Class Reference

AMBA-PV core transaction interface.
`#include <core/amba_pv_ext_core_ifs.h>`
Inherits sc_core::sc_interface.
Inherited by amba_pv::ext::amba_pv_ace_slave_base[virtual], and amba_pv::ext::amba_pv_slave_base< BUSWIDTH >[vir

Public Member Functions

- virtual void b_transport (int socket_id, amba_pv_transaction &trans, sc_core::sc_time &t)=0

  Blocking transport.
- virtual unsigned int transport_dbg (int socket_id, amba_pv_transaction &trans)=0

  Debug access to a slave.
- virtual bool get_direct_mem_ptr (int socket_id, amba_pv_transaction &trans, tlm::tlm_dmi &dmi_data)=0

  Requests a DMI access based on the specified transaction.

7.31.1 Detailed Description

AMBA-PV core transaction interface.
This is a tagged variant of the `tlm::tlm_fw_transport_if` interface. This interface is used for the forward path.

Note

AMBA-PV slaves and AMBA-PV ACE slaves must implement the amba_pv_fw_transport_if interface.

7.31.2 Member Function Documentation

7.31.2.1 b_transport()

```
virtual void amba_pv::ext::amba_pv_fw_transport_if::b_transport (
            int socket_id,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [pure virtual]
```

Blocking transport.

Parameters

socket_id	socket identifier (index into bound interfaces on the master side).
trans	transaction.
t	timing annotation.

Implemented in amba_pv::ext::amba_pv_ace_slave_base, and amba_pv::ext::amba_pv_slave_base< BUSWIDTH >.

7.31.2.2 transport_dbg()

```
virtual unsigned int amba_pv::ext::amba_pv_fw_transport_if::transport_dbg (
            int socket_id,
            amba_pv_transaction & trans )  [pure virtual]
```

Debug access to a slave.
This use the same path as the b_transport() interface. This debug access must be performed without any of the delays, waits, event notifications or side effects associated with a regular transaction. This debug access is, therefore, non-intrusive.

Parameters

socket_id	socket identifier (index into bound interfaces on the master side).
trans	transaction.

Returns

number of bytes read or written or, if error, 0.

Implemented in amba_pv::ext::amba_pv_ace_slave_base, and amba_pv::ext::amba_pv_slave_base< BUSWIDTH >.

7.31.2.3 get_direct_mem_ptr()

```
virtual bool amba_pv::ext::amba_pv_fw_transport_if::get_direct_mem_ptr (
            int socket_id,
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [pure virtual]
```

Requests a DMI access based on the specified transaction.
Returns a reference to a DMI descriptor of type `tlm_dmi` that contains the bounds of the DMI region.

Parameters

socket_id	socket identifier (index into bound interfaces on the master side).
trans	transaction.
dmi_data	DMI Descriptor.

Returns

`true` if DMI access is granted, `false` otherwise.

Implemented in amba_pv::ext::amba_pv_ace_slave_base, and amba_pv::ext::amba_pv_slave_base< BUSWIDTH >.

7.32 amba_pv::amba_pv_heap_allocator Class Reference

AMBA-PV heap memory allocator.
`#include <models/amba_pv_heap_allocator.h>`

Static Public Member Functions

- static uint8_t * allocate (const std::string &name, const sc_dt::uint64 &addr, const std::size_t &size)

  Internal - allocate memory aligned to host memory page size.
- static void deallocate (const std::string &name, const sc_dt::uint64 &addr, unsigned char *data)

  Internal - free host page aligned memory allocated by allocate()

7.32.1 Detailed Description

AMBA-PV heap memory allocator.
The amba_pv_heap_allocator class models a simple allocator and features:

- allocate function which allocates the memory needed. Allocated memory will be host page size aligned
- deallocate function which frees the memory when no longer required

7.32.2 Member Function Documentation

7.32.2.1 allocate()

```
uint8_t * amba_pv::amba_pv_heap_allocator::allocate (
            const std::string & name,
            const sc_dt::uint64 & addr,
            const std::size_t & size )  [inline], [static]
```

Internal - allocate memory aligned to host memory page size.

Parameters

size	in bytes of allocation required

7.32.2.2 deallocate()

```
void amba_pv::amba_pv_heap_allocator::deallocate (
            const std::string & name,
            const sc_dt::uint64 & addr,
            unsigned char * data )  [inline], [static]
```

Internal - free host page aligned memory allocated by allocate()

Parameters

pointer	returned by previous call to allocate()

7.33 amba_pv::amba_pv_if< BUSWIDTH > Class Template Reference

AMBA-PV user-layer transaction interface.
`#include <user/amba_pv_if.h>`

Public Member Functions

- virtual ~amba_pv_if ()

  Destructor.
- unsigned int get_bus_width_bytes () const

  Returns the bus width, in bytes, based on BUSWIDTH.
- virtual amba_pv_resp_t read (int socket_id, const sc_dt::uint64 &addr, unsigned char *data, unsigned int size, const amba_pv_control *ctrl, sc_core::sc_time &t)=0

  Completes a read transaction.
- virtual amba_pv_resp_t write (int socket_id, const sc_dt::uint64 &addr, unsigned char *data, unsigned int size, const amba_pv_control *ctrl, unsigned char *strb, sc_core::sc_time &t)=0

  Completes a write transaction.

- virtual amba_pv_resp_t burst_read (int socket_id, const sc_dt::uint64 &addr, unsigned char *data, unsigned int length, unsigned int size, const amba_pv_control *ctrl, amba_pv_burst_t burst, sc_core::sc_time &t)=0

  Completes a burst read transaction.
- virtual amba_pv_resp_t burst_write (int socket_id, const sc_dt::uint64 &addr, unsigned char *data, unsigned int length, unsigned int size, const amba_pv_control *ctrl, amba_pv_burst_t burst, unsigned char *strb, unsigned int strb_length, sc_core::sc_time &t)=0

  Completes a burst write transaction.
- virtual bool get_direct_mem_ptr (int socket_id, tlm::tlm_command command, const sc_dt::uint64 &addr, const amba_pv_control *ctrl, tlm::tlm_dmi &dmi_data)=0

  Requests DMI access to the specified address and returns a reference to a DMI descriptor.
- virtual unsigned int debug_read (int socket_id, const sc_dt::uint64 &addr, unsigned char *data, unsigned int length, const amba_pv_control *ctrl)=0

  Non-intrusive debug read transaction.
- virtual unsigned int debug_write (int socket_id, const sc_dt::uint64 &addr, unsigned char *data, unsigned int length, const amba_pv_control *ctrl)=0

  Non-intrusive debug write transaction.
- virtual amba_pv_resp_t atomic_store (int socket_id, const sc_dt::uint64 &addr, unsigned char *data, unsigned int length, unsigned int size, const amba_pv_control *ctrl, amba_pv_atomic_subop_t subop, amba_pv_atomic_endianness_t endianness, sc_core::sc_time &t)=0

  Completes an atomic store transaction.
- virtual amba_pv_resp_t atomic_load (int socket_id, const sc_dt::uint64 &addr, unsigned char *data, unsigned int length, unsigned int size, const amba_pv_control *ctrl, amba_pv_atomic_subop_t subop, amba_pv_atomic_endianness_t endianness, sc_core::sc_time &t)=0

  Completes an atomic load transaction.
- virtual amba_pv_resp_t atomic_swap (int socket_id, const sc_dt::uint64 &addr, unsigned char *data, unsigned int length, unsigned int size, const amba_pv_control *ctrl, sc_core::sc_time &t)=0

  Completes an atomic swap transaction.
- virtual amba_pv_resp_t atomic_compare (int socket_id, const sc_dt::uint64 &addr, unsigned char *data, unsigned int length, unsigned int size, const amba_pv_control *ctrl, sc_core::sc_time &t)=0

  Completes an atomic compare transaction.

7.33.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_if< BUSWIDTH >

AMBA-PV user-layer transaction interface.
This interface is implemented by amba_pv_master_socket and amba_pv_slave_base.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024; defaults to 64.

See also

amba_pv_master_socket, amba_pv_slave_base, amba_pv_slave_socket

7.33.2 Constructor & Destructor Documentation

7.33.2.1 ~amba_pv_if()

```
template<unsigned int BUSWIDTH = 64>
virtual amba_pv::amba_pv_if< BUSWIDTH >::~amba_pv_if ( )  [inline], [virtual]
```

Destructor.

7.33.3 Member Function Documentation

7.33.3.1 get_bus_width_bytes()

```
template<unsigned int BUSWIDTH = 64>
unsigned int amba_pv::amba_pv_if< BUSWIDTH >::get_bus_width_bytes ( ) const  [inline]
```

Returns the bus width, in bytes, based on *BUSWIDTH*.

7.33.3.2 read()

```
template<unsigned int BUSWIDTH = 64>
virtual amba_pv_resp_t amba_pv::amba_pv_if< BUSWIDTH >::read (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [pure virtual]
```

Completes a read transaction.

Parameters

socket_id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes.
size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to NULL if unused on the master side).
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

Implemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, amba_pv::amba_pv_simple_memory< BUSWIDTH >, amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::amba_pv_master_socket< 64 >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::ext::amba_pv_slave_ba amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, and amba_pv::amba_pv_slave_base< 64 >.

7.33.3.3 write()

```
template<unsigned int BUSWIDTH = 64>
virtual amba_pv_resp_t amba_pv::amba_pv_if< BUSWIDTH >::write (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int size,
            const amba_pv_control * ctrl,
            unsigned char * strb,
            sc_core::sc_time & t )  [pure virtual]
```

Completes a write transaction.

Parameters

socket↩ _id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes.
size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to NULL if unused on the master side).
strb	write strobes pointer (set to NULL if none). It must point to an array of *size* elements.
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

Implemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, amba_pv::amba_pv_simple_memory< BUSWIDTH >, amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::amba_pv_master_socket< 64 >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::ext::amba_pv_slave_ba amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, and amba_pv::amba_pv_slave_base< 64 >.

7.33.3.4 burst_read()

```
template<unsigned int BUSWIDTH = 64>
virtual amba_pv_resp_t amba_pv::amba_pv_if< BUSWIDTH >::burst_read (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst,
            sc_core::sc_time & t )  [pure virtual]
```

Completes a burst read transaction.

Parameters

socket↩ _id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of (*size* ∗ *length*) bytes.
length	transaction burst length as in [1-16].
size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to NULL if unused on the master side).
burst	transaction burst type, one of AMBA_PV_INCR, AMBA_PV_FIXED, or AMBA_PV_WRAP.
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

Implemented in amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >, amba_pv::amba_pv_master_socket< BUSWIDTH amba_pv::amba_pv_master_socket< 64 >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::ext::amba_pv_slave_ba

amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, and amba_pv::amba_pv_slave_base< 64 >.

7.33.3.5 burst_write()

```
template<unsigned int BUSWIDTH = 64>
virtual amba_pv_resp_t amba_pv::amba_pv_if< BUSWIDTH >::burst_write (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst,
            unsigned char * strb,
            unsigned int strb_length,
            sc_core::sc_time & t )  [pure virtual]
```

Completes a burst write transaction.

Parameters

socket_id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of (*size* ∗ *length*) bytes.
length	transaction burst length as in [1-16].
size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to NULL if unused on the master side).
burst	transaction burst type, one of AMBA_PV_INCR, AMBA_PV_FIXED, or AMBA_PV_WRAP.
strb	write strobes pointer (set to NULL if none).
strb_length	Write strobes length. It must be a multiple of *size*.
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

Implemented in amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >, amba_pv::amba_pv_master_socket< BUSWIDTH amba_pv::amba_pv_master_socket< 64 >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::ext::amba_pv_slave_ba amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, and amba_pv::amba_pv_slave_base< 64 >.

7.33.3.6 get_direct_mem_ptr()

```
template<unsigned int BUSWIDTH = 64>
virtual bool amba_pv::amba_pv_if< BUSWIDTH >::get_direct_mem_ptr (
            int socket_id,
            tlm::tlm_command command,
            const sc_dt::uint64 & addr,
            const amba_pv_control * ctrl,
            tlm::tlm_dmi & dmi_data )  [pure virtual]
```

Requests DMI access to the specified address and returns a reference to a DMI descriptor.
The DMI descriptor contains the bounds of the DMI region.

Parameters

socket↩ _id	socket identifier (index into bound interfaces on the master side).
command	`tlm::TLM_READ_COMMAND` for a DMI read access request. `tlm::TLM_WRITE_COMMAND` for a DMI write access request.
addr	address to which the DMI access is requested.
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).
dmi_data	returned DMI descriptor.

Returns

`true` if a DMI region is granted, `false` otherwise.

Implemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, amba_pv::amba_pv_simple_memory< BUSWIDTH >, amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::amba_pv_master_socket< 64 >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::ext::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, and amba_pv::amba_pv_slave_base< 64 >.

7.33.3.7 debug_read()

```
template<unsigned int BUSWIDTH = 64>
virtual unsigned int amba_pv::amba_pv_if< BUSWIDTH >::debug_read (
             int socket_id,
             const sc_dt::uint64 & addr,
             unsigned char * data,
             unsigned int length,
             const amba_pv_control * ctrl )  [pure virtual]
```

Non-intrusive debug read transaction.

Parameters

socket↩ _id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of *length* bytes.
length	transaction length.
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).

Returns

number of bytes read or, if error, 0.

Implemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, amba_pv::amba_pv_simple_memory< BUSWIDTH >, amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::amba_pv_master_socket< 64 >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::ext::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, and amba_pv::amba_pv_slave_base< 64 >.

7.33.3.8 debug_write()

```
template<unsigned int BUSWIDTH = 64>
virtual unsigned int amba_pv::amba_pv_if< BUSWIDTH >::debug_write (
             int socket_id,
```

```
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            const amba_pv_control * ctrl )  [pure virtual]
```

Non-intrusive debug write transaction.

Parameters

socket_id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of *length* bytes.
length	transaction length.
ctrl	AMBA 3 control information (set to NULL if unused on the master side).

Returns

number of bytes written or, if error, 0.

Implemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, amba_pv::amba_pv_simple_memory< BUSWIDTH >, amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::amba_pv_master_socket< 64 >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::ext::amba_pv_slave_ba amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, and amba_pv::amba_pv_slave_base< 64 >.

7.33.3.9 atomic_store()

```
template<unsigned int BUSWIDTH = 64>
virtual amba_pv_resp_t amba_pv::amba_pv_if< BUSWIDTH >::atomic_store (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time & t )  [pure virtual]
```

Completes an atomic store transaction.

Parameters

socket_id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes.
length	sets the data transfers in a transaction.
size	sets the transaction size in bytes. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to NULL if unused on the master side).
subop	operation type of the atomic transaction.
endianness	endianness of the atomic operation. Data is interpreted in big endian order if enabled.
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

Note

Byte enable is unsupported for atomic transactions.

The product of size and length must be one of [1, 2, 4, 8].

Implemented in amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >, amba_pv::amba_pv_master_socket< BUSWIDTH amba_pv::amba_pv_master_socket< 64 >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::ext::amba_pv_slave_ba amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_bas amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, and amba_pv::amba_pv_simple_memory< BUSWIDTH >.

7.33.3.10 atomic_load()

```
template<unsigned int BUSWIDTH = 64>
virtual amba_pv_resp_t amba_pv::amba_pv_if< BUSWIDTH >::atomic_load (
             int socket_id,
             const sc_dt::uint64 & addr,
             unsigned char * data,
             unsigned int length,
             unsigned int size,
             const amba_pv_control * ctrl,
             amba_pv_atomic_subop_t subop,
             amba_pv_atomic_endianness_t endianness,
             sc_core::sc_time & t )  [pure virtual]
```

Completes an atomic load transaction.

Parameters

socket_id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes. The array initially contains the sending data, then the original data at the address before the atomic operation is returned to the array.
length	sets the data transfers in a transaction.
size	sets the transaction size in bytes. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to NULL if unused on the master side).
subop	operation type of the atomic transaction.
endianness	endianness of the atomic operation. Data is interpreted in big endian order if enabled.
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

Note

Byte enable is unsupported for atomic transactions.

The product of size and length must be one of [1, 2, 4, 8].

Implemented in amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >, amba_pv::amba_pv_master_socket< BUSWIDTH amba_pv::amba_pv_master_socket< 64 >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::ext::amba_pv_slave_ba amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_bas amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, and amba_pv::amba_pv_simple_memory< BUSWIDTH >.

7.33.3.11 atomic_swap()

```
template<unsigned int BUSWIDTH = 64>
virtual amba_pv_resp_t amba_pv::amba_pv_if< BUSWIDTH >::atomic_swap (
             int socket_id,
             const sc_dt::uint64 & addr,
             unsigned char * data,
             unsigned int length,
             unsigned int size,
             const amba_pv_control * ctrl,
             sc_core::sc_time & t )  [pure virtual]
```

Completes an atomic swap transaction.

Parameters

socket_id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes. The array initially contains the sending data, then the original data at the address before the atomic operation is returned to the array.
length	sets the data transfers in a transaction.
size	sets the transaction size in bytes. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).
t	timing annotation.

Returns

`AMBA_PV_OKAY` if the transaction is successful.

Note

Byte enable is unsupported for atomic transactions.

The product of `size` and `length` must be one of [1, 2, 4, 8].

Implemented in amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >, amba_pv::amba_pv_master_socket< BUSWIDTH amba_pv::amba_pv_master_socket< 64 >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::ext::amba_pv_slave_ba amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_bas amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, and amba_pv::amba_pv_simple_memory< BUSWIDTH >.

7.33.3.12 atomic_compare()

```
template<unsigned int BUSWIDTH = 64>
virtual amba_pv_resp_t amba_pv::amba_pv_if< BUSWIDTH >::atomic_compare (
             int socket_id,
             const sc_dt::uint64 & addr,
             unsigned char * data,
             unsigned int length,
             unsigned int size,
             const amba_pv_control * ctrl,
             sc_core::sc_time & t )  [pure virtual]
```

Completes an atomic compare transaction.

Parameters

socket↩ _id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes. The array initially comprises of comarping and swapping data, after the transaction, the original data at the address before the atomic operation is returned to the array.
length	sets the data transfers in a transaction.
size	sets the transaction size in bytes covering both comparing and swapping data. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).
t	timing annotation.

Returns

`AMBA_PV_OKAY` if the transaction is successful.

Note

Transaction data pointer points to an array comprising of comparing and swapping data, with their order determined by the transaction address. Comparing data is sent first if the address points to the lower half of the transaction (i.e. lowest address byte); swapping data is sent first if the address points to the upper half of the transaction (i.e. lowest address plus half the ∗*size*∗).

The original data at the address is returned to *data* pointer starting from the lowest byte. The returning size is half the sending *size*.

Byte enable is unsupported for atomic transactions.

The product of `size` and `length` must be one of [2, 4, 8, 16, 32] while `size` must be larger than 1.

Implemented in amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >, amba_pv::amba_pv_master_socket< BUSWIDTH amba_pv::amba_pv_master_socket< 64 >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::ext::amba_pv_slave_ba amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_base< BUSWIDTH >, amba_pv::amba_pv_slave_bas amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, and amba_pv::amba_pv_simple_memory< BUSWIDTH >.

7.34 amba_pv::amba_pv_master_base Class Reference

Base class for all AMBA-PV master modules.
`#include <user/amba_pv_master_base.h>`
Inherits amba_pv::amba_pv_bw_transport_if.

Public Member Functions

- amba_pv_master_base (const std::string &)

  Constructor.
- std::string get_name () const

  Returns the name of this master.

Protected Member Functions

- virtual void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)

  Invalidates DMI pointers previously established for the specified DMI region.

7.34.1 Detailed Description

Base class for all AMBA-PV master modules.
amba_pv_master_base is intended to be bound to one or more amba_pv_master_socket.

Note

amba_pv_master_base is not an `sc_module`.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 amba_pv_master_base()

```
amba_pv::amba_pv_master_base::amba_pv_master_base (
            const std::string & name )  [inline], [explicit]
```

Constructor.

Parameters

name	master name.

7.34.3 Member Function Documentation

7.34.3.1 get_name()

```
std::string amba_pv::amba_pv_master_base::get_name ( ) const  [inline]
```

Returns the name of this master.

7.34.3.2 invalidate_direct_mem_ptr()

```
void amba_pv::amba_pv_master_base::invalidate_direct_mem_ptr (
            int ,
            sc_dt::uint64 ,
            sc_dt::uint64  )  [inline], [protected], [virtual]
```

Invalidates DMI pointers previously established for the specified DMI region.
This default implementation does nothing.
Implements amba_pv::amba_pv_bw_transport_if.

7.35 amba_pv::ext::amba_pv_master_base Class Reference

Base class for all AMBA-PV master modules.
`#include <user/amba_pv_ext_master_base.h>`
Inherits amba_pv::ext::amba_pv_bw_transport_if.

Public Member Functions

- amba_pv_master_base (const std::string &)
  Constructor.
- std::string get_name () const
  Returns the name of this master.

Protected Member Functions

- virtual void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)

  Invalidates DMI pointers previously established for the specified DMI region.

7.35.1 Detailed Description

Base class for all AMBA-PV master modules.
amba_pv_master_base is intended to be bound to one or more amba_pv_master_socket.

Note

amba_pv_master_base is not an `sc_module`.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 amba_pv_master_base()

```
amba_pv::ext::amba_pv_master_base::amba_pv_master_base (
            const std::string & name )  [inline], [explicit]
```

Constructor.

Parameters

name	master name.

7.35.3 Member Function Documentation

7.35.3.1 get_name()

```
std::string amba_pv::ext::amba_pv_master_base::get_name ( ) const  [inline]
```

Returns the name of this master.

7.35.3.2 invalidate_direct_mem_ptr()

```
void amba_pv::ext::amba_pv_master_base::invalidate_direct_mem_ptr (
            int ,
            sc_dt::uint64 ,
            sc_dt::uint64  )  [inline], [protected], [virtual]
```

Invalidates DMI pointers previously established for the specified DMI region.
This default implementation does nothing.
Implements amba_pv::ext::amba_pv_bw_transport_if.

7.36 amba_pv::amba_pv_master_socket< BUSWIDTH > Class Template Reference

AMBA-PV socket to be instantiated on the master side.
`#include <sockets/amba_pv_master_socket.h>`
Inherits amba_pv::amba_pv_if< 64 >, amba_pv::amba_pv_socket_base, and tlm_utils::simple_initiator_socket_tagged< amba_pv_bw_transport_if, 64, amba_pv_protocol_types >.

Public Member Functions

- amba_pv_master_socket ()

  Default constructor.
- amba_pv_master_socket (const char *, int=0)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this socket.
- virtual amba_pv_resp_t read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes a read transaction.
- amba_pv_resp_t read (const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes a read transaction.
- virtual amba_pv_resp_t write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, unsigned char *, sc_core::sc_time &)

  Completes a write transaction.
- amba_pv_resp_t write (const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, unsigned char *, sc_core::sc_time &)

  Completes a write transaction.
- virtual amba_pv_resp_t burst_read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_burst_t, sc_core::sc_time &)

  Completes a burst read transaction.
- amba_pv_resp_t burst_read (const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_burst_t, sc_core::sc_time &)

  Completes a burst read transaction.
- virtual amba_pv_resp_t burst_write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_burst_t, unsigned char *, unsigned int, sc_core::sc_time &)

  Completes a burst write transaction.
- amba_pv_resp_t burst_write (const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_burst_t, unsigned char *, unsigned int, sc_core::sc_time &)

  Completes a burst write transaction.
- virtual bool get_direct_mem_ptr (int, tlm::tlm_command, const sc_dt::uint64 &, const amba_pv_control *, tlm::tlm_dmi &)

  Requests DMI access to the specified address and returns a reference to a DMI descriptor.
- bool get_direct_mem_ptr (tlm::tlm_command, const sc_dt::uint64 &, const amba_pv_control *, tlm::tlm_dmi &)

  Requests DMI access to the specified address and returns a reference to a DMI descriptor.
- virtual unsigned int debug_read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug read transaction.
- virtual unsigned int debug_write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug write transaction.
- unsigned int debug_read (const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug read transaction.
- unsigned int debug_write (const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug write transaction.
- virtual amba_pv_resp_t atomic_store (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic store transaction.
- amba_pv_resp_t atomic_store (const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

Completes an atomic store transaction.

- virtual amba_pv_resp_t atomic_load (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic load transaction.
- amba_pv_resp_t atomic_load (const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic load transaction.
- virtual amba_pv_resp_t atomic_swap (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic swap transaction.
- amba_pv_resp_t atomic_swap (const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic swap transaction.
- virtual amba_pv_resp_t atomic_compare (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic compare transaction.
- amba_pv_resp_t atomic_compare (const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic compare transaction.
- void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking transport.
- void b_transport (amba_pv_transaction &, sc_core::sc_time &)

  Blocking transport.
- unsigned int transport_dbg (int, amba_pv_transaction &)

  Debug access to a target.
- unsigned int transport_dbg (amba_pv_transaction &)

  Debug access to a target.
- bool get_direct_mem_ptr (int, amba_pv_transaction &, tlm::tlm_dmi &)

  Requests a DMI access based on the specified transaction.
- bool get_direct_mem_ptr (amba_pv_transaction &, tlm::tlm_dmi &)

  Requests a DMI access based on the specified transaction.
- void bind (amba_pv_bw_transport_if &)

  Binds the specified interface to this socket.
- void operator() (amba_pv_bw_transport_if &)

  Binds the specified interface to this socket.

7.36.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_master_socket< BUSWIDTH >

AMBA-PV socket to be instantiated on the master side.
This socket is for use as a master socket bound to one or more slave sockets.
amba_pv_master_socket provides implementations for the amba_pv_if user-layer interface.
This socket inherits from the OSCI TLM 2.0 `tlm_utils::simple_initiator_socket_tagged` class. A tagged socket enables a component to determine through which socket an incoming method call arrived. This is required if there are multiple master sockets such as in, for example, a bus decoder.

Note

The current implementation of amba_pv_master_socket inherits from OSCI TLM 2.0 `tlm_utils::simple_initiator_socket_tagged`. Hence, if compiling applications that use amba_pv_master_socket with OSCI SystemC, it is required to define the macro `SC_INCLUDE_DYNAMIC_PROCESSES` before including the OSCI SystemC header file.

This socket, as its base class `tlm_utils::simple_initiator_socket_tagged`, does not support hierarchical binding, master-socket-to-master-socket or slave-socket-to-slave-socket

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 amba_pv_master_socket() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_master_socket< BUSWIDTH >::amba_pv_master_socket  [inline]
```

Default constructor.

7.36.2.2 amba_pv_master_socket() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_master_socket< BUSWIDTH >::amba_pv_master_socket (
            const char * name,
            int socket_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.
socket↩ _id	socket identifier (defaults to 0).

7.36.3 Member Function Documentation

7.36.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_master_socket< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string of this socket.

Reimplemented in amba_pv::amba_pv_ace_master_socket< BUSWIDTH >, and amba_pv::amba_pv_ace_master_socket< 64 >.

7.36.3.2 read() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::read (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline], [virtual]
```

Completes a read transaction.

Implements amba_pv::amba_pv_if< 64 >.

7.36.3.3 read() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::read (
             const sc_dt::uint64 & addr,
             unsigned char * data,
             unsigned int size,
             const amba_pv_control * ctrl,
             sc_core::sc_time & t )  [inline]
```

Completes a read transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes.
size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to NULL if unused on the master side).
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

7.36.3.4 write() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::write (
             int socket_id,
             const sc_dt::uint64 & addr,
             unsigned char * data,
             unsigned int size,
             const amba_pv_control * ctrl,
             unsigned char * strb,
             sc_core::sc_time & t )  [inline], [virtual]
```

Completes a write transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.36.3.5 write() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::write (
             const sc_dt::uint64 & addr,
             unsigned char * data,
             unsigned int size,
             const amba_pv_control * ctrl,
             unsigned char * strb,
             sc_core::sc_time & t )  [inline]
```

Completes a write transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes.
size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().

Parameters

ctrl	AMBA 3 control information (set to NULL if unused on the master side).
strb	write strobes pointer (set to NULL if none). It must point to an array of *size* elements.
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

7.36.3.6 burst_read() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::burst_read (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst,
            sc_core::sc_time & t )  [inline], [virtual]
```

Completes a burst read transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.36.3.7 burst_read() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::burst_read (
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst,
            sc_core::sc_time & t )  [inline]
```

Completes a burst read transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of (*size* ∗ *length*) bytes.
length	transaction burst length as in [1-16].
size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to NULL if unused on the master side).
burst	transaction burst type, one of AMBA_PV_INCR, AMBA_PV_FIXED, or AMBA_PV_WRAP.
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

7.36.3.8 burst_write() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::burst_write (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst,
            unsigned char * strb,
            unsigned int strb_length,
            sc_core::sc_time & t )  [inline], [virtual]
```

Completes a burst write transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.36.3.9 burst_write() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::burst_write (
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst,
            unsigned char * strb,
            unsigned int strb_length,
            sc_core::sc_time & t )  [inline]
```

Completes a burst write transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of (*size* ∗ *length*) bytes.
length	transaction burst length as in [1-16].
size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to NULL if unused on the master side).
burst	transaction burst type, one of AMBA_PV_INCR, AMBA_PV_FIXED, or AMBA_PV_WRAP.
strb	write strobes pointer (set to NULL if none).
strb_length	Write strobes length. It must be a multiple of *size*.
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

7.36.3.10 get_direct_mem_ptr() [1/4]

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_master_socket< BUSWIDTH >::get_direct_mem_ptr (
            int socket_id,
            tlm::tlm_command command,
```

```
            const sc_dt::uint64 & addr,
            const amba_pv_control * ctrl,
            tlm::tlm_dmi & dmi_data )  [inline], [virtual]
```

Requests DMI access to the specified address and returns a reference to a DMI descriptor.
Implements amba_pv::amba_pv_if< 64 >.

7.36.3.11 get_direct_mem_ptr() [2/4]

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_master_socket< BUSWIDTH >::get_direct_mem_ptr (
            tlm::tlm_command command,
            const sc_dt::uint64 & addr,
            const amba_pv_control * ctrl,
            tlm::tlm_dmi & dmi_data )  [inline]
```

Requests DMI access to the specified address and returns a reference to a DMI descriptor.

Parameters

command	`tlm::TLM_READ_COMMAND` for a DMI read access request. `tlm::TLM_WRITE_COMMAND` for a DMI write access request.
addr	address to which the DMI access is requested.
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).
dmi_data	returned DMI descriptor.

Returns

`true` if a DMI region is granted, `false` otherwise.

7.36.3.12 debug_read() [1/2]

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_master_socket< BUSWIDTH >::debug_read (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            const amba_pv_control * ctrl )  [inline], [virtual]
```

Non-intrusive debug read transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.36.3.13 debug_write() [1/2]

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_master_socket< BUSWIDTH >::debug_write (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            const amba_pv_control * ctrl )  [inline], [virtual]
```

Non-intrusive debug write transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.36.3.14 debug_read() [2/2]

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_master_socket< BUSWIDTH >::debug_read (
             const sc_dt::uint64 & addr,
             unsigned char * data,
             unsigned int length,
             const amba_pv_control * ctrl )  [inline]
```

Non-intrusive debug read transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of *length* bytes.
length	transaction length.
ctrl	AMBA 3 control information (set to NULL if unused on the master side).

Returns

number of bytes read or, if error, 0.

7.36.3.15 debug_write() [2/2]

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_master_socket< BUSWIDTH >::debug_write (
             const sc_dt::uint64 & addr,
             unsigned char * data,
             unsigned int length,
             const amba_pv_control * ctrl )  [inline]
```

Non-intrusive debug write transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of *length* bytes.
length	transaction length.
ctrl	AMBA 3 control information (set to NULL if unused on the master side).

Returns

number of bytes written or, if error, 0.

7.36.3.16 atomic_store() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::atomic_store (
             int socket_id,
             const sc_dt::uint64 & addr,
             unsigned char * data,
             unsigned int length,
             unsigned int size,
             const amba_pv_control * ctrl,
             amba_pv_atomic_subop_t subop,
```

```
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time & t )  [inline], [virtual]
```

Completes an atomic store transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.36.3.17 atomic_store() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::atomic_store (
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time & t )  [inline]
```

Completes an atomic store transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes.
length	sets the data transfers in a transaction.
size	sets the transaction size in bytes. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).
subop	operation type of the atomic transaction.
endianness	endianness of the atomic operation. Data is interpreted in big endian order if enabled.
t	timing annotation.

Returns

`AMBA_PV_OKAY` if the transaction is successful.

Note

Byte enable is unsupported for atomic transactions.

The product of `size` and `length` must be one of [1, 2, 4, 8].

7.36.3.18 atomic_load() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::atomic_load (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time & t )  [inline], [virtual]
```

Completes an atomic load transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.36.3.19 atomic_load() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::atomic_load (
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time & t )  [inline]
```

Completes an atomic load transaction.

Parameters

socket_id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes. The array initially contains the sending data, then the original data at the address before the atomic operation is returned to the array.
length	sets the data transfers in a transaction.
size	sets the transaction size in bytes. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).
subop	operation type of the atomic transaction.
endianness	endianness of the atomic operation. Data is interpreted in big endian order if enabled.
t	timing annotation.

Returns

`AMBA_PV_OKAY` if the transaction is successful.

Note

Byte enable is unsupported for atomic transactions.

The product of `size` and `length` must be one of [1, 2, 4, 8].

7.36.3.20 atomic_swap() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::atomic_swap (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline], [virtual]
```

Completes an atomic swap transaction.

Implements amba_pv::amba_pv_if< 64 >.

7.36.3.21 atomic_swap() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::atomic_swap (
```

```
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline]
```

Completes an atomic swap transaction.

Parameters

socket_id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes. The array initially contains the sending data, then the original data at the address before the atomic operation is returned to the array.
length	sets the data transfers in a transaction.
size	sets the transaction size in bytes. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).
t	timing annotation.

Returns

`AMBA_PV_OKAY` if the transaction is successful.

Note

Byte enable is unsupported for atomic transactions.

The product of `size` and `length` must be one of [1, 2, 4, 8].

7.36.3.22 atomic_compare() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::atomic_compare (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline], [virtual]
```

Completes an atomic compare transaction.

Implements amba_pv::amba_pv_if< 64 >.

7.36.3.23 atomic_compare() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_master_socket< BUSWIDTH >::atomic_compare (
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline]
```

Completes an atomic compare transaction.

Parameters

socket_id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes. The array initially comprises of comarping and swapping data, after the transaction, the original data at the address before the atomic operation is returned to the array.
length	sets the data transfers in a transaction.
size	sets the transaction size in bytes covering both comparing and swapping data. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).
t	timing annotation.

Returns

`AMBA_PV_OKAY` if the transaction is successful.

Note

Transaction data pointer points to an array comprising of comparing and swapping data, with their order determined by the transaction address. Comparing data is sent first if the address points to the lower half of the transaction (i.e. lowest address byte); swapping data is sent first if the address points to the upper half of the transaction (i.e. lowest address plus half the ∗*size*∗).

The original data at the address is returned to *data* pointer starting from the lowest byte. The returning size is half the sending *size*.

Byte enable is unsupported for atomic transactions.

The product of `size` and `length` must be one of [2, 4, 8, 16, 32] while `size` must be larger than 1.

7.36.3.24 b_transport() [1/2]

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_master_socket< BUSWIDTH >::b_transport (
             int socket_id,
             amba_pv_transaction & trans,
             sc_core::sc_time & t )  [inline]
```

Blocking transport.
This version of the method forwards the b_transport() call to the slave socket bound to this master socket.

Parameters

socket_id	socket identifier (ignored on the master side).
trans	transaction.
t	timing annotation.

7.36.3.25 b_transport() [2/2]

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_master_socket< BUSWIDTH >::b_transport (
             amba_pv_transaction & trans,
             sc_core::sc_time & t )  [inline]
```

Blocking transport.

Parameters

trans	transaction.
t	timing annotation.

7.36.3.26 transport_dbg() [1/2]

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_master_socket< BUSWIDTH >::transport_dbg (
            int socket_id,
            amba_pv_transaction & trans )  [inline]
```

Debug access to a target.
This version of the method forwards the transport_dbg() call to the slave socket bound to this master socket.

Parameters

socket_id	socket identifier (ignored on the master side).
trans	transaction.

Returns

number of bytes read or written or, if error, 0.

7.36.3.27 transport_dbg() [2/2]

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_master_socket< BUSWIDTH >::transport_dbg (
            amba_pv_transaction & trans )  [inline]
```

Debug access to a target.

Parameters

trans	transaction.

Returns

number of bytes read or written or, if error, 0.

7.36.3.28 get_direct_mem_ptr() [3/4]

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_master_socket< BUSWIDTH >::get_direct_mem_ptr (
            int socket_id,
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [inline]
```

Requests a DMI access based on the specified transaction.
This version of the method forwards the get_direct_mem_ptr() call to the slave socket bound to this master socket.

Parameters

socket↩ _id	socket identifier (ignored on the master side).
trans	transaction.
dmi_data	DMI Descriptor.

Returns

`true` if DMI access is granted, `false` otherwise.

7.36.3.29 get_direct_mem_ptr() [4/4]

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_master_socket< BUSWIDTH >::get_direct_mem_ptr (
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [inline]
```

Requests a DMI access based on the specified transaction.

Parameters

trans	transaction.
dmi_data	DMI Descriptor.

Returns

`true` if DMI access is granted, `false` otherwise.

7.36.3.30 bind()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_master_socket< BUSWIDTH >::bind (
            amba_pv_bw_transport_if & iface )  [inline]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_bw_transport_if interface to bind to this socket.

7.36.3.31 operator()()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_master_socket< BUSWIDTH >::operator() (
            amba_pv_bw_transport_if & iface )  [inline]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_bw_transport_if interface to bind to this socket.

7.37 amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL > Class Template Reference

AMBA-PV socket to be instantiated on the master side.

`#include <sockets/amba_pv_ext_master_socket.h>`

Inherits amba_pv::amba_pv_if< 64 >, and amba_pv::ext::amba_pv_base_master_socket< 64, 1, sc_core::SC_ONE_OR_MORE_B

Public Member Functions

- amba_pv_master_socket ()

  Default constructor.
- amba_pv_master_socket (const char *, int=0)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this socket.
- virtual amba_pv_resp_t read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes a read transaction.
- amba_pv_resp_t read (const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes a read transaction.
- virtual amba_pv_resp_t write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, unsigned char *, sc_core::sc_time &)

  Completes a write transaction.
- amba_pv_resp_t write (const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, unsigned char *, sc_core::sc_time &)

  Completes a write transaction.
- virtual amba_pv_resp_t burst_read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_burst_t, sc_core::sc_time &)

  Completes a burst read transaction.
- amba_pv_resp_t burst_read (const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_burst_t, sc_core::sc_time &)

  Completes a burst read transaction.
- virtual amba_pv_resp_t burst_write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_burst_t, unsigned char *, unsigned int, sc_core::sc_time &)

  Completes a burst write transaction.
- amba_pv_resp_t burst_write (const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_burst_t, unsigned char *, unsigned int, sc_core::sc_time &)

  Completes a burst write transaction.
- virtual bool get_direct_mem_ptr (int, tlm::tlm_command, const sc_dt::uint64 &, const amba_pv_control *, tlm::tlm_dmi &)

  Requests DMI access to the specified address and returns a reference to a DMI descriptor.
- bool get_direct_mem_ptr (tlm::tlm_command, const sc_dt::uint64 &, const amba_pv_control *, tlm::tlm_dmi &)

  Requests DMI access to the specified address and returns a reference to a DMI descriptor.
- virtual unsigned int debug_read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug read transaction.
- virtual unsigned int debug_write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug write transaction.
- unsigned int debug_read (const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug read transaction.

- unsigned int debug_write (const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug write transaction.
- virtual amba_pv_resp_t atomic_store (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic store transaction.
- amba_pv_resp_t atomic_store (const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic store transaction.
- virtual amba_pv_resp_t atomic_load (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic load transaction.
- amba_pv_resp_t atomic_load (const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic load transaction.
- virtual amba_pv_resp_t atomic_swap (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic swap transaction.
- amba_pv_resp_t atomic_swap (const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic swap transaction.
- virtual amba_pv_resp_t atomic_compare (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic compare transaction.
- amba_pv_resp_t atomic_compare (const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic compare transaction.
- void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking transport.
- void b_transport (amba_pv_transaction &, sc_core::sc_time &)

  Blocking transport.
- unsigned int transport_dbg (int, amba_pv_transaction &)

  Debug access to a target.
- unsigned int transport_dbg (amba_pv_transaction &)

  Debug access to a target.
- bool get_direct_mem_ptr (int, amba_pv_transaction &, tlm::tlm_dmi &)

  Requests a DMI access based on the specified transaction.
- bool get_direct_mem_ptr (amba_pv_transaction &, tlm::tlm_dmi &)

  Requests a DMI access based on the specified transaction.

7.37.1 Detailed Description

template<unsigned int BUSWIDTH = 64, int N = 1, sc_core::sc_port_policy POL = sc_core::SC_ONE_OR_MORE_BOUND>
class amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >

AMBA-PV socket to be instantiated on the master side.
This socket is for use as a master socket bound to one or more slave sockets.
amba_pv_master_socket provides implementations for the amba_pv_if user-layer interface.
To use this class, you must define the `AMBA_PV_INCLUDE_HIERARCHICAL_BINDING` macro at compile time.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.
N	number of bindings. Defaults to 1.
POL	port binding policy. Defaults to `sc_core::SC_ONE_OR_MORE_BOUND`.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 amba_pv_master_socket() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::amba_pv_master_socket  [inline]
```

Default constructor.

7.37.2.2 amba_pv_master_socket() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::amba_pv_master_socket (
            const char * name,
            int socket_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.
socket↩ _id	socket identifier (defaults to 0).

7.37.3 Member Function Documentation

7.37.3.1 kind()

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
const char * amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::kind  [inline], [virtual]
```

Returns the kind string of this socket.

Reimplemented from amba_pv::ext::amba_pv_base_master_socket< 64, 1, sc_core::SC_ONE_OR_MORE_BOUND >.

7.37.3.2 read() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::read (
            int index,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline], [virtual]
```

Completes a read transaction.

Implements amba_pv::amba_pv_if< 64 >.

7.37.3.3 read() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::read (
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int size,
```

```
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline]
```

Completes a read transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes.
size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to NULL if unused on the master side).
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

7.37.3.4 write() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::write (
            int index,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int size,
            const amba_pv_control * ctrl,
            unsigned char * strb,
            sc_core::sc_time & t )  [inline], [virtual]
```

Completes a write transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.37.3.5 write() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::write (
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int size,
            const amba_pv_control * ctrl,
            unsigned char * strb,
            sc_core::sc_time & t )  [inline]
```

Completes a write transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes.
size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to NULL if unused on the master side).
strb	write strobes pointer (set to NULL if none). It must point to an array of *size* elements.
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

7.37.3.6 burst_read() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::burst_read (
              int index,
              const sc_dt::uint64 & addr,
              unsigned char * data,
              unsigned int length,
              unsigned int size,
              const amba_pv_control * ctrl,
              amba_pv_burst_t burst,
              sc_core::sc_time & t )  [inline], [virtual]
```

Completes a burst read transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.37.3.7 burst_read() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::burst_read (
              const sc_dt::uint64 & addr,
              unsigned char * data,
              unsigned int length,
              unsigned int size,
              const amba_pv_control * ctrl,
              amba_pv_burst_t burst,
              sc_core::sc_time & t )  [inline]
```

Completes a burst read transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of (*size* ∗ *length*) bytes.
length	transaction burst length as in [1-16].
size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to NULL if unused on the master side).
burst	transaction burst type, one of AMBA_PV_INCR, AMBA_PV_FIXED, or AMBA_PV_WRAP.
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

7.37.3.8 burst_write() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::burst_write (
              int index,
              const sc_dt::uint64 & addr,
              unsigned char * data,
```

```
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst,
            unsigned char * strb,
            unsigned int strb_length,
            sc_core::sc_time & t )  [inline], [virtual]
```

Completes a burst write transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.37.3.9 burst_write() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::burst_write (
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst,
            unsigned char * strb,
            unsigned int strb_length,
            sc_core::sc_time & t )  [inline]
```

Completes a burst write transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of (*size* ∗ *length*) bytes.
length	transaction burst length as in [1-16].
size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128]. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to NULL if unused on the master side).
burst	transaction burst type, one of AMBA_PV_INCR, AMBA_PV_FIXED, or AMBA_PV_WRAP.
strb	write strobes pointer (set to NULL if none).
strb_length	Write strobes length. It must be a multiple of *size*.
t	timing annotation.

Returns

AMBA_PV_OKAY if the transaction is successful.

7.37.3.10 get_direct_mem_ptr() [1/4]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
bool amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::get_direct_mem_ptr (
            int index,
            tlm::tlm_command command,
            const sc_dt::uint64 & addr,
            const amba_pv_control * ctrl,
            tlm::tlm_dmi & dmi_data )  [inline], [virtual]
```

Requests DMI access to the specified address and returns a reference to a DMI descriptor.
Implements amba_pv::amba_pv_if< 64 >.

7.37.3.11 get_direct_mem_ptr() [2/4]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
bool amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::get_direct_mem_ptr (
            tlm::tlm_command command,
            const sc_dt::uint64 & addr,
            const amba_pv_control * ctrl,
            tlm::tlm_dmi & dmi_data )  [inline]
```

Requests DMI access to the specified address and returns a reference to a DMI descriptor.

Parameters

command	`tlm::TLM_READ_COMMAND` for a DMI read access request, `tlm::TLM_WRITE_COMMAND` for a DMI write access request.
addr	address to which the DMI access is requested.
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).
dmi_data	returned DMI descriptor.

Returns

`true` if a DMI region is granted, `false` otherwise.

7.37.3.12 debug_read() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
unsigned int amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::debug_read (
            int index,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            const amba_pv_control * ctrl )  [inline], [virtual]
```

Non-intrusive debug read transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.37.3.13 debug_write() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
unsigned int amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::debug_write (
            int index,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            const amba_pv_control * ctrl )  [inline], [virtual]
```

Non-intrusive debug write transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.37.3.14 debug_read() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
unsigned int amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::debug_read (
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            const amba_pv_control * ctrl )  [inline]
```

Non-intrusive debug read transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of *length* bytes.
length	transaction length.
ctrl	AMBA 3 control information (set to NULL if unused on the master side).

Returns

number of bytes read or, if error, 0.

7.37.3.15 debug_write() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
unsigned int amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::debug_write (
             const sc_dt::uint64 & addr,
             unsigned char * data,
             unsigned int length,
             const amba_pv_control * ctrl )  [inline]
```

Non-intrusive debug write transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of *length* bytes.
length	transaction length.
ctrl	AMBA 3 control information (set to NULL if unused on the master side).

Returns

number of bytes written or, if error, 0.

7.37.3.16 atomic_store() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::atomic_store (
             int socket_id,
             const sc_dt::uint64 & addr,
             unsigned char * data,
             unsigned int length,
             unsigned int size,
             const amba_pv_control * ctrl,
             amba_pv_atomic_subop_t subop,
             amba_pv_atomic_endianness_t endianness,
             sc_core::sc_time & t )  [inline], [virtual]
```

Completes an atomic store transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.37.3.17 atomic_store() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::atomic_store (
             const sc_dt::uint64 & addr,
```

```
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time & t )  [inline]
```

Completes an atomic store transaction.

Parameters

addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes.
length	sets the data transfers in a transaction.
size	sets the transaction size in bytes. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).
subop	operation type of the atomic transaction.
endianness	endianness of the atomic operation. Data is interpreted in big endian order if enabled.
t	timing annotation.

Returns

`AMBA_PV_OKAY` if the transaction is successful.

Note

Byte enable is unsupported for atomic transactions.

The product of `size` and `length` must be one of [1, 2, 4, 8].

7.37.3.18 atomic_load() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::atomic_load (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time & t )  [inline], [virtual]
```

Completes an atomic load transaction.

Implements amba_pv::amba_pv_if< 64 >.

7.37.3.19 atomic_load() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::atomic_load (
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
```

```
            const amba_pv_control * ctrl,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time & t )  [inline]
```

Completes an atomic load transaction.

Parameters

socket_id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes. The array initially contains the sending data, then the original data at the address before the atomic operation is returned to the array.
length	sets the data transfers in a transaction.
size	sets the transaction size in bytes. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).
subop	operation type of the atomic transaction.
endianness	endianness of the atomic operation. Data is interpreted in big endian order if enabled.
t	timing annotation.

Returns

`AMBA_PV_OKAY` if the transaction is successful.

Note

Byte enable is unsupported for atomic transactions.

The product of `size` and `length` must be one of [1, 2, 4, 8].

7.37.3.20 atomic_swap() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::atomic_swap (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline], [virtual]
```

Completes an atomic swap transaction.

Implements amba_pv::amba_pv_if< 64 >.

7.37.3.21 atomic_swap() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::atomic_swap (
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline]
```

Completes an atomic swap transaction.

Parameters

socket↩ _id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.
data	transaction data pointer. It must point to an array of *size* bytes. The array initially contains the sending data, then the original data at the address before the atomic operation is returned to the array.
length	sets the data transfers in a transaction.
size	sets the transaction size in bytes. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).
t	timing annotation.

Returns

`AMBA_PV_OKAY` if the transaction is successful.

Note

Byte enable is unsupported for atomic transactions.

The product of `size` and `length` must be one of [1, 2, 4, 8].

7.37.3.22 atomic_compare() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::atomic_compare (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline], [virtual]
```

Completes an atomic compare transaction.

Implements amba_pv::amba_pv_if< 64 >.

7.37.3.23 atomic_compare() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv_resp_t amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::atomic_compare (
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline]
```

Completes an atomic compare transaction.

Parameters

socket↩ _id	socket identifier (index into bound interfaces on the master side).
addr	transaction address.

Parameters

data	transaction data pointer. It must point to an array of *size* bytes. The array initially comprises of comarping and swapping data, after the transaction, the original data at the address before the atomic operation is returned to the array.
length	sets the data transfers in a transaction.
size	sets the transaction size in bytes covering both comparing and swapping data. The transaction size must be less than or equal to the value returned by get_bus_width_bytes().
ctrl	AMBA 3 control information (set to `NULL` if unused on the master side).
t	timing annotation.

Returns

`AMBA_PV_OKAY` if the transaction is successful.

Note

Transaction data pointer points to an array comprising of comparing and swapping data, with their order determined by the transaction address. Comparing data is sent first if the address points to the lower half of the transaction (i.e. lowest address byte); swapping data is sent first if the address points to the upper half of the transaction (i.e. lowest address plus half the ∗*size*∗).

The original data at the address is returned to *data* pointer starting from the lowest byte. The returning size is half the sending *size*.

Byte enable is unsupported for atomic transactions.

The product of `size` and `length` must be one of [2, 4, 8, 16, 32] while `size` must be larger than 1.

7.37.3.24 b_transport() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::b_transport (
            int index,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline]
```

Blocking transport.
This version of the method forwards the b_transport() call to the *index'ed* slave socket bound to this master socket.

Parameters

index	interface index (for sockets bound more than once).
trans	transaction.
t	timing annotation.

7.37.3.25 b_transport() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::b_transport (
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline]
```

Blocking transport.

Parameters

trans	transaction.

Parameters

t	timing annotation.

7.37.3.26 transport_dbg() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
unsigned int amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::transport_dbg (
            int index,
            amba_pv_transaction & trans )  [inline]
```

Debug access to a target.
This version of the method forwards the transport_dbg() call to the *index'ed* slave socket bound to this master socket.

Parameters

index	interface index (for sockets bound more than once).
trans	transaction.

Returns

number of bytes read or written or, if error, 0.

7.37.3.27 transport_dbg() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
unsigned int amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::transport_dbg (
            amba_pv_transaction & trans )  [inline]
```

Debug access to a target.

Parameters

trans	transaction.

Returns

number of bytes read or written or, if error, 0.

7.37.3.28 get_direct_mem_ptr() [3/4]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
bool amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::get_direct_mem_ptr (
            int index,
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [inline]
```

Requests a DMI access based on the specified transaction.
This version of the method forwards the get_direct_mem_ptr() call to the *index'ed* slave socket bound to this master socket.

Parameters

index	interface index (for sockets bound more than once).
trans	transaction.
dmi_data	DMI Descriptor.

Returns

`true` if DMI access is granted, `false` otherwise.

7.37.3.29 get_direct_mem_ptr() [4/4]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
bool amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >::get_direct_mem_ptr (
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [inline]
```

Requests a DMI access based on the specified transaction.

Parameters

trans	transaction.
dmi_data	DMI Descriptor.

Returns

`true` if DMI access is granted, `false` otherwise.

7.38 amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR > Class Template Reference

AMBA-PV advanced memory model.
`#include <models/amba_pv_memory.h>`
Inherits amba_pv::amba_pv_memory_base< 64 >, and sc_core::sc_module.

Public Member Functions

- amba_pv_memory (const sc_core::sc_module_name &, const sc_dt::uint64 &, unsigned int=4096)
  Constructor.
- amba_pv_memory (const sc_core::sc_module_name &, const sc_dt::uint64 &, unsigned char, unsigned int=4096)
  Constructor.
- ~amba_pv_memory ()
  Destructor.
- virtual const char * kind () const
  Returns the kind string of this memory.
- unsigned int get_page_size () const
  Returns the page size of this memory.
- void set_fill_pattern (unsigned char, unsigned char)
  Sets the fill pattern used for uninitialized memory.
- void set_fill_pattern32 (unsigned long, unsigned long)
  Sets the fill pattern used for uninitialized memory.
- void save (const std::string &)
  Saves the contents of this memory to the specified file.
- void save (std::ostream &)
  Saves the contents of this memory to the specified stream.
- void restore (const std::string &)
  Restore the contents of this memory from the specified file.
- void restore (std::istream &)
  Restore the contents of this memory from the specified stream.

Data Fields

- amba_pv_slave_socket< BUSWIDTH > amba_pv_s

  Slave socket.

Protected Member Functions

- virtual amba_pv_resp_t read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes a read transaction.
- virtual amba_pv_resp_t write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, unsigned char *, sc_core::sc_time &)

  Completes a write transaction.
- virtual bool get_direct_mem_ptr (int, tlm::tlm_command, const sc_dt::uint64 &, const amba_pv_control *, tlm::tlm_dmi &)

  Requests DMI access to the specified address and returns a reference to a DMI descriptor.
- virtual unsigned int debug_read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug read transaction.
- virtual unsigned int debug_write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug write transaction.
- virtual amba_pv_resp_t atomic_store (int, const sc_dt::uint64 &, unsigned char *const, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic store transaction.
- virtual amba_pv_resp_t atomic_load (int, const sc_dt::uint64 &, unsigned char *const, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic load transaction.
- virtual amba_pv_resp_t atomic_compare (int, const sc_dt::uint64 &, unsigned char *const, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic compare.
- virtual amba_pv_resp_t atomic_swap (int, const sc_dt::uint64 &, unsigned char *const, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic swap.

7.38.1 Detailed Description

template<unsigned int BUSWIDTH = 64, class ALLOCATOR = amba_pv_heap_allocator>
class amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >

AMBA-PV advanced memory model.
The amba_pv_memory class models an AMBA compatible memory at the PV level and features:

- page-based organization with allocate on write policy for optimized heap usage
- a read to non-allocated pages returns the default value
- constructor parameter for page size
- save and restore.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 amba_pv_memory() [1/2]

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::amba_pv_memory (
            const sc_core::sc_module_name & name,
            const sc_dt::uint64 & size,
            unsigned int page_size = 4096 )  [inline]
```

Constructor.

Parameters

name	memory name.
size	memory size in bytes. *size* is rounded up to the next multiple of 4096.
page_size	memory page size in bytes. Defaults to 4096. *page_size* is rounded up to the next multiple of 4096.

7.38.2.2 amba_pv_memory() [2/2]

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::amba_pv_memory (
            const sc_core::sc_module_name & name,
            const sc_dt::uint64 & size,
            unsigned char fill_char,
            unsigned int page_size = 4096 )  [inline]
```

Constructor.

Parameters

name	memory name.
size	memory size in bytes. *size* is rounded up to the next multiple of 4096.
fill_char	fill character used for uninitialized memory.
page_size	memory page size in bytes. Defaults to 4096. *page_size* is rounded up to the next multiple of 4096.

Note

This constructor is deprecated. Use the other constructor instead.

7.38.2.3 ~amba_pv_memory()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::~amba_pv_memory  [inline]
```

Destructor.

7.38.3 Member Function Documentation

7.38.3.1 kind()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
const char * amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::kind  [inline], [virtual]
```

Returns the kind string of this memory.

7.38.3.2 get_page_size()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
unsigned int amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::get_page_size  [inline]
```

Returns the page size of this memory.

7.38.3.3 set_fill_pattern()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
void amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::set_fill_pattern (
            unsigned char fill_char1,
            unsigned char fill_char2 )  [inline]
```

Sets the fill pattern used for uninitialized memory.
This does not affect any pages that have already been allocated, so ideally this must be called before the first write transaction to this memory.

Parameters

fill_char1	value for odd characters in uninitialized memory.
fill_char2	value for even characters in uninitialized memory.

7.38.3.4 set_fill_pattern32()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
void amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::set_fill_pattern32 (
            unsigned long fill_word1,
            unsigned long fill_word2 )  [inline]
```

Sets the fill pattern used for uninitialized memory.
This does not affect any pages that have already been allocated, so ideally this must be called before the first write transaction to this memory. A little endian memory model is assumed. i.e. the least significant byte of fill_word1 is the first byte in a page of memory.

Parameters

fill_word1	value for odd words in uninitialized memory.
fill_word2	value for even words in uninitialized memory.

7.38.3.5 save() [1/2]

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
void amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::save (
            const std::string & file )  [inline]
```

Saves the contents of this memory to the specified *file*.

Parameters

file	name of file to save memory contents to.

7.38.3.6 save() [2/2]

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
void amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::save (
            std::ostream & os )  [inline]
```

Saves the contents of this memory to the specified stream.

Parameters

os	stream to save memory contents to.

7.38.3.7 restore() [1/2]

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
void amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::restore (
            const std::string & file )  [inline]
```

Restore the contents of this memory from the specified *file*.

Parameters

file	name of file to restore this memory contents from.

7.38.3.8 restore() [2/2]

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
void amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::restore (
            std::istream & is )  [inline]
```

Restore the contents of this memory from the specified stream.

Parameters

is	stream to restore this memory contents from.

7.38.3.9 read()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
amba_pv_resp_t amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::read (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes a read transaction.

Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.38.3.10 write()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
amba_pv_resp_t amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::write (
            int socket_id,
```

```
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int size,
            const amba_pv_control * ctrl,
            unsigned char * strb,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes a write transaction.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.38.3.11 get_direct_mem_ptr()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
bool amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::get_direct_mem_ptr (
            int socket_id,
            tlm::tlm_command command,
            const sc_dt::uint64 & addr,
            const amba_pv_control * ctrl,
            tlm::tlm_dmi & dmi_data )  [inline], [protected], [virtual]
```

Requests DMI access to the specified address and returns a reference to a DMI descriptor.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.38.3.12 debug_read()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
unsigned int amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::debug_read (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            const amba_pv_control * ctrl )  [inline], [protected], [virtual]
```

Non-intrusive debug read transaction.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.38.3.13 debug_write()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
unsigned int amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::debug_write (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            const amba_pv_control * ctrl )  [inline], [protected], [virtual]
```

Non-intrusive debug write transaction.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.38.3.14 atomic_store()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
amba_pv_resp_t amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::atomic_store (
            int ,
            const sc_dt::uint64 & addr,
            unsigned char * const data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ,
```

```
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time &  )  [inline], [protected], [virtual]
```

Completes an atomic store transaction.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.38.3.15 atomic_load()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
amba_pv_resp_t amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::atomic_load (
            int ,
            const sc_dt::uint64 & addr,
            unsigned char * const data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time &  )  [inline], [protected], [virtual]
```

Completes an atomic load transaction.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.38.3.16 atomic_compare()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
amba_pv_resp_t amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::atomic_compare (
            int ,
            const sc_dt::uint64 & addr,
            unsigned char * const data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes an atomic compare.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.38.3.17 atomic_swap()

```
template<unsigned int BUSWIDTH, class ALLOCATOR >
amba_pv_resp_t amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::atomic_swap (
            int ,
            const sc_dt::uint64 & addr,
            unsigned char * const data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ,
            sc_core::sc_time &  )  [inline], [protected], [virtual]
```

Completes an atomic swap.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.38.4 Field Documentation

7.38.4.1 amba_pv_s

```
template<unsigned int BUSWIDTH = 64, class ALLOCATOR = amba_pv_heap_allocator>
amba_pv_slave_socket<BUSWIDTH> amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >::amba_pv_s
```

Slave socket.

7.39 amba_pv::amba_pv_memory_base< BUSWIDTH > Class Template Reference

AMBA-PV memory model base class.

`#include <models/amba_pv_memory_base.h>`

Inherits amba_pv::amba_pv_slave_base< 64 >.

Public Member Functions

- amba_pv_memory_base (const std::string &, const sc_dt::uint64 &)

  Constructor.
- sc_dt::uint64 get_addr_limit () const

  Returns the address limit (this is, the first not allowed address) of this memory.

Protected Member Functions

- virtual void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking transport.

7.39.1 Detailed Description

template< unsigned int BUSWIDTH = 64 >
class amba_pv::amba_pv_memory_base< BUSWIDTH >

AMBA-PV memory model base class.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.39.2 Constructor & Destructor Documentation

7.39.2.1 amba_pv_memory_base()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_memory_base< BUSWIDTH >::amba_pv_memory_base (
            const std::string & name,
            const sc_dt::uint64 & size )  [inline]
```

Constructor.

Parameters

name	memory name.
size	memory size in bytes. *size* is rounded up to the next multiple of 4096.

7.39.3 Member Function Documentation

7.39.3.1 get_addr_limit()

```
template<unsigned int BUSWIDTH>
sc_dt::uint64 amba_pv::amba_pv_memory_base< BUSWIDTH >::get_addr_limit  [inline]
```

Returns the address limit (this is, the first not allowed address) of this memory.

7.39.3.2 b_transport()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_memory_base< BUSWIDTH >::b_transport (
            int socket_id,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Blocking transport.
This version of the method sets the DMI allowed attribute to `true` for non-exclusive accesses and forward the b_transport() call to the base class.
Reimplemented from amba_pv::amba_pv_slave_base< 64 >.

7.40 amba_pv::amba_pv_protocol_checker< BUSWIDTH > Class Template Reference

AMBA-PV protocol checker model.
`#include <models/amba_pv_protocol_checker.h>`
Inherits amba_pv::amba_pv_fw_transport_if, amba_pv::amba_pv_bw_transport_if, and amba_pv::amba_pv_protocol_checker_base<

Public Member Functions

- amba_pv_protocol_checker (const sc_core::sc_module_name &, bool=true)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this protocol checker.

Data Fields

- amba_pv_slave_socket< BUSWIDTH > amba_pv_s

  Slave socket.
- amba_pv_master_socket< BUSWIDTH > amba_pv_m

  Master socket.

Protected Member Functions

- virtual void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking transport.
- virtual unsigned int transport_dbg (int, amba_pv_transaction &)

  Debug access to a target.
- virtual bool get_direct_mem_ptr (int, amba_pv_transaction &, tlm::tlm_dmi &)

  Requests a DMI access based on the specified transaction.
- virtual void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)

  Invalidates DMI pointers previously established for the specified DMI region.

7.40.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_protocol_checker< BUSWIDTH >

AMBA-PV protocol checker model.
The amba_pv_protocol_checker model is used for confirming that your model complies with the AMBA-PV protocol. You can instantiate the protocol checker between any pair of AMBA-PV master and slave sockets. The transactions that pass through are checked against the AMBA-PV protocol and errors reported using the SystemC reporting mechanism. All errors are reported with a message type of "amba_pv_protocol_checker" and with a severity of SC_ERROR. Recommendations are reported with a severity of SC_WARNING.
You can configure the the protocol checker to specifically test your model against one of the AXI, AHB or APB protocols.

Note

The AMBA-PV protocol checker model does not perform any OSCI TLM 2.0 BP checks.

The AMBA-PV protocol checker model might have an effect on performance.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.40.2 Constructor & Destructor Documentation

7.40.2.1 amba_pv_protocol_checker()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_protocol_checker< BUSWIDTH >::amba_pv_protocol_checker (
            const sc_core::sc_module_name & name,
            bool recommend_on = true )  [inline], [explicit]
```

Constructor.
Constructs a new amba_pv_protocol_checker with parameter for configuring recommended rules.

Parameters

name	protocol checker name.
recommend_on	`true` to enable reporting of protocol recommendations, `false` otherwise.

See also

recommend_on()

7.40.3 Member Function Documentation

7.40.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_protocol_checker< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string of this protocol checker.

7.40.3.2 b_transport()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_protocol_checker< BUSWIDTH >::b_transport (
             int socket_id,
             amba_pv_transaction & trans,
             sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Blocking transport.
This version of the method completes the transaction and checks it complies with the AMBA buses protocols.
Implements amba_pv::amba_pv_fw_transport_if.

7.40.3.3 transport_dbg()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_protocol_checker< BUSWIDTH >::transport_dbg (
             int socket_id,
             amba_pv_transaction & trans )  [inline], [protected], [virtual]
```

Debug access to a target.
This version of the method forwards this debug access to the slave and checks it complies with the AMBA buses protocols.
Implements amba_pv::amba_pv_fw_transport_if.

7.40.3.4 get_direct_mem_ptr()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_protocol_checker< BUSWIDTH >::get_direct_mem_ptr (
             int socket_id,
             amba_pv_transaction & trans,
             tlm::tlm_dmi & dmi_data )  [inline], [protected], [virtual]
```

Requests a DMI access based on the specified transaction.
This version of the method forwards this DMI access request to the slave and checks it complies with the AMBA buses protocols.
Implements amba_pv::amba_pv_fw_transport_if.

7.40.3.5 invalidate_direct_mem_ptr()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_protocol_checker< BUSWIDTH >::invalidate_direct_mem_ptr (
             int socket_id,
             sc_dt::uint64 start_range,
             sc_dt::uint64 end_range )  [inline], [protected], [virtual]
```

Invalidates DMI pointers previously established for the specified DMI region.
This version of the method forwards this DMI call to the master.
Implements amba_pv::amba_pv_bw_transport_if.

7.40.4 Field Documentation

7.40.4.1 amba_pv_s

```
template<unsigned int BUSWIDTH = 64>
amba_pv_slave_socket<BUSWIDTH> amba_pv::amba_pv_protocol_checker< BUSWIDTH >::amba_pv_s
```

Slave socket.

7.40.4.2 amba_pv_m

```
template<unsigned int BUSWIDTH = 64>
amba_pv_master_socket<BUSWIDTH> amba_pv::amba_pv_protocol_checker< BUSWIDTH >::amba_pv_m
```

Master socket.

7.41 amba_pv::amba_pv_protocol_checker_base< BUSWIDTH > Class Template Reference

AMBA-PV protocol checker base model.

`#include <models/amba_pv_protocol_checker_base.h>`

Inherits sc_core::sc_module.

Public Member Functions

- amba_pv_protocol_checker_base (const sc_core::sc_module_name &, bool recommend_on, amba_pv_protocol_t protocol)

  Constructor.
- virtual ~amba_pv_protocol_checker_base ()

  Destructor.
- void recommend_on (bool=true)

  Enables or disables reporting of protocol recommendations.
- virtual void check_protocol (amba_pv_protocol_t protocol)

  Selects the AMBA protocol checks to perform.

Protected Member Functions

- void atomic_checks (const amba_pv_transaction &, const amba_pv_extension *)

  Performs atomic checks.

7.41.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_protocol_checker_base< BUSWIDTH >

AMBA-PV protocol checker base model.

The amba_pv_protocol_checker_base model is used for confirming that your model complies with the AMBA-PV protocol.

You can instantiate the protocol checker between any pair of AMBA-PV master and slave sockets. The transactions that pass through are checked against the AMBA-PV protocol and errors reported using the SystemC reporting mechanism. All errors are reported with a message type of "`amba_pv_protocol_checker_base`" and with a severity of `SC_ERROR`. Recommendations are reported with a severity of `SC_WARNING`.

You can configure the the protocol checker to specifically test your model against one of the ACE, AXI, AHB or APB protocols.

Note

The AMBA-PV protocol checker model does not perform any TLM 2.0 BP checks.

The AMBA-PV protocol checker model might have an effect on performance.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.41.2 Constructor & Destructor Documentation

7.41.2.1 amba_pv_protocol_checker_base()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_protocol_checker_base< BUSWIDTH >::amba_pv_protocol_checker_base (
            const sc_core::sc_module_name & name,
            bool recommend_on,
            amba_pv_protocol_t protocol )  [inline], [explicit]
```

Constructor.
Constructs a new amba_pv_protocol_checker_base with parameter for configuring recommended rules.

Parameters

name	protocol checker name.
recommend_on	`true` to enable reporting of protocol recommendations, `false` otherwise.
protocol	selected AMBA protocol checks as one of `AMBA_PV_APB`, `AMBA_PV_AHB`, `AMBA_PV_AXI3`, `AMBA_PV_AXI4_LITE`, `AMBA_PV_AXI4`, `AMBA_PV_ACE_LITE`, `AMBA_PV_ACE`

See also

recommend_on()

7.41.2.2 ~amba_pv_protocol_checker_base()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_protocol_checker_base< BUSWIDTH >::~amba_pv_protocol_checker_base  [inline],
[virtual]
```

Destructor.

7.41.3 Member Function Documentation

7.41.3.1 recommend_on()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_protocol_checker_base< BUSWIDTH >::recommend_on (
            bool recommend_on = true )  [inline]
```

Enables or disables reporting of protocol recommendations.
If *recommend_on* is set to `false`, no recommendations are reported and the following warning is issued:
`Warning: amba_pv_protocol_checker_base: All AMBA-PV recommended rules have been disabled by recommend_on()`

Parameters

recommend_on	`true` to enable reporting of recommendations (default), `false` otherwise.

7.41.3.2 check_protocol()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_protocol_checker_base< BUSWIDTH >::check_protocol (
            amba_pv_protocol_t protocol )  [inline], [virtual]
```

Selects the AMBA protocol checks to perform.
The protocol checker tests your model against the selected AMBA protocol.
If *protocol* is set to anything other than AMBA_PV_AXI3, transactions are checked against that protocol and the following warning is issued:

```
Warning: amba_pv_protocol_checker_base: AMBA_PV_protocol protocol rules have been selected by
      check_protocol()
```

Where AMBA_PV_protocol is the selected protocol.

Parameters

protocol	selected AMBA protocol checks as one of AMBA_PV_APB, AMBA_PV_AHB, AMBA_PV_AXI3, AMBA_PV_AXI4_LITE, AMBA_PV_AXI4, AMBA_PV_ACE_LITE, AMBA_PV_ACE

7.41.3.3 atomic_checks()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_protocol_checker_base< BUSWIDTH >::atomic_checks (
            const amba_pv_transaction & trans,
            const amba_pv_extension * ex )  [inline], [protected]
```

Performs atomic checks.
This method runs a number of sanity checks for atomic transfers. If any of those checks fails, an error message is output explaining what went wrong. The method may abort early if the atomic access transaction parameter is a NULL pointer or the atomic operation is set to amba_pv_atomic_op_t::AMBA_PV_NONATOMIC.

Parameters

trans	specifies the transaction parameters
ex	specifies the atomic access transaction parameters

7.42 amba_pv::amba_pv_protocol_types Struct Reference

AMBA-PV protocol types.
`#include <core/amba_pv_types.h>`

7.42.1 Detailed Description

AMBA-PV protocol types.
This structure defines the payload and phase types for AMBA-PV. It is used for the TYPES template parameter with OSCI TLM 2.0 classes and interfaces.
If using amba_pv_protocol_types with OSCI TLM 2.0 classes and interfaces, the following rules apply to the OSCI TLM 2.0 GP attributes:

- The data length attribute must be greater than or equal to the burst size times the burst length. If not, an error response of tlm::TLM_BURST_ERROR_REPONSE is returned.
- The streaming width attribute must be equal to the burst size for a fixed burst. If not, an error response of tlm::TLM_BURST_ERROR_REPONSE is returned.
- The byte enable pointer attribute must be NULL on read transactions. If not, an error response of tlm::↩ TLM_BYTE_ENABLE_ERROR_REPONSE is returned.
- The byte enable length attribute must be a multiple of the burst size on write transactions. If not, an error response of tlm::TLM_BYTE_ENABLE_ERROR_REPONSE is returned.
- If the address attribute is not aligned on the burst size, only the address of the first burst beat must be unaligned, the addresses of subsequent data transfers being aligned.

Note

This does not enforce any requirements on slaves for read transactions. This must be represented with appropriate byte enables for write transactions.

AMBA 3 buses specific signals are defined in an extension class.

See also

amba_pv_extension

7.43 amba_pv::amba_pv_response Class Reference

AMBA-PV response class.
`#include <bus/amba_pv_response.h>`

Public Member Functions

- amba_pv_response ()
  Default constructor.
- amba_pv_response (amba_pv_resp_t)
  Constructor.
- void set_resp (amba_pv_resp_t)
  Sets transaction response.
- amba_pv_resp_t get_resp () const
  Returns transaction response.
- bool is_incomplete () const
  Returns wether or not the response is incomplete.
- void set_incomplete ()
  Sets the response to incomplete.
- bool is_okay () const
  Returns wether or not the `OKAY` response is set.
- void set_okay ()
  Sets the `OKAY` response.
- bool is_exokay () const
  Returns wether or not the response is `EXOKAY`.
- void set_exokay ()
  Sets the `EXOKAY` response.
- bool is_slverr () const
  Returns wether or not the response is `SLVERR`.
- void set_slverr ()
  Sets the `SLVERR` response.
- bool is_decerr () const
  Returns wether or not the response is `DECERR`.
- void set_decerr ()
  Sets the `DECERR` response.
- bool is_pass_dirty () const
  Returns wether or not the `PassDirty` response bit is set.
- void set_pass_dirty (bool=true)
  Sets the `PassDirty` response bit.
- bool is_shared () const
  Returns wether or not the `IsShared` response bit is set.
- void set_shared (bool=true)
  Sets the `IsShared` response bit.

- bool is_snoop_data_transfer () const

  Returns wether or not the `DataTransfer` snoop response bit is set.
- void set_snoop_data_transfer (bool=true)

  Sets the `DataTransfer` snoop response bit.
- bool is_snoop_error () const

  Returns wether or not the `Error` snoop response bit is set.
- void set_snoop_error (bool=true)

  Sets the `Error` snoop response bit.
- bool is_snoop_was_unique () const

  Returns wether or not the `WasUnique` snoop response bit is set.
- void set_snoop_was_unique (bool=true)

  Sets the `WasUnique` snoop response bit.
- void reset ()

  Resets all members of this AMBA-PV extension to their default value.

7.43.1 Detailed Description

AMBA-PV response class.

7.43.2 Constructor & Destructor Documentation

7.43.2.1 amba_pv_response() [1/2]

```
amba_pv::amba_pv_response::amba_pv_response ( )  [inline]
```

Default constructor.
By default:

- the response is initialized to `AMBA_PV_INCOMPLETE`.

7.43.2.2 amba_pv_response() [2/2]

```
amba_pv::amba_pv_response::amba_pv_response (
            amba_pv_resp_t resp )  [inline]
```

Constructor.

Parameters

resp	AMBA-PV transaction response

7.43.3 Member Function Documentation

7.43.3.1 set_resp()

```
void amba_pv::amba_pv_response::set_resp (
            amba_pv_resp_t resp )  [inline]
```

Sets transaction response.

Parameters

resp	transaction response.

See also

get_resp(), set_incomplete(), set_okay(), set_exokay(), set_slverr(), set_decerr(), set_pass_dirty(), set_shared()

7.43.3.2 get_resp()

`amba_pv_resp_t amba_pv::amba_pv_response::get_resp ( ) const [inline]`

Returns transaction response.

See also

set_resp(), is_incomplete(), is_okay(), is_exokay(), is_slverr(), is_decerr(), is_pass_dirty(), is_shared()

7.43.3.3 is_incomplete()

`bool amba_pv::amba_pv_response::is_incomplete ( ) const [inline]`

Returns wether or not the response is incomplete.
An incomplete reponse indicates that the slave did not attempt to perform the access.

Returns

`true` if the response is incomplete, `false` otherwise.

See also

set_incomplete()

7.43.3.4 set_incomplete()

`void amba_pv::amba_pv_response::set_incomplete ( ) [inline]`

Sets the response to incomplete.

See also

is_incomplete()

7.43.3.5 is_okay()

`bool amba_pv::amba_pv_response::is_okay ( ) const [inline]`

Returns wether or not the `OKAY` response is set.
The `OKAY` reponse indicates if a normal access has been successful. It indicates also an exclusive access failure.

Returns

`true` if the `OKAY` response is set, `false` otherwise.

See also

set_okay()

7.43.3.6 set_okay()

`void amba_pv::amba_pv_response::set_okay ( ) [inline]`

Sets the `OKAY` response.

See also

is_okay(), set_pass_dirty(), set_shared()

7.43.3.7 is_exokay()

```
bool amba_pv::amba_pv_response::is_exokay ( ) const  [inline]
```

Returns wether or not the response is EXOKAY.
If true, the EXOKAY response indicates that either the read or write portion of an exclusive access has been successful.

Returns

true if the response is EXOKAY, false otherwise.

See also

set_exokay(), is_pass_dirty(), is_shared()

7.43.3.8 set_exokay()

```
void amba_pv::amba_pv_response::set_exokay ( )  [inline]
```

Sets the EXOKAY response.
The PassDirty and IsShared response flags will be cleared.

See also

is_exokay(), set_pass_dirty(), set_shared()

7.43.3.9 is_slverr()

```
bool amba_pv::amba_pv_response::is_slverr ( ) const  [inline]
```

Returns wether or not the response is SLVERR.
The SLVERR response is used if the access has reached the slave successfully, but the slave returned an error condition to the originating master.

Returns

true if the response is SLVERR, false otherwise.

See also

set_slverr()

7.43.3.10 set_slverr()

```
void amba_pv::amba_pv_response::set_slverr ( )  [inline]
```

Sets the SLVERR response.

See also

is_slverr()

7.43.3.11 is_decerr()

```
bool amba_pv::amba_pv_response::is_decerr ( ) const  [inline]
```

Returns wether or not the response is DECERR.
The DECERR response is generated typically by an interconnect component to indicate that there is no slave at the transaction address.

Returns

`true` if the response is `DECERR`, `false` otherwise.

See also

set_decerr()

7.43.3.12 set_decerr()

```
void amba_pv::amba_pv_response::set_decerr ( )  [inline]
```

Sets the `DECERR` response.

See also

is_decerr()

7.43.3.13 is_pass_dirty()

```
bool amba_pv::amba_pv_response::is_pass_dirty ( ) const  [inline]
```

Returns wether or not the `PassDirty` response bit is set.
The `PassDirty` response bit indicates the cache line is dirty with respect to main memory. For ACE this bit is a part of both read and snoop responses.

Returns

`true` if the `PassDirty` bit is set, `false` otherwise.

See also

set_pass_dirty(), set_okay(), set_exokay()

7.43.3.14 set_pass_dirty()

```
void amba_pv::amba_pv_response::set_pass_dirty (
            bool pass_dirty = true )  [inline]
```

Sets the `PassDirty` response bit.
The `PassDirty` response bit indicates the cache line is dirty with respect to main memory. For ACE this bit is a part of both read and snoop responses.

Parameters

pass_dirty	status of `PassDirty` bit

See also

is_pass_dirty(), is_okay(), is_exokay()

7.43.3.15 is_shared()

```
bool amba_pv::amba_pv_response::is_shared ( ) const  [inline]
```

Returns wether or not the `IsShared` response bit is set.
The `IsShared` response bit hints that another copy of the data might be held in another cache. For ACE this bit is a part of both read and snoop responses.

Returns

`true` if the `IsShared` bit is set, `false` otherwise.

See also

set_shared(), set_okay(), set_exokay()

7.43.3.16 set_shared()

```
void amba_pv::amba_pv_response::set_shared (
            bool is_shared = true )  [inline]
```

Sets the `IsShared` response bit.
The `IsShared` response bit hints that another copy of the data might be held in another cache. For ACE this bit is a part of both read and snoop responses.

Parameters

is_shared	status of `IsShared` bit

See also

is_shared(), is_okay(), is_exokay()

7.43.3.17 is_snoop_data_transfer()

```
bool amba_pv::amba_pv_response::is_snoop_data_transfer ( ) const  [inline]
```

Returns wether or not the `DataTransfer` snoop response bit is set.
The `DataTransfer` response bit indicates that a full cache line of data will be provided on the snoop data channel for this transaction.

Returns

`true` if the `DataTransfer` bit is set, `false` otherwise.

See also

set_snoop_data_transfer()

7.43.3.18 set_snoop_data_transfer()

```
void amba_pv::amba_pv_response::set_snoop_data_transfer (
            bool data_transfer = true )  [inline]
```

Sets the `DataTransfer` snoop response bit.
The `DataTransfer` response bit indicates that a full cache line of data will be provided on the snoop data channel for this transaction.

Parameters

data_transfer	status of `DataTransfer` bit

See also

is_snoop_data_transfer()

7.43.3.19 is_snoop_error()

```
bool amba_pv::amba_pv_response::is_snoop_error ( ) const  [inline]
```

Returns wether or not the `Error` snoop response bit is set.
The `Error` response bit indicates that the snooped cache line is in error.

Returns

`true` if the `Error` bit is set, `false` otherwise.

See also

set_snoop_error()

7.43.3.20 set_snoop_error()

```
void amba_pv::amba_pv_response::set_snoop_error (
            bool error = true )  [inline]
```

Sets the `Error` snoop response bit.
The `Error` response bit indicates that the snooped cache line is in error.

Parameters

error	status of `Error` bit

See also

is_snoop_error()

7.43.3.21 is_snoop_was_unique()

```
bool amba_pv::amba_pv_response::is_snoop_was_unique ( ) const  [inline]
```

Returns wether or not the `WasUnique` snoop response bit is set.
The `WasUnique` bit indicates that the cache line was held in a Unique state before the snoop.

Returns

`true` if the `WasUnique` bit is set, `false` otherwise.

See also

set_snoop_was_unique()

7.43.3.22 set_snoop_was_unique()

```
void amba_pv::amba_pv_response::set_snoop_was_unique (
            bool was_unique = true )  [inline]
```

Sets the `WasUnique` snoop response bit.
The `WasUnique` bit indicates that the cache line was held in a Unique state before the snoop.

Parameters

was_unique	status of `WasUnique` bit

See also

is_snoop_was_unique()

7.43.3.23 reset()

```
void amba_pv::amba_pv_response::reset ( )  [inline]
```

Resets all members of this AMBA-PV extension to their default value.

7.44 amba_pv::amba_pv_simple_memory< BUSWIDTH > Class Template Reference

AMBA-PV simple memory model.

`#include <models/amba_pv_simple_memory.h>`

Inherits amba_pv::amba_pv_memory_base< 64 >, and sc_core::sc_module.

Public Member Functions

- amba_pv_simple_memory (const sc_core::sc_module_name &, const sc_dt::uint64 &)

  Constructor.
- amba_pv_simple_memory (const sc_core::sc_module_name &, const sc_dt::uint64 &, unsigned char)

  Constructor.
- amba_pv_simple_memory (const sc_core::sc_module_name &, const sc_dt::uint64 &, unsigned char, unsigned char)

  Constructor.
- ~amba_pv_simple_memory ()

  Destructor.
- virtual const char * kind () const

  Returns the kind string of this memory.

Data Fields

- amba_pv_slave_socket< BUSWIDTH > amba_pv_s

  Slave socket.

Protected Member Functions

- virtual amba_pv_resp_t read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes a read transaction.
- virtual amba_pv_resp_t write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, unsigned char *, sc_core::sc_time &)

  Completes a write transaction.
- virtual bool get_direct_mem_ptr (int, tlm::tlm_command, const sc_dt::uint64 &, const amba_pv_control *, tlm::tlm_dmi &)

  Requests DMI access to the specified address and returns a reference to a DMI descriptor.
- virtual unsigned int debug_read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug read transaction.
- virtual unsigned int debug_write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug write transaction.

- virtual amba_pv_resp_t atomic_store (int, const sc_dt::uint64 &, unsigned char *const, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic store transaction.
- virtual amba_pv_resp_t atomic_load (int, const sc_dt::uint64 &, unsigned char *const, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic load transaction.
- virtual amba_pv_resp_t atomic_compare (int, const sc_dt::uint64 &, unsigned char *const, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic compare.
- virtual amba_pv_resp_t atomic_swap (int, const sc_dt::uint64 &, unsigned char *const, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic swap.

7.44.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_simple_memory< BUSWIDTH >

AMBA-PV simple memory model.
The amba_pv_simple_memory class models an AMBA compatible memory at the PV level.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.44.2 Constructor & Destructor Documentation

7.44.2.1 amba_pv_simple_memory() [1/3]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_simple_memory< BUSWIDTH >::amba_pv_simple_memory (
            const sc_core::sc_module_name & name,
            const sc_dt::uint64 & size )  [inline]
```

Constructor.

Parameters

name	memory name.
size	memory size in bytes. *size* is rounded up to the next multiple of 4096.

7.44.2.2 amba_pv_simple_memory() [2/3]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_simple_memory< BUSWIDTH >::amba_pv_simple_memory (
            const sc_core::sc_module_name & name,
            const sc_dt::uint64 & size,
            unsigned char fill_char )  [inline]
```

Constructor.

Parameters

name	memory name.
size	memory size in bytes. *size* is rounded up to the next multiple of 4096.
fill_char	fill character used for uninitialized the memory.

7.44.2.3 amba_pv_simple_memory() [3/3]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_simple_memory< BUSWIDTH >::amba_pv_simple_memory (
            const sc_core::sc_module_name & name,
            const sc_dt::uint64 & size,
            unsigned char fill_char1,
            unsigned char fill_char2 )  [inline]
```

Constructor.

Parameters

name	memory name.
size	memory size in bytes. *size* is rounded up to the next multiple of 4096.
fill_char1	fill character used for uninitialized memory.
fill_char2	fill character used for uninitialized memory.

7.44.2.4 ~amba_pv_simple_memory()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_simple_memory< BUSWIDTH >::~amba_pv_simple_memory  [inline]
```

Destructor.

7.44.3 Member Function Documentation

7.44.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_simple_memory< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string of this memory.

7.44.3.2 read()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_simple_memory< BUSWIDTH >::read (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes a read transaction.

Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.44.3.3 write()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_simple_memory< BUSWIDTH >::write (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int size,
            const amba_pv_control * ctrl,
            unsigned char * strb,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes a write transaction.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.44.3.4 get_direct_mem_ptr()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_simple_memory< BUSWIDTH >::get_direct_mem_ptr (
            int socket_id,
            tlm::tlm_command command,
            const sc_dt::uint64 & addr,
            const amba_pv_control * ctrl,
            tlm::tlm_dmi & dmi_data )  [inline], [protected], [virtual]
```

Requests DMI access to the specified address and returns a reference to a DMI descriptor.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.44.3.5 debug_read()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_simple_memory< BUSWIDTH >::debug_read (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            const amba_pv_control * ctrl )  [inline], [protected], [virtual]
```

Non-intrusive debug read transaction.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.44.3.6 debug_write()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_simple_memory< BUSWIDTH >::debug_write (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            const amba_pv_control * ctrl )  [inline], [protected], [virtual]
```

Non-intrusive debug write transaction.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.44.3.7 atomic_store()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_simple_memory< BUSWIDTH >::atomic_store (
            int ,
```

```
            const sc_dt::uint64 & addr,
            unsigned char * const data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time &  )  [inline], [protected], [virtual]
```

Completes an atomic store transaction.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.44.3.8 atomic_load()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_simple_memory< BUSWIDTH >::atomic_load (
            int ,
            const sc_dt::uint64 & addr,
            unsigned char * const data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time &  )  [inline], [protected], [virtual]
```

Completes an atomic load transaction.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.44.3.9 atomic_compare()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_simple_memory< BUSWIDTH >::atomic_compare (
            int ,
            const sc_dt::uint64 & addr,
            unsigned char * const data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes an atomic compare.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.44.3.10 atomic_swap()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_simple_memory< BUSWIDTH >::atomic_swap (
            int ,
            const sc_dt::uint64 & addr,
            unsigned char * const data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ,
            sc_core::sc_time &  )  [inline], [protected], [virtual]
```

Completes an atomic swap.
Reimplemented from amba_pv::amba_pv_slave_base< BUSWIDTH >.

7.44.4 Field Documentation

7.44.4.1 amba_pv_s

```
template<unsigned int BUSWIDTH = 64>
amba_pv_slave_socket<BUSWIDTH> amba_pv::amba_pv_simple_memory< BUSWIDTH >::amba_pv_s
```

Slave socket.

7.45 amba_pv::amba_pv_simple_probe< BUSWIDTH > Class Template Reference

AMBA-PV simple probe model.

`#include <models/amba_pv_simple_probe.h>`

Inherits amba_pv::amba_pv_fw_transport_if, amba_pv::amba_pv_bw_transport_if, and amba_pv::amba_pv_simple_probe_base< 64

Public Member Functions

- amba_pv_simple_probe (const sc_core::sc_module_name &, bool=true)
  Constructor.
- virtual ~amba_pv_simple_probe ()
  Destructor.
- virtual const char * kind () const
  Returns the kind string of this probe.

Data Fields

- amba_pv_slave_socket< BUSWIDTH > amba_pv_s
  Slave socket.
- amba_pv_master_socket< BUSWIDTH > amba_pv_m
  Master socket.

Protected Member Functions

- virtual void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)
  Blocking transport.
- virtual unsigned int transport_dbg (int, amba_pv_transaction &)
  Debug access to a target.
- virtual bool get_direct_mem_ptr (int, amba_pv_transaction &, tlm::tlm_dmi &)
  Requests a DMI access based on the specified transaction.
- virtual void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)
  Invalidates DMI pointers previously established for the specified DMI region.

7.45.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_simple_probe< BUSWIDTH >

AMBA-PV simple probe model.

The amba_pv_simple_probe model prints the contents of transaction between a master and a slave to `std::cout`, a file, or a stream.

Note

If configured for printing transactions, the amba_pv_simple_probe model might have an effect on performance.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.45.2 Constructor & Destructor Documentation

7.45.2.1 amba_pv_simple_probe()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_simple_probe< BUSWIDTH >::amba_pv_simple_probe (
            const sc_core::sc_module_name & name,
            bool trans_verbose = true )  [inline], [explicit]
```

Constructor.

Parameters

name	probe name.
trans_verbose	`true` to print transactions (default), `false` otherwise.

See also

set_trans_verbose()

7.45.2.2 ~amba_pv_simple_probe()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_simple_probe< BUSWIDTH >::~amba_pv_simple_probe  [inline], [virtual]
```

Destructor.

7.45.3 Member Function Documentation

7.45.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_simple_probe< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string of this probe.

Reimplemented from amba_pv::amba_pv_simple_probe_base< 64 >.

7.45.3.2 b_transport()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_simple_probe< BUSWIDTH >::b_transport (
            int socket_id,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Blocking transport.

This version of the method completes the transaction and prints its contents.

Implements amba_pv::amba_pv_fw_transport_if.

7.45.3.3 transport_dbg()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_simple_probe< BUSWIDTH >::transport_dbg (
            int socket_id,
            amba_pv_transaction & trans )  [inline], [protected], [virtual]
```

Debug access to a target.
This version of the method forwards this debug access to the slave and prints its contents.
Implements amba_pv::amba_pv_fw_transport_if.

7.45.3.4 get_direct_mem_ptr()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_simple_probe< BUSWIDTH >::get_direct_mem_ptr (
            int socket_id,
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [inline], [protected], [virtual]
```

Requests a DMI access based on the specified transaction.
This version of the method forwards this DMI access request to the slave and prints its contents.
Implements amba_pv::amba_pv_fw_transport_if.

7.45.3.5 invalidate_direct_mem_ptr()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_simple_probe< BUSWIDTH >::invalidate_direct_mem_ptr (
            int socket_id,
            sc_dt::uint64 start_range,
            sc_dt::uint64 end_range )  [inline], [protected], [virtual]
```

Invalidates DMI pointers previously established for the specified DMI region.
This version of the method forwards this DMI call to the master after printing its arguments.
Implements amba_pv::amba_pv_bw_transport_if.

7.45.4 Field Documentation

7.45.4.1 amba_pv_s

```
template<unsigned int BUSWIDTH = 64>
amba_pv_slave_socket<BUSWIDTH> amba_pv::amba_pv_simple_probe< BUSWIDTH >::amba_pv_s
```

Slave socket.

7.45.4.2 amba_pv_m

```
template<unsigned int BUSWIDTH = 64>
amba_pv_master_socket<BUSWIDTH> amba_pv::amba_pv_simple_probe< BUSWIDTH >::amba_pv_m
```

Master socket.

7.46 amba_pv::amba_pv_simple_probe_base< BUSWIDTH > Class Template Reference

AMBA-PV simple probe base model.
`#include <models/amba_pv_simple_probe_base.h>`
Inherits sc_core::sc_module.

Public Member Functions

- amba_pv_simple_probe_base (const sc_core::sc_module_name &, bool=true)

  Constructor.
- virtual ~amba_pv_simple_probe_base ()

  Destructor.
- virtual const char * kind () const

  Returns the kind string of this probe.
- void set_trans_verbose (bool=true)

  Sets verbosity of this probe.
- void set_transport_verbose (bool=true)

  Sets verbosity of this probe with regard to `b_transport()` regular transactions.
- bool is_transport_verbose () const

  Gets verbosity of this probe with regard to `b_transport()` regular transactions.
- void set_debug_verbose (bool=true)

  Sets verbosity of this probe with regard to `transport_dbg()` debug transactions.
- bool is_debug_verbose () const

  Gets verbosity of this probe with regard to `transport_dbg()` debug transactions.
- void set_dmi_verbose (bool=true)

  Sets verbosity of this probe with regard to DMI transactions.
- bool is_dmi_verbose () const

  Gets verbosity of this probe with regard to DMI transactions.
- void set_data_verbose (bool=true)

  Sets verbosity of this probe with regard to transactions data.
- bool is_data_verbose () const

  Gets verbosity of this probe with regard to transactions data.
- void set_start_time (const sc_core::sc_time &)

  Sets start time of this probe.
- void set_stop_time (const sc_core::sc_time &)

  Sets stop time of this probe.

7.46.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_simple_probe_base< BUSWIDTH >

AMBA-PV simple probe base model.
The amba_pv_simple_probe_base model prints the contents of transaction between a master and a slave to `std::cout`, a file, or a stream.

Note

If configured for printing transactions, the amba_pv_simple_probe_base model might have an effect on performance.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 amba_pv_simple_probe_base()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::amba_pv_simple_probe_base (
            const sc_core::sc_module_name & name,
            bool trans_verbose = true )  [inline], [explicit]
```

Constructor.

Parameters

name	probe name.
trans_verbose	`true` to print transactions (default), `false` otherwise.

See also

set_trans_verbose()

7.46.2.2 ~amba_pv_simple_probe_base()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::~amba_pv_simple_probe_base  [inline], [virtual]
```

Destructor.

7.46.3 Member Function Documentation

7.46.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string of this probe.

Reimplemented in amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >, and amba_pv::amba_pv_simple_probe< BUSWIDTH >

7.46.3.2 set_trans_verbose()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::set_trans_verbose (
            bool verbose = true )  [inline]
```

Sets verbosity of this probe.

If verbosity is set to `false`, no transactions are printed.

Parameters

verbose	`true` to print transactions (default), `false` otherwise.

See also

set_transport_verbose(), set_debug_verbose(), set_dmi_verbose()

7.46.3.3 set_transport_verbose()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::set_transport_verbose (
            bool verbose = true )  [inline]
```

Sets verbosity of this probe with regard to `b_transport()` regular transactions.
If verbosity is set to `false`, no regular transactions are printed.

Parameters

verbose	`true` to print `b_transport()` regular transactions (default), `false` otherwise.

See also

is_transport_verbose()

7.46.3.4 is_transport_verbose()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::is_transport_verbose  [inline]
```

Gets verbosity of this probe with regard to `b_transport()` regular transactions.

Returns

Returns `true` if regular transactions are printed (default), `false` otherwise.

See also

set_transport_verbose()

7.46.3.5 set_debug_verbose()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::set_debug_verbose (
            bool verbose = true )  [inline]
```

Sets verbosity of this probe with regard to `transport_dbg()` debug transactions.
If verbosity is set to `false`, no debug transactions are printed.

Parameters

verbose	`true` to print debug transactions (default), `false` otherwise.

See also

is_debug_verbose()

7.46.3.6 is_debug_verbose()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::is_debug_verbose  [inline]
```

Gets verbosity of this probe with regard to `transport_dbg()` debug transactions.

Returns

Returns `true` if debug transactions are printed (default), `false` otherwise.

See also

set_debug_verbose()

7.46.3.7 set_dmi_verbose()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::set_dmi_verbose (
            bool verbose = true )  [inline]
```

Sets verbosity of this probe with regard to DMI transactions.
If verbosity is set to `false`, no DMI transactions are printed.

Parameters

verbose	`true` to print DMI transactions (default), `false` otherwise.

See also

is_dmi_verbose()

7.46.3.8 is_dmi_verbose()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::is_dmi_verbose  [inline]
```

Gets verbosity of this probe with regard to DMI transactions.

Returns

Returns `true` if DMI transactions are printed (default), `false` otherwise.

See also

set_dmi_verbose()

7.46.3.9 set_data_verbose()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::set_data_verbose (
            bool verbose = true )  [inline]
```

Sets verbosity of this probe with regard to transactions data.
If verbosity is set to `false`, the data pointer is printed instead.

Parameters

verbose	`true` to print transactions data (default), `false` otherwise.

See also

is_data_verbose()

7.46.3.10 is_data_verbose()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::is_data_verbose  [inline]
```

Gets verbosity of this probe with regard to transactions data.

Returns

Returns `true` if transactions data are printed, `false` otherwise.

See also

set_data_verbose()

7.46.3.11 set_start_time()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::set_start_time (
            const sc_core::sc_time & start_time )  [inline]
```

Sets start time of this probe.

Parameters

start_time	simulation time at which to start printing transactions.

See also

set_stop_time()

7.46.3.12 set_stop_time()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_simple_probe_base< BUSWIDTH >::set_stop_time (
            const sc_core::sc_time & stop_time )  [inline]
```

Sets stop time of this probe.

Parameters

stop_time	simulation time at which to stop printing transactions.

See also

set_start_time()

7.47 amba_pv::amba_pv_slave_base< BUSWIDTH > Class Template Reference

Base class for all AMBA-PV slave modules.

`#include <user/amba_pv_slave_base.h>`

Inherits amba_pv::amba_pv_fw_transport_if, and amba_pv::amba_pv_if< 64 >.

Public Member Functions

- amba_pv_slave_base (const std::string &)
  Constructor.
- amba_pv_slave_base (const std::string &, const sc_core::sc_time &, const sc_core::sc_time &)
  Constructor.
- std::string get_name () const
  Returns the name of this slave.
- sc_core::sc_time get_read_latency () const
  Returns the read latency of this slave.
- void set_read_latency (const sc_core::sc_time &)
  Sets the read latency of this slave.

- sc_core::sc_time get_write_latency () const

  Returns the write latency of this slave.
- void set_write_latency (const sc_core::sc_time &)

  Sets the write latency of this slave.

Protected Member Functions

- virtual void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking transport.
- virtual unsigned int transport_dbg (int, amba_pv_transaction &)

  Debug access to a target.
- virtual bool get_direct_mem_ptr (int, amba_pv_transaction &, tlm::tlm_dmi &)

  Requests a DMI access based on the specified transaction.
- virtual amba_pv_resp_t read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes a read transaction.
- virtual amba_pv_resp_t write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, unsigned char *, sc_core::sc_time &)

  Completes a write transaction.
- virtual amba_pv_resp_t burst_read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_burst_t, sc_core::sc_time &)

  Completes a burst read transaction.
- virtual amba_pv_resp_t burst_write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_burst_t, unsigned char *, unsigned int, sc_core::sc_time &)

  Completes a burst write transaction.
- virtual bool get_direct_mem_ptr (int, tlm::tlm_command, const sc_dt::uint64 &, const amba_pv_control *, tlm::tlm_dmi &)

  Requests DMI access to the specified address and returns a reference to a DMI descriptor.
- virtual unsigned int debug_read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug read transaction.
- virtual unsigned int debug_write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug write transaction.
- virtual amba_pv_resp_t atomic_store (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic store transaction.
- virtual amba_pv_resp_t atomic_load (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic load transaction.
- virtual amba_pv_resp_t atomic_swap (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic swap transaction.
- virtual amba_pv_resp_t atomic_compare (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic compare transaction.

7.47.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_slave_base< BUSWIDTH >

Base class for all AMBA-PV slave modules.
amba_pv_slave_base is intended to be bound to amba_pv_slave_socket.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

Note

amba_pv_slave_base is not an `sc_module`.

7.47.2 Constructor & Destructor Documentation

7.47.2.1 amba_pv_slave_base() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_slave_base< BUSWIDTH >::amba_pv_slave_base (
            const std::string & name )  [inline], [explicit]
```

Constructor.

Parameters

name	slave name.

7.47.2.2 amba_pv_slave_base() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_slave_base< BUSWIDTH >::amba_pv_slave_base (
            const std::string & name,
            const sc_core::sc_time & read_latency,
            const sc_core::sc_time & write_latency )  [inline]
```

Constructor.

Parameters

name	slave name
read_latency	average read latency per byte
write_latency	average write latency per byte.

7.47.3 Member Function Documentation

7.47.3.1 get_name()

```
template<unsigned int BUSWIDTH>
std::string amba_pv::amba_pv_slave_base< BUSWIDTH >::get_name  [inline]
```

Returns the name of this slave.

7.47.3.2 get_read_latency()

```
template<unsigned int BUSWIDTH>
sc_core::sc_time amba_pv::amba_pv_slave_base< BUSWIDTH >::get_read_latency  [inline]
```

Returns the read latency of this slave.
This function returns the average read latency per byte.

7.47.3.3 set_read_latency()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_slave_base< BUSWIDTH >::set_read_latency (
            const sc_core::sc_time & t )  [inline]
```

Sets the read latency of this slave.

Parameters

t	average read latency per byte.

7.47.3.4 get_write_latency()

```
template<unsigned int BUSWIDTH>
sc_core::sc_time amba_pv::amba_pv_slave_base< BUSWIDTH >::get_write_latency  [inline]
```

Returns the write latency of this slave.
This function returns the average write latency per byte.

7.47.3.5 set_write_latency()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_slave_base< BUSWIDTH >::set_write_latency (
            const sc_core::sc_time & t )  [inline]
```

Sets the write latency of this slave.

Parameters

t	average write latency per byte.

7.47.3.6 b_transport()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_slave_base< BUSWIDTH >::b_transport (
            int socket_id,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Blocking transport.
This version of the method translates the blocking transport call into amba_pv_if user-layer calls. In addition, the following rules are cheked:

- The data length attribute must be greater than or equal to the burst size times the burst length. If not, an error response of `tlm::TLM_BURST_ERROR_REPONSE` is returned.
- The streaming width attribute must be equal to the burst size for a fixed burst. If not, an error response of `tlm::TLM_BURST_ERROR_REPONSE` is returned.
- The byte enable pointer attribute must be `NULL` on read transactions. If not, an error response of `tlm::TLM_BYTE_ENABLE_ERROR_REPONSE` is returned.
- The byte enable length attribute must be a multiple of the burst size on write transactions. If not, an error response of `tlm::TLM_BYTE_ENABLE_ERROR_REPONSE` is returned.

Implements amba_pv::amba_pv_fw_transport_if.
Reimplemented in amba_pv::amba_pv_memory_base< BUSWIDTH >, and amba_pv::amba_pv_memory_base< 64 >.

7.47.3.7 transport_dbg()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_slave_base< BUSWIDTH >::transport_dbg (
            int socket_id,
            amba_pv_transaction & trans )  [inline], [protected], [virtual]
```

Debug access to a target.
This version of the method translates the transport_dbg() call into amba_pv_if user-layer calls.
Implements amba_pv::amba_pv_fw_transport_if.

7.47.3.8 get_direct_mem_ptr() [1/2]

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_slave_base< BUSWIDTH >::get_direct_mem_ptr (
            int socket_id,
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [inline], [protected], [virtual]
```

Requests a DMI access based on the specified transaction.
This version of the method translates the DMI access request call into amba_pv_if user-layer calls.
Implements amba_pv::amba_pv_fw_transport_if.

7.47.3.9 read()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_slave_base< BUSWIDTH >::read (
            int ,
            const sc_dt::uint64 & ,
            unsigned char * ,
            unsigned int ,
            const amba_pv_control * ,
            sc_core::sc_time &  )  [inline], [protected], [virtual]
```

Completes a read transaction.
This version of the method causes an error.
Implements amba_pv::amba_pv_if< 64 >.
Reimplemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, and amba_pv::amba_pv_simple_memory< BUSWI

7.47.3.10 write()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_slave_base< BUSWIDTH >::write (
            int ,
            const sc_dt::uint64 & ,
            unsigned char * ,
            unsigned int ,
            const amba_pv_control * ,
            unsigned char * ,
            sc_core::sc_time &  )  [inline], [protected], [virtual]
```

Completes a write transaction.
This version of the method causes an error.
Implements amba_pv::amba_pv_if< 64 >.
Reimplemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, and amba_pv::amba_pv_simple_memory< BUSWI

7.47.3.11 burst_read()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_slave_base< BUSWIDTH >::burst_read (
```

```
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes a burst read transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.47.3.12 burst_write()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_slave_base< BUSWIDTH >::burst_write (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst,
            unsigned char * strb,
            unsigned int strb_length,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes a burst write transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.47.3.13 get_direct_mem_ptr() [2/2]

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_slave_base< BUSWIDTH >::get_direct_mem_ptr (
            int ,
            tlm::tlm_command ,
            const sc_dt::uint64 & ,
            const amba_pv_control * ,
            tlm::tlm_dmi & dmi_data )  [inline], [protected], [virtual]
```

Requests DMI access to the specified address and returns a reference to a DMI descriptor.
This version of the method returns `false` and denies DMI access to the entire memory region.
Implements amba_pv::amba_pv_if< 64 >.
Reimplemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, and amba_pv::amba_pv_simple_memory< BUSWI

7.47.3.14 debug_read()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_slave_base< BUSWIDTH >::debug_read (
            int ,
            const sc_dt::uint64 & ,
            unsigned char * ,
            unsigned int ,
            const amba_pv_control *  )  [inline], [protected], [virtual]
```

Non-intrusive debug read transaction.
This version of the method returns `0`.
Implements amba_pv::amba_pv_if< 64 >.
Reimplemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, and amba_pv::amba_pv_simple_memory< BUSWI

7.47.3.15 debug_write()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_slave_base< BUSWIDTH >::debug_write (
            int ,
            const sc_dt::uint64 & ,
            unsigned char * ,
            unsigned int ,
            const amba_pv_control *  )  [inline], [protected], [virtual]
```

Non-intrusive debug write transaction.
This version of the method returns 0.
Implements amba_pv::amba_pv_if< 64 >.
Reimplemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, and amba_pv::amba_pv_simple_memory< BUSWI

7.47.3.16 atomic_store()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_slave_base< BUSWIDTH >::atomic_store (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes an atomic store transaction.
This version of the method causes an error.
Implements amba_pv::amba_pv_if< 64 >.
Reimplemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, and amba_pv::amba_pv_simple_memory< BUSWI

7.47.3.17 atomic_load()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_slave_base< BUSWIDTH >::atomic_load (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes an atomic load transaction.
This version of the method causes an error.
Implements amba_pv::amba_pv_if< 64 >.
Reimplemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, and amba_pv::amba_pv_simple_memory< BUSWI

7.47.3.18 atomic_swap()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_slave_base< BUSWIDTH >::atomic_swap (
            int socket_id,
            const sc_dt::uint64 & addr,
```

```
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes an atomic swap transaction.
This version of the method causes an error.
Implements amba_pv::amba_pv_if< 64 >.
Reimplemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, and amba_pv::amba_pv_simple_memory< BUSWI

7.47.3.19 atomic_compare()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::amba_pv_slave_base< BUSWIDTH >::atomic_compare (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes an atomic compare transaction.
This version of the method causes an error.
Implements amba_pv::amba_pv_if< 64 >.
Reimplemented in amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >, and amba_pv::amba_pv_simple_memory< BUSWI

7.48 amba_pv::ext::amba_pv_slave_base< BUSWIDTH > Class Template Reference

Base class for all AMBA-PV slave modules.
`#include <user/amba_pv_ext_slave_base.h>`
Inherits amba_pv::ext::amba_pv_fw_transport_if, and amba_pv::amba_pv_if< 64 >.

Public Member Functions

- amba_pv_slave_base (const std::string &)
  Constructor.
- amba_pv_slave_base (const std::string &, const sc_core::sc_time &, const sc_core::sc_time &)
  Constructor.
- std::string get_name () const
  Returns the name of this slave.
- sc_core::sc_time get_read_latency () const
  Returns the read latency of this slave.
- void set_read_latency (const sc_core::sc_time &)
  Sets the read latency of this slave.
- sc_core::sc_time get_write_latency () const
  Returns the write latency of this slave.
- void set_write_latency (const sc_core::sc_time &)
  Sets the write latency of this slave.

Protected Member Functions

- virtual void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)
  Blocking transport.

- virtual unsigned int transport_dbg (int, amba_pv_transaction &)

  Debug access to a target.
- virtual bool get_direct_mem_ptr (int, amba_pv_transaction &, tlm::tlm_dmi &)

  Requests a DMI access based on the specified transaction.
- virtual amba_pv_resp_t read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes a read transaction.
- virtual amba_pv_resp_t write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *, unsigned char *, sc_core::sc_time &)

  Completes a write transaction.
- virtual amba_pv_resp_t burst_read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_burst_t, sc_core::sc_time &)

  Completes a burst read transaction.
- virtual amba_pv_resp_t burst_write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_burst_t, unsigned char *, unsigned int, sc_core::sc_time &)

  Completes a burst write transaction.
- virtual bool get_direct_mem_ptr (int, tlm::tlm_command, const sc_dt::uint64 &, const amba_pv_control *, tlm::tlm_dmi &)

  Requests DMI access to the specified address and returns a reference to a DMI descriptor.
- virtual unsigned int debug_read (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug read transaction.
- virtual unsigned int debug_write (int, const sc_dt::uint64 &, unsigned char *, unsigned int, const amba_pv_control *)

  Non-intrusive debug write transaction.
- virtual amba_pv_resp_t atomic_store (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic store transaction.
- virtual amba_pv_resp_t atomic_load (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, amba_pv_atomic_subop_t, amba_pv_atomic_endianness_t, sc_core::sc_time &)

  Completes an atomic load transaction.
- virtual amba_pv_resp_t atomic_swap (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic swap transaction.
- virtual amba_pv_resp_t atomic_compare (int, const sc_dt::uint64 &, unsigned char *, unsigned int, unsigned int, const amba_pv_control *, sc_core::sc_time &)

  Completes an atomic compare transaction.

7.48.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::ext::amba_pv_slave_base< BUSWIDTH >

Base class for all AMBA-PV slave modules.
amba_pv_slave_base is intended to be bound to amba_pv_slave_socket.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

Note

amba_pv_slave_base is not an `sc_module`.

7.48.2 Constructor & Destructor Documentation

7.48.2.1 amba_pv_slave_base() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::amba_pv_slave_base (
            const std::string & name )  [inline], [explicit]
```

Constructor.

Parameters

name	slave name.

7.48.2.2 amba_pv_slave_base() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::amba_pv_slave_base (
            const std::string & name,
            const sc_core::sc_time & read_latency,
            const sc_core::sc_time & write_latency )  [inline]
```

Constructor.

Parameters

name	slave name
read_latency	average read latency per byte
write_latency	average write latency per byte.

7.48.3 Member Function Documentation

7.48.3.1 get_name()

```
template<unsigned int BUSWIDTH>
std::string amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::get_name  [inline]
```

Returns the name of this slave.

7.48.3.2 get_read_latency()

```
template<unsigned int BUSWIDTH>
sc_core::sc_time amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::get_read_latency  [inline]
```

Returns the read latency of this slave.
This function returns the average read latency per byte.

7.48.3.3 set_read_latency()

```
template<unsigned int BUSWIDTH>
void amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::set_read_latency (
            const sc_core::sc_time & t )  [inline]
```

Sets the read latency of this slave.

Parameters

t	average read latency per byte.

7.48.3.4 get_write_latency()

```
template<unsigned int BUSWIDTH>
sc_core::sc_time amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::get_write_latency  [inline]
```

Returns the write latency of this slave.
This function returns the average write latency per byte.

7.48.3.5 set_write_latency()

```
template<unsigned int BUSWIDTH>
void amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::set_write_latency (
            const sc_core::sc_time & t )  [inline]
```

Sets the write latency of this slave.

Parameters

t	average write latency per byte.

7.48.3.6 b_transport()

```
template<unsigned int BUSWIDTH>
void amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::b_transport (
            int socket_id,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Blocking transport.
This version of the method translates the blocking transport call into amba_pv_if user-layer calls. In addition, the following rules are cheked:

- The data length attribute must be greater than or equal to the burst size times the burst length. If not, an error response of `tlm::TLM_BURST_ERROR_REPONSE` is returned.
- The streaming width attribute must be equal to the burst size for a fixed burst. If not, an error response of `tlm::TLM_BURST_ERROR_REPONSE` is returned.
- The byte enable pointer attribute must be `NULL` on read transactions. If not, an error response of `tlm::TLM_BYTE_ENABLE_ERROR_REPONSE` is returned.
- The byte enable length attribute must be a multiple of the burst size on write transactions. If not, an error response of `tlm::TLM_BYTE_ENABLE_ERROR_REPONSE` is returned.

Implements amba_pv::ext::amba_pv_fw_transport_if.

7.48.3.7 transport_dbg()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::transport_dbg (
            int socket_id,
            amba_pv_transaction & trans )  [inline], [protected], [virtual]
```

Debug access to a target.
This version of the method translates the transport_dbg() call into amba_pv_if user-layer calls.
Implements amba_pv::ext::amba_pv_fw_transport_if.

7.48.3.8 get_direct_mem_ptr() [1/2]

```
template<unsigned int BUSWIDTH>
bool amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::get_direct_mem_ptr (
            int socket_id,
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [inline], [protected], [virtual]
```

Requests a DMI access based on the specified transaction.
This version of the method translates the DMI access request call into amba_pv_if user-layer calls.
Implements amba_pv::ext::amba_pv_fw_transport_if.

7.48.3.9 read()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::read (
            int ,
            const sc_dt::uint64 & ,
            unsigned char * ,
            unsigned int ,
            const amba_pv_control * ,
            sc_core::sc_time &  )  [inline], [protected], [virtual]
```

Completes a read transaction.
This version of the method causes an error.
Implements amba_pv::amba_pv_if< 64 >.

7.48.3.10 write()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::write (
            int ,
            const sc_dt::uint64 & ,
            unsigned char * ,
            unsigned int ,
            const amba_pv_control * ,
            unsigned char * ,
            sc_core::sc_time &  )  [inline], [protected], [virtual]
```

Completes a write transaction.
This version of the method causes an error.
Implements amba_pv::amba_pv_if< 64 >.

7.48.3.11 burst_read()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::burst_read (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
```

```
            amba_pv_burst_t burst,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes a burst read transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.48.3.12 burst_write()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::burst_write (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst,
            unsigned char * strb,
            unsigned int strb_length,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes a burst write transaction.
Implements amba_pv::amba_pv_if< 64 >.

7.48.3.13 get_direct_mem_ptr() [2/2]

```
template<unsigned int BUSWIDTH>
bool amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::get_direct_mem_ptr (
            int ,
            tlm::tlm_command ,
            const sc_dt::uint64 & ,
            const amba_pv_control * ,
            tlm::tlm_dmi & dmi_data )  [inline], [protected], [virtual]
```

Requests DMI access to the specified address and returns a reference to a DMI descriptor.
This version of the method returns `false` and denies DMI access to the entire memory region.
Implements amba_pv::amba_pv_if< 64 >.

7.48.3.14 debug_read()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::debug_read (
            int ,
            const sc_dt::uint64 & ,
            unsigned char * ,
            unsigned int ,
            const amba_pv_control *  )  [inline], [protected], [virtual]
```

Non-intrusive debug read transaction.
This version of the method returns `0`.
Implements amba_pv::amba_pv_if< 64 >.

7.48.3.15 debug_write()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::debug_write (
            int ,
            const sc_dt::uint64 & ,
            unsigned char * ,
```

```
            unsigned int ,
            const amba_pv_control *  )  [inline], [protected], [virtual]
```

Non-intrusive debug write transaction.
This version of the method returns 0.
Implements amba_pv::amba_pv_if< 64 >.

7.48.3.16 atomic_store()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::atomic_store (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes an atomic store transaction.
This version of the method causes an error.
Implements amba_pv::amba_pv_if< 64 >.

7.48.3.17 atomic_load()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::atomic_load (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_atomic_subop_t subop,
            amba_pv_atomic_endianness_t endianness,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes an atomic load transaction.
This version of the method causes an error.
Implements amba_pv::amba_pv_if< 64 >.

7.48.3.18 atomic_swap()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::atomic_swap (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes an atomic swap transaction.
This version of the method causes an error.
Implements amba_pv::amba_pv_if< 64 >.

7.48.3.19 atomic_compare()

```
template<unsigned int BUSWIDTH>
amba_pv_resp_t amba_pv::ext::amba_pv_slave_base< BUSWIDTH >::atomic_compare (
            int socket_id,
            const sc_dt::uint64 & addr,
            unsigned char * data,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Completes an atomic compare transaction.
This version of the method causes an error.
Implements amba_pv::amba_pv_if< 64 >.

7.49 amba_pv::amba_pv_slave_socket< BUSWIDTH > Class Template Reference

AMBA-PV socket to be instantiated on the slave side.
`#include <sockets/amba_pv_slave_socket.h>`
Inherits amba_pv::amba_pv_socket_base, and tlm_utils::simple_target_socket_tagged< amba_pv_fw_transport_if, 64, amba_pv_protocol_types >.

Public Member Functions

- amba_pv_slave_socket ()
  Default constructor.
- amba_pv_slave_socket (const char *, int=0)
  Constructor.
- virtual const char * kind () const
  Returns the kind string of this socket.
- void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)
  Invalidates DMI pointers previously established for the specified DMI region.
- void invalidate_direct_mem_ptr (sc_dt::uint64, sc_dt::uint64)
  Invalidates DMI pointers previously established for the specified DMI region.
- void bind (amba_pv_fw_transport_if &)
  Binds the specified interface to this socket.
- void operator() (amba_pv_fw_transport_if &)
  Binds the specified interface to this socket.

7.49.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_slave_socket< BUSWIDTH >

AMBA-PV socket to be instantiated on the slave side.
This socket inherits from the OSCI TLM 2.0 `tlm_utils::simple_target_socket_tagged` class. A tagged socket allows a component to determine through which socket an incoming method call arrived. This is required if there are multiple slave sockets such as in, for example, an interconnect or a multi-port memory.

Note

This version of the amba_pv_slave_socket class inherits from the OSCI TLM 2.0 `tlm_utils::simple_target_socket_tagged` class. Hence, if compiling applications that use amba_pv_slave_socket with OSCI SystemC, it is required to define the macro `SC_INCLUDE_DYNAMIC_PROCESSES` before including the OSCI SystemC header file.

This socket, as its base class `tlm_utils::simple_target_socket_tagged`, does not support hierarchical binding, master-socket-to-master-socket or slave-socket-to-slave-socket.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.49.2 Constructor & Destructor Documentation

7.49.2.1 amba_pv_slave_socket() [1/2]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_slave_socket< BUSWIDTH >::amba_pv_slave_socket  [inline]
```

Default constructor.

7.49.2.2 amba_pv_slave_socket() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_slave_socket< BUSWIDTH >::amba_pv_slave_socket (
            const char * name,
            int socket_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.
socket_id	socket identifier (defaults to 0).

7.49.3 Member Function Documentation

7.49.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_slave_socket< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string of this socket.

Reimplemented in amba_pv::amba_pv_ace_slave_socket< BUSWIDTH >, and amba_pv::amba_pv_ace_slave_socket< 64 >.

7.49.3.2 invalidate_direct_mem_ptr() [1/2]

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_slave_socket< BUSWIDTH >::invalidate_direct_mem_ptr (
            int socket_id,
```

```
            sc_dt::uint64 start_range,
            sc_dt::uint64 end_range )  [inline]
```

Invalidates DMI pointers previously established for the specified DMI region.

Parameters

socket_id	socket identifier (ignored on the slave side).
start_range	DMI region start address.
end_range	DMI region end address.

This version of the method forwards the invalidate_direct_mem_ptr() call to the master socket this slave socket is bound to.

7.49.3.3 invalidate_direct_mem_ptr() [2/2]

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_slave_socket< BUSWIDTH >::invalidate_direct_mem_ptr (
            sc_dt::uint64 start_range,
            sc_dt::uint64 end_range )  [inline]
```

Invalidates DMI pointers previously established for the specified DMI region.

Parameters

start_range	DMI region start address.
end_range	DMI region end address.

This version of the method forwards the invalidate_direct_mem_ptr() call to the master socket this slave socket is bound to.

7.49.3.4 bind()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_slave_socket< BUSWIDTH >::bind (
            amba_pv_fw_transport_if & iface )  [inline]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_fw_transport_if interface to bind to this socket.

7.49.3.5 operator()()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_slave_socket< BUSWIDTH >::operator() (
            amba_pv_fw_transport_if & iface )  [inline]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_fw_transport_if interface to bind to this socket.

7.50 amba_pv::ext::amba_pv_slave_socket< BUSWIDTH, N, POL > Class Template Reference

AMBA-PV socket to be instantiated on the slave side.
`#include <sockets/amba_pv_ext_slave_socket.h>`
Inherits amba_pv::ext::amba_pv_base_slave_socket< 64, 1, sc_core::SC_ONE_OR_MORE_BOUND >.

Public Member Functions

- amba_pv_slave_socket ()

  Default constructor.
- amba_pv_slave_socket (const char *, int=0)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this socket.
- void invalidate_direct_mem_ptr (int, sc_dt::uint64, sc_dt::uint64)

  Invalidates DMI pointers previously established for the specified DMI region.
- void invalidate_direct_mem_ptr (sc_dt::uint64, sc_dt::uint64)

  Invalidates DMI pointers previously established for the specified DMI region.

7.50.1 Detailed Description

template<unsigned int BUSWIDTH = 64, int N = 1, sc_core::sc_port_policy POL = sc_core::SC_ONE_OR_MORE_BOUND>
class amba_pv::ext::amba_pv_slave_socket< BUSWIDTH, N, POL >

AMBA-PV socket to be instantiated on the slave side.
This socket is for use as a slave socket bound to one or more master sockets.
To use this class, you must define the `AMBA_PV_INCLUDE_HIERARCHICAL_BINDING` macro at compile time.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.
N	number of bindings. Defaults to 1.
POL	port binding policy. Defaults to `sc_core::SC_ONE_OR_MORE_BOUND`.

7.50.2 Constructor & Destructor Documentation

7.50.2.1 amba_pv_slave_socket() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_slave_socket< BUSWIDTH, N, POL >::amba_pv_slave_socket  [inline]
```
Default constructor.

7.50.2.2 amba_pv_slave_socket() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
amba_pv::ext::amba_pv_slave_socket< BUSWIDTH, N, POL >::amba_pv_slave_socket (
            const char * name,
            int socket_id = 0 )  [inline], [explicit]
```
Constructor.

Parameters

name	socket name.
socket↩ _id	socket identifier (defaults to 0).

7.50.3 Member Function Documentation

7.50.3.1 kind()

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
const char * amba_pv::ext::amba_pv_slave_socket< BUSWIDTH, N, POL >::kind  [inline], [virtual]
```

Returns the kind string of this socket.

Reimplemented from amba_pv::ext::amba_pv_base_slave_socket< 64, 1, sc_core::SC_ONE_OR_MORE_BOUND >.

7.50.3.2 invalidate_direct_mem_ptr() [1/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_slave_socket< BUSWIDTH, N, POL >::invalidate_direct_mem_ptr (
            int index,
            sc_dt::uint64 start_range,
            sc_dt::uint64 end_range )  [inline]
```

Invalidates DMI pointers previously established for the specified DMI region.

Parameters

index	interface index (for sockets bound more than once).
start_range	DMI region start address.
end_range	DMI region end address.

This version of the method forwards the invalidate_direct_mem_ptr() call to the *index'ed* master socket this slave socket is bound to.

7.50.3.3 invalidate_direct_mem_ptr() [2/2]

```
template<unsigned int BUSWIDTH, int N, sc_core::sc_port_policy POL>
void amba_pv::ext::amba_pv_slave_socket< BUSWIDTH, N, POL >::invalidate_direct_mem_ptr (
            sc_dt::uint64 start_range,
            sc_dt::uint64 end_range )  [inline]
```

Invalidates DMI pointers previously established for the specified DMI region.

Parameters

start_range	DMI region start address.
end_range	DMI region end address.

This version of the method forwards the invalidate_direct_mem_ptr() call to the master socket this slave socket is bound to.

7.51 amba_pv::amba_pv_snoop_socket< BUSWIDTH > Class Template Reference

AMBA-PV slave socket used to implement the upstream ACE snoop interface.
`#include <sockets/amba_pv_snoop_socket.h>`
Inherits amba_pv::amba_pv_socket_base, and tlm_utils::simple_target_socket_tagged< amba_pv_bw_snoop_if, 64, amba_pv_protocol_types >.

Public Member Functions

- amba_pv_snoop_socket ()

  Default constructor.
- amba_pv_snoop_socket (const char *, int=0)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this socket.
- void bind (amba_pv_bw_snoop_if &iface)

  Binds the specified interface to this socket.
- void operator() (amba_pv_bw_snoop_if &iface)

  Binds the specified interface to this socket.

7.51.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_snoop_socket< BUSWIDTH >

AMBA-PV slave socket used to implement the upstream ACE snoop interface.
amba_pv_snoop_socket is the upstream ACE slave socket that is a private data member of each amba_pv_ace_master_socket.
amba_pv_snoop_socket provides implementations for the amba_pv_bw_snoop_if user-layer interface.
This socket inherits from the OSCI TLM 2.0 `tlm_utils::simple_target_socket_tagged` class. A tagged socket enables a component to determine through which socket an incoming method call arrived. This is required if there are multiple master sockets such as in, for example, a bus decoder.

Note

The current implementation of amba_pv_snoop_socket inherits from OSCI TLM 2.0 `tlm_utils::simple_target_socket_tagged`. Hence, if compiling applications that use amba_pv_snoop_socket with OSCI SystemC, it is required to define the macro `SC_INCLUDE_DYNAMIC_PROCESSES` before including the OSCI SystemC header file.

This socket, as its base class `tlm_utils::simple_target_socket_tagged`, does not support hierarchical binding, master-socket-to-master-socket or slave-socket-to-slave-socket

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.51.2 Constructor & Destructor Documentation

7.51.2.1 amba_pv_snoop_socket() [1/2]

`template<unsigned int BUSWIDTH>`
`amba_pv::amba_pv_snoop_socket< BUSWIDTH >::amba_pv_snoop_socket [inline]`
Default constructor.

7.51.2.2 amba_pv_snoop_socket() [2/2]

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_snoop_socket< BUSWIDTH >::amba_pv_snoop_socket (
            const char * name,
            int socket_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.
socket↩ _id	socket identifier (defaults to 0).

7.51.3 Member Function Documentation

7.51.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_snoop_socket< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string of this socket.

7.51.3.2 bind()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_snoop_socket< BUSWIDTH >::bind (
            amba_pv_bw_snoop_if & iface )  [inline]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_bw_snoop_if interface to bind to this socket.

7.51.3.3 operator()()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_snoop_socket< BUSWIDTH >::operator() (
            amba_pv_bw_snoop_if & iface )  [inline]
```

Binds the specified interface to this socket.

Parameters

iface	amba_pv_bw_snoop_if interface to bind to this socket.

7.52 amba_pv::amba_pv_socket_array< SOCKET > Class Template Reference

AMBA-PV socket array class.

```
#include <sockets/amba_pv_socket_array.h>
```

Public Member Functions

- amba_pv_socket_array (const char *, unsigned int)

  Constructor.
- ~amba_pv_socket_array ()

  Destructor.
- SOCKET & operator[ ] (unsigned int)

  Returns the socket at the specified index.
- const SOCKET & operator[ ] (unsigned int) const

  Returns the socket at the specified index.
- unsigned int size () const

  Returns the socket array size.

7.52.1 Detailed Description

template<class SOCKET>
class amba_pv::amba_pv_socket_array< SOCKET >

AMBA-PV socket array class.

Parameters

SOCKET	socket type.

7.52.2 Constructor & Destructor Documentation

7.52.2.1 amba_pv_socket_array()

```
template<class SOCKET >
amba_pv::amba_pv_socket_array< SOCKET >::amba_pv_socket_array (
            const char * name,
            unsigned int size )  [inline]
```

Constructor.

Parameters

name	socket array name that is used to generate the individual socket names of form "name%d", where d is the zero-based index of the socket.
size	socket array size.

7.52.2.2 ~amba_pv_socket_array()

```
template<class SOCKET >
amba_pv::amba_pv_socket_array< SOCKET >::~amba_pv_socket_array  [inline]
```

Destructor.

7.52.3 Member Function Documentation

7.52.3.1 operator[]() [1/2]

```
template<class SOCKET >
SOCKET & amba_pv::amba_pv_socket_array< SOCKET >::operator[] (
            unsigned int index )  [inline]
```

Returns the socket at the specified index.

Parameters

index	index of the accessed socket.

Returns

Socket reference.

7.52.3.2 operator[]() [2/2]

```
template<class SOCKET >
const SOCKET & amba_pv::amba_pv_socket_array< SOCKET >::operator[] (
            unsigned int index ) const  [inline]
```

Returns the socket at the specified index.

Parameters

index	index of the accessed socket.

Returns

Socket const reference.

7.52.3.3 size()

```
template<class SOCKET >
unsigned int amba_pv::amba_pv_socket_array< SOCKET >::size  [inline]
```

Returns the socket array size.

7.53 amba_pv::amba_pv_socket_base Class Reference

AMBA-PV socket base class.

`#include <sockets/amba_pv_socket_base.h>`

Inherited by amba_pv::amba_pv_master_socket< 64 >, amba_pv::amba_pv_master_socket< BUSWIDTH >, amba_pv::amba_pv_slave_socket< 64 >, amba_pv::amba_pv_slave_socket< BUSWIDTH >, amba_pv::amba_pv_snoop_socket< amba_pv::amba_pv_snoop_socket< BUSWIDTH >, amba_pv::ext::amba_pv_ace_base_master_socket< 64, 1, sc_core::SC_ONE_ amba_pv::ext::amba_pv_ace_base_slave_socket< 64, 1, sc_core::SC_ONE_OR_MORE_BOUND >, amba_pv::ext::amba_pv_base amba_pv::ext::amba_pv_base_slave_socket< 64, 1, sc_core::SC_ONE_OR_MORE_BOUND >, amba_pv::amba_pv_master_socke amba_pv::amba_pv_slave_socket< BUSWIDTH >, amba_pv::amba_pv_snoop_socket< BUSWIDTH >, amba_pv::ext::amba_pv_a amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >, amba_pv::ext::amba_pv_base_master_socket< BUSWI and amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >.

Public Member Functions

- amba_pv_socket_base (int=0)

  Constructor.
- int get_socket_id () const

Returns the identifier of this socket.

- void set_socket_id (int)

  Sets the identifier of this socket.

7.53.1 Detailed Description

AMBA-PV socket base class.

7.53.2 Constructor & Destructor Documentation

7.53.2.1 amba_pv_socket_base()

```
amba_pv::amba_pv_socket_base::amba_pv_socket_base (
            int socket_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

socket_id	socket identifier (defaults to 0).

7.53.3 Member Function Documentation

7.53.3.1 get_socket_id()

```
int amba_pv::amba_pv_socket_base::get_socket_id ( ) const  [inline]
```

Returns the identifier of this socket.

7.53.3.2 set_socket_id()

```
void amba_pv::amba_pv_socket_base::set_socket_id (
            int socket_id )  [inline]
```

Sets the identifier of this socket.

Parameters

socket_id	socket identifier.

7.54 amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH > Class Template Reference

AMBA-PV to TLM 2.0 BP bridge module.

`#include <models/amba_pv_bridges.h>`

Inherits amba_pv::amba_pv_fw_transport_if, and sc_core::sc_module.

Public Member Functions

- amba_pv_to_tlm_bridge (const sc_core::sc_module_name &)

  Constructor.

- virtual const char * kind () const

  Returns the kind string of this bridge.
- bool is_overwrite_exok_with_ok () const

  Returns whether the bridge is configured to replace AMBA_PV_EXOKAY with AMBA_PV_OKAY.
- void set_overwrite_exok_with_ok (bool)

  Sets whether the bridge is configured to replace AMBA_PV_EXOKAY with AMBA_PV_OKAY.

Data Fields

- amba_pv_slave_socket< BUSWIDTH > amba_pv_s

  Slave socket from AMBA-PV.
- tlm_utils::simple_initiator_socket< amba_pv_to_tlm_bridge, BUSWIDTH, tlm::tlm_base_protocol_types > tlm_m

  Master socket to TLM 2.0 BP.

Protected Member Functions

- virtual void b_transport (int, amba_pv_transaction &, sc_core::sc_time &)

  Blocking transport.
- virtual bool get_direct_mem_ptr (int, amba_pv_transaction &trans, tlm::tlm_dmi &)

  Requests a DMI access based on the specified transaction.
- virtual unsigned int transport_dbg (int, amba_pv_transaction &trans)

  Debug access to a target.

7.54.1 Detailed Description

template<unsigned int BUSWIDTH = 64>
class amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >

AMBA-PV to TLM 2.0 BP bridge module.
The amba_pv_to_tlm_bridge class converts AMBA-PV transactions into TLM 2.0 BP transactions.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024. Defaults to 64.

7.54.2 Constructor & Destructor Documentation

7.54.2.1 amba_pv_to_tlm_bridge()

```
template<unsigned int BUSWIDTH>
amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >::amba_pv_to_tlm_bridge (
            const sc_core::sc_module_name & name )  [inline], [explicit]
```

Constructor.

Parameters

name	bridge name.

7.54.3 Member Function Documentation

7.54.3.1 kind()

```
template<unsigned int BUSWIDTH>
const char * amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >::kind  [inline], [virtual]
```

Returns the kind string of this bridge.

7.54.3.2 is_overwrite_exok_with_ok()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >::is_overwrite_exok_with_ok  [inline]
```

Returns whether the bridge is configured to replace AMBA_PV_EXOKAY with AMBA_PV_OKAY.

7.54.3.3 set_overwrite_exok_with_ok()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >::set_overwrite_exok_with_ok (
            bool value )  [inline]
```

Sets whether the bridge is configured to replace AMBA_PV_EXOKAY with AMBA_PV_OKAY.

7.54.3.4 b_transport()

```
template<unsigned int BUSWIDTH>
void amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >::b_transport (
            int ,
            amba_pv_transaction & trans,
            sc_core::sc_time & t )  [inline], [protected], [virtual]
```

Blocking transport.
This version of the method converts an AMBA-PV transaction into an TLM 2.0 BP transaction. The main change is conversion of wrapping bursts into incremental burts. In addition, the transaction address is aligned to the burst size.
Implements amba_pv::amba_pv_fw_transport_if.

7.54.3.5 get_direct_mem_ptr()

```
template<unsigned int BUSWIDTH>
bool amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >::get_direct_mem_ptr (
            int ,
            amba_pv_transaction & trans,
            tlm::tlm_dmi & dmi_data )  [inline], [protected], [virtual]
```

Requests a DMI access based on the specified transaction.
This version of the method converts AMBA-PV DMI request into TLM 2.0 BP request.
Implements amba_pv::amba_pv_fw_transport_if.

7.54.3.6 transport_dbg()

```
template<unsigned int BUSWIDTH>
unsigned int amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >::transport_dbg (
            int ,
            amba_pv_transaction & trans )  [inline], [protected], [virtual]
```

Debug access to a target.
This version of the method converts AMBA-PV debug transaction into TLM 2.0 BP debug transaction.
Implements amba_pv::amba_pv_fw_transport_if.

7.54.4 Field Documentation

7.54.4.1 amba_pv_s

```
template<unsigned int BUSWIDTH = 64>
amba_pv_slave_socket<BUSWIDTH> amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >::amba_pv_s
```

Slave socket from AMBA-PV.

7.54.4.2 tlm_m

```
template<unsigned int BUSWIDTH = 64>
tlm_utils::simple_initiator_socket<amba_pv_to_tlm_bridge, BUSWIDTH, tlm::tlm_base_protocol_↩
types> amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >::tlm_m
```

Master socket to TLM 2.0 BP.

7.55 amba_pv::amba_pv_trans_lock Class Reference

AMBA-PV transaction lock wrapper.

`#include <user/amba_pv_mm.h>`

Public Member Functions

- amba_pv_trans_lock (amba_pv_transaction *, bool=true)

  Constructor.
- ~amba_pv_trans_lock ()

  Destructor.
- void acquire ()

  Acquires the ownership of the controlled transaction.
- void release ()

  Releases the ownership of the controlled transaction.

7.55.1 Detailed Description

AMBA-PV transaction lock wrapper.
amba_pv_trans_lock class allows for controlling transaction for a block of code and ensuring that it is released when the block is exited, whether by running off the end, by using a control flow statement such as `break` or `return`, or by throwing an exception.

Note

No two amba_pv_trans_lock instances can control the same transaction.

amba_pv_trans_lock is not an `sc_module`.

7.55.2 Constructor & Destructor Documentation

7.55.2.1 amba_pv_trans_lock()

```
amba_pv::amba_pv_trans_lock::amba_pv_trans_lock (
            amba_pv_transaction * trans,
            bool acquire = true )  [inline], [explicit]
```

Constructor.

Parameters

trans	transaction pointer to control.
acquire	`true` to acquire onwership of the transaction, `false` otherwise.

7.55.2.2 ~amba_pv_trans_lock()

`amba_pv::amba_pv_trans_lock::~amba_pv_trans_lock ( ) [inline]`
Destructor.

7.55.3 Member Function Documentation

7.55.3.1 acquire()

`void amba_pv::amba_pv_trans_lock::acquire ( ) [inline]`
Acquires the ownership of the controlled transaction.

7.55.3.2 release()

`void amba_pv::amba_pv_trans_lock::release ( ) [inline]`
Releases the ownership of the controlled transaction.

7.56 amba_pv::amba_pv_trans_pool Class Reference

AMBA-PV transaction pool.
`#include <user/amba_pv_mm.h>`
Inherits tlm::tlm_mm_interface.

Data Structures

- class transaction_allocator
  AMBA-PV transaction allocator.

Public Member Functions

- amba_pv_trans_pool (size_t=0, transaction_allocator *=NULL)
  Constructor.
- ~amba_pv_trans_pool ()
  Destructor.
- amba_pv_transaction * allocate ()
  Allocates and returns a new AMBA-PV transaction with associated AMBA-PV extension.
- amba_pv_transaction * allocate (unsigned int, const amba_pv_control *)
  Allocates and returns a new AMBA-PV transaction with associated AMBA-PV extension.
- amba_pv_transaction * allocate (unsigned int, unsigned int, const amba_pv_control *, amba_pv_burst_t)
  Allocates and returns a new AMBA-PV transaction.
- bool empty () const
  Returns whether this manager's pool is empty (i.e.
- size_t size () const
  Returns the number of transactions in this manager's pool.
- void reserve (size_t)
  Frees a previously allocated AMBA-PV transaction.

7.56.1 Detailed Description

AMBA-PV transaction pool.
This class implements the `tlm::tlm_mm_interface` and provides a memory pool from which transaction can be (de)allocated.

Note

amba_pv_trans_pool is not an `sc_module`.

7.56.2 Constructor & Destructor Documentation

7.56.2.1 amba_pv_trans_pool()

```
amba_pv::amba_pv_trans_pool::amba_pv_trans_pool (
            size_t n = 0,
            transaction_allocator * alloc = NULL )  [inline], [explicit]
```

Constructor.

Parameters

n	number of transactions initially allocated in the pool
alloc	transaction allocator for this pool.

7.56.2.2 ~amba_pv_trans_pool()

```
amba_pv::amba_pv_trans_pool::~amba_pv_trans_pool ( )  [inline]
```

Destructor.

7.56.3 Member Function Documentation

7.56.3.1 allocate() [1/3]

```
amba_pv_transaction * amba_pv::amba_pv_trans_pool::allocate ( )  [inline]
```

Allocates and returns a new AMBA-PV transaction with associated AMBA-PV extension.
allocate() will also (re)set the associated AMBA-PV extension.

7.56.3.2 allocate() [2/3]

```
amba_pv_transaction * amba_pv::amba_pv_trans_pool::allocate (
            unsigned int size,
            const amba_pv_control * ctrl )  [inline]
```

Allocates and returns a new AMBA-PV transaction with associated AMBA-PV extension.
allocate() will also (re)set the associated AMBA-PV extension.

Parameters

size	transaction size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128].
ctrl	optional AMBA 4 control information (set to `NULL` if unused).

7.56.3.3 allocate() [3/3]

```
amba_pv_transaction * amba_pv::amba_pv_trans_pool::allocate (
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst )  [inline]
```

Allocates and returns a new AMBA-PV transaction.
allocate() will also (re)set the associated AMBA-PV extension.

Parameters

length	transaction burst length as in [1-256].
size	transaction burst size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128].
ctrl	optional AMBA 4 control information (set to NULL if unused).
burst	transaction burst type as one of AMBA_PV_INCR, AMBA_PV_FIXED, AMBA_PV_WRAP.

7.56.3.4 empty()

```
bool amba_pv::amba_pv_trans_pool::empty ( ) const  [inline]
```

Returns whether this manager's pool is empty (i.e.
whether its size is 0).

7.56.3.5 size()

```
size_t amba_pv::amba_pv_trans_pool::size ( ) const  [inline]
```

Returns the number of transactions in this manager's pool.

7.56.3.6 reserve()

```
void amba_pv::amba_pv_trans_pool::reserve (
            size_t n )  [inline]
```

Frees a previously allocated AMBA-PV transaction.

7.57 amba_pv::amba_pv_trans_ptr Class Reference

AMBA-PV transaction smart pointer.
`#include <user/amba_pv_mm.h>`

Public Member Functions

- amba_pv_trans_ptr (amba_pv_transaction *=NULL)

  Constructor.
- ~amba_pv_trans_ptr ()

  Destructor.
- amba_pv_transaction * release ()

  Returns a pointer to the managed transaction and releases the ownership.
- void reset (amba_pv_transaction *=NULL)

  Replaces the managed transaction.
- void swap (amba_pv_trans_ptr &)

  Swaps the managed transactions.

7.57.1 Detailed Description

AMBA-PV transaction smart pointer.
amba_pv_trans_ptr is a smart pointer that retains sole ownership of a transaction through a pointer and releases that transaction when the amba_pv_trans_ptr goes out of scope.

Note

No two amba_pv_trans_ptr instances can manage the same transaction.

amba_pv_trans_ptr is not an `sc_module`.

7.57.2 Constructor & Destructor Documentation

7.57.2.1 amba_pv_trans_ptr()

```
amba_pv::amba_pv_trans_ptr::amba_pv_trans_ptr (
            amba_pv_transaction * trans = NULL )  [inline], [explicit]
```

Constructor.

Parameters

trans	transaction pointer to manage.

7.57.2.2 ~amba_pv_trans_ptr()

```
amba_pv::amba_pv_trans_ptr::~amba_pv_trans_ptr ( )  [inline]
```

Destructor.

7.57.3 Member Function Documentation

7.57.3.1 release()

```
amba_pv_transaction * amba_pv::amba_pv_trans_ptr::release ( )  [inline]
```

Returns a pointer to the managed transaction and releases the ownership.

7.57.3.2 reset()

```
void amba_pv::amba_pv_trans_ptr::reset (
            amba_pv_transaction * trans = NULL )  [inline]
```

Replaces the managed transaction.

Parameters

trans	pointer to new transaction to manage.

7.57.3.3 swap()

```
void amba_pv::amba_pv_trans_ptr::swap (
            amba_pv_trans_ptr & ptr )  [inline]
```

Swaps the managed transactions.

Parameters

ptr	transaction smart pointer to swap managed transactions with.

7.58 amba_pv::amba_pv_attributes::attribute_ref Class Reference

A reference to a specific attribute in a map of attributes that is not accessed until it is required.
`#include <bus/amba_pv_attributes.h>`
Inherits amba_pv::amba_pv_attributes::const_attribute_ref.

Public Member Functions

- attribute_ref (map_type &, const std::string &)

  Constructor.
- template<typename T >
  void set_value (T)

  Sets the value of this attribute reference.
- void set_value (const char *)

  Sets the value of this attribute reference.
- void set_value (const std::string &)

  Sets the value of this attribute reference.
- template<typename T >
  amba_pv_attributes::attribute_ref & operator= (T v)

  Sets the value of this attribute reference.

7.58.1 Detailed Description

A reference to a specific attribute in a map of attributes that is not accessed until it is required.

7.58.2 Constructor & Destructor Documentation

7.58.2.1 attribute_ref()

```
amba_pv::amba_pv_attributes::attribute_ref::attribute_ref (
            map_type & m,
            const std::string & n )  [inline]
```

Constructor.

Parameters

m	a reference to a map of attributes.
n	the name of the attribute.

7.58.3 Member Function Documentation

7.58.3.1 set_value() [1/3]

```
template<typename T >
void amba_pv::amba_pv_attributes::attribute_ref::set_value (
            T v )  [inline]
```

Sets the value of this attribute reference.

Parameters

v	the value to assign to the attribute.

7.58.3.2 set_value() [2/3]

```
void amba_pv::amba_pv_attributes::attribute_ref::set_value (
            const char * s )  [inline]
```

Sets the value of this attribute reference.

Parameters

s	the value to assign to the attribute.

7.58.3.3 set_value() [3/3]

```
void amba_pv::amba_pv_attributes::attribute_ref::set_value (
            const std::string & s )  [inline]
```

Sets the value of this attribute reference.

Parameters

s	the value to assign to the attribute.

7.58.3.4 operator=()

```
template<typename T >
amba_pv_attributes::attribute_ref & amba_pv::amba_pv_attributes::attribute_ref::operator= (
            T v )  [inline]
```

Sets the value of this attribute reference.

Parameters

v	the value to assign to the attribute.

7.59 amba_pv::amba_pv_attributes::const_attribute_ref Class Reference

A `const` reference to a specific attribute in a map of attributes that is not accessed until it is required.
`#include <bus/amba_pv_attributes.h>`
Inherited by amba_pv::amba_pv_attributes::attribute_ref.

Public Member Functions

- const_attribute_ref (const map_type &, const std::string &)

  Constructor.
- const std::string & get_name () const

  Returns the name of this attribute reference.

- template<typename T >
  bool get_value (T &) const

  Returns the value of this attribute reference.
- bool get_value (std::string &) const

  Returns the value of this attribute reference.

7.59.1 Detailed Description

A `const` reference to a specific attribute in a map of attributes that is not accessed until it is required.

7.59.2 Constructor & Destructor Documentation

7.59.2.1 const_attribute_ref()

```
amba_pv::amba_pv_attributes::const_attribute_ref::const_attribute_ref (
            const map_type & m,
            const std::string & n )  [inline]
```

Constructor.

Parameters

m	a `const` reference to a map of attributes.
n	the name of the attribute.

7.59.3 Member Function Documentation

7.59.3.1 get_name()

```
const std::string & amba_pv::amba_pv_attributes::const_attribute_ref::get_name ( ) const  [inline]
```

Returns the name of this attribute reference.

7.59.3.2 get_value() [1/2]

```
template<typename T >
bool amba_pv::amba_pv_attributes::const_attribute_ref::get_value (
            T & v ) const  [inline]
```

Returns the value of this attribute reference.

Parameters

v	the value of the attribute.

7.59.3.3 get_value() [2/2]

```
bool amba_pv::amba_pv_attributes::const_attribute_ref::get_value (
            std::string & s ) const  [inline]
```

Returns the value of this attribute reference.

Parameters

s	the value of the attribute.

7.60 amba_pv::atomic_subop_impl::do_add Struct Reference

A functor for in-place addition operation.
`#include <bus/amba_pv_atomic_subop_impl.h>`

Public Member Functions

- template<typename T >
  void operator() (unsigned char *memory, unsigned char *data) const

  Completes an addition operation.

7.60.1 Detailed Description

A functor for in-place addition operation.

7.60.2 Member Function Documentation

7.60.2.1 operator()()

```
template<typename T >
void amba_pv::atomic_subop_impl::do_add::operator() (
            unsigned char * memory,
            unsigned char * data ) const  [inline]
```

Completes an addition operation.

Template Parameters

T	integer type that aligns with the size of the atomic transaction.

Parameters

memory	data pointer pointing to the target address in the memory.
data	transaction data pointer.

7.61 amba_pv::atomic_subop_impl::do_bit_clear Struct Reference

A functor for in-place bit clear operation.
`#include <bus/amba_pv_atomic_subop_impl.h>`

Public Member Functions

- template<typename T >
  void operator() (unsigned char *memory, unsigned char *data) const

  Completes a bit clear operation.

7.61.1 Detailed Description

A functor for in-place bit clear operation.

7.61.2 Member Function Documentation

7.61.2.1 operator()()

```
template<typename T >
void amba_pv::atomic_subop_impl::do_bit_clear::operator() (
            unsigned char * memory,
            unsigned char * data ) const  [inline]
```

Completes a bit clear operation.

Template Parameters

T	integer type that aligns with the size of the atomic transaction.

Parameters

memory	data pointer pointing to the target address in the memory.
data	transaction data pointer.

7.62 amba_pv::atomic_subop_impl::do_bit_set Struct Reference

A functor for in-place bit set operation.

`#include <bus/amba_pv_atomic_subop_impl.h>`

Public Member Functions

- template<typename T >
  void operator() (unsigned char *memory, unsigned char *data) const

  Completes a bit set operation.

7.62.1 Detailed Description

A functor for in-place bit set operation.

7.62.2 Member Function Documentation

7.62.2.1 operator()()

```
template<typename T >
void amba_pv::atomic_subop_impl::do_bit_set::operator() (
            unsigned char * memory,
            unsigned char * data ) const  [inline]
```

Completes a bit set operation.

Template Parameters

T	integer type that aligns with the size of the atomic transaction.

Parameters

memory	data pointer pointing to the target address in the memory.
data	transaction data pointer.

7.63 amba_pv::atomic_subop_impl::do_signed_max Struct Reference

A functor for in-place signed max operation.
`#include <bus/amba_pv_atomic_subop_impl.h>`

Public Member Functions

- template<typename T >
  void operator() (unsigned char *memory, unsigned char *data) const

  Completes a signed max operation.

7.63.1 Detailed Description

A functor for in-place signed max operation.

7.63.2 Member Function Documentation

7.63.2.1 operator()()

```
template<typename T >
void amba_pv::atomic_subop_impl::do_signed_max::operator() (
            unsigned char * memory,
            unsigned char * data ) const  [inline]
```

Completes a signed max operation.

Template Parameters

T	integer type that aligns with the size of the atomic transaction.

Parameters

memory	data pointer pointing to the target address in the memory.
data	transaction data pointer.

7.64 amba_pv::atomic_subop_impl::do_signed_min Struct Reference

A functor for in-place signed min operation.
`#include <bus/amba_pv_atomic_subop_impl.h>`

Public Member Functions

- template<typename T >
  void operator() (unsigned char *memory, unsigned char *data) const

  Completes a signed min operation.

7.64.1 Detailed Description

A functor for in-place signed min operation.

7.64.2 Member Function Documentation

7.64.2.1 operator()()

```
template<typename T >
void amba_pv::atomic_subop_impl::do_signed_min::operator() (
            unsigned char * memory,
            unsigned char * data ) const  [inline]
```

Completes a signed min operation.

Template Parameters

T	integer type that aligns with the size of the atomic transaction.

Parameters

memory	data pointer pointing to the target address in the memory.
data	transaction data pointer.

7.65 amba_pv::atomic_subop_impl::do_unsigned_max Struct Reference

A functor for in-place unsigned max operation.

`#include <bus/amba_pv_atomic_subop_impl.h>`

Public Member Functions

- template<typename T >
  void operator() (unsigned char *memory, unsigned char *data) const

  Completes an unsigned max operation.

7.65.1 Detailed Description

A functor for in-place unsigned max operation.

7.65.2 Member Function Documentation

7.65.2.1 operator()()

```
template<typename T >
void amba_pv::atomic_subop_impl::do_unsigned_max::operator() (
            unsigned char * memory,
            unsigned char * data ) const  [inline]
```

Completes an unsigned max operation.

Template Parameters

T	integer type that aligns with the size of the atomic transaction.

Parameters

memory	data pointer pointing to the target address in the memory.
data	transaction data pointer.

7.66 amba_pv::atomic_subop_impl::do_unsigned_min Struct Reference

A functor for in-place unsigned min operation.
`#include <bus/amba_pv_atomic_subop_impl.h>`

Public Member Functions

- template<typename T >
  void operator() (unsigned char *memory, unsigned char *data) const

  Completes an unsigned min operation.

7.66.1 Detailed Description

A functor for in-place unsigned min operation.

7.66.2 Member Function Documentation

7.66.2.1 operator()()

```
template<typename T >
void amba_pv::atomic_subop_impl::do_unsigned_min::operator() (
            unsigned char * memory,
            unsigned char * data ) const  [inline]
```

Completes an unsigned min operation.

Template Parameters

T	integer type that aligns with the size of the atomic transaction.

Parameters

memory	data pointer pointing to the target address in the memory.
data	transaction data pointer.

7.67 amba_pv::atomic_subop_impl::do_xor Struct Reference

A functor for in-place exclusive or operation.
`#include <bus/amba_pv_atomic_subop_impl.h>`

Public Member Functions

- template<typename T >
  void operator() (unsigned char *memory, unsigned char *data) const

  Completes an exclusive or operation.

7.67.1 Detailed Description

A functor for in-place exclusive or operation.

7.67.2 Member Function Documentation

7.67.2.1 operator()()

```
template<typename T >
void amba_pv::atomic_subop_impl::do_xor::operator() (
            unsigned char * memory,
            unsigned char * data ) const  [inline]
```

Completes an exclusive or operation.

Template Parameters

T	integer type that aligns with the size of the atomic transaction.

Parameters

memory	data pointer pointing to the target address in the memory.
data	transaction data pointer.

7.68 amba_pv::nonblocking_transport_if< REQ, RSP > Class Template Reference

Non-blocking transport core interface.
`#include <signal/signal_core_ifs.h>`
Inherits sc_core::sc_interface.

Public Member Functions

- virtual RSP nb_transport (const REQ &req)=0

  Bidirectional non-blocking transport.

7.68.1 Detailed Description

template<typename REQ, typename RSP>
class amba_pv::nonblocking_transport_if< REQ, RSP >

Non-blocking transport core interface.
This is a non-blocking variant of the `tlm::tlm_transport_if` interface.

Parameters

REQ	request type.
RSP	response type.

7.68.2 Member Function Documentation

7.68.2.1 nb_transport()

```
template<typename REQ , typename RSP >
virtual RSP amba_pv::nonblocking_transport_if< REQ, RSP >::nb_transport (
            const REQ & req )  [pure virtual]
```

Bidirectional non-blocking transport.

Parameters

req	signal request.

Returns

the signal response.

7.69 amba_pv::signal_export_base Class Reference

Signal export base class.
`#include <signal/signal_slave_export.h>`
Inherited by amba_pv::signal_slave_export< STATE >, and amba_pv::signal_state_slave_export< STATE >.

Public Member Functions

- signal_export_base (int=0)
  Constructor.
- int get_export_id () const
  Returns the export identifier.
- void set_export_id (int)
  Sets the export identifier.

7.69.1 Detailed Description

Signal export base class.

7.69.2 Constructor & Destructor Documentation

7.69.2.1 signal_export_base()

```
amba_pv::signal_export_base::signal_export_base (
            int export_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

export_id	export identifier (defaults to 0).

7.69.3 Member Function Documentation

7.69.3.1 get_export_id()

```
int amba_pv::signal_export_base::get_export_id ( ) const  [inline]
```

Returns the export identifier.

7.69.3.2 set_export_id()

```
void amba_pv::signal_export_base::set_export_id (
            int export_id )  [inline]
```

Sets the export identifier.

Parameters

export_id	export identifier.

7.70 amba_pv::signal_from_sc_bridge< STATE > Class Template Reference

Generic sc_signal to Signal bridge module.
`#include <signal/signal_bridges.h>`
Inherits sc_core::sc_module.

Public Member Functions

- signal_from_sc_bridge (const sc_core::sc_module_name &)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this bridge.

Data Fields

- sc_core::sc_in< STATE > signal_in

  SystemC signal in.
- signal_master_port< STATE > signal_m

  Master port to Signal.

7.70.1 Detailed Description

template<typename STATE>
class amba_pv::signal_from_sc_bridge< STATE >

Generic sc_signal to Signal bridge module.
The signal_from_sc_bridge class translates `sc_signal` events into Signal transactions.

Parameters

STATE	signal state type.

7.70.2 Constructor & Destructor Documentation

7.70.2.1 signal_from_sc_bridge()

```
template<typename STATE >
amba_pv::signal_from_sc_bridge< STATE >::signal_from_sc_bridge (
            const sc_core::sc_module_name & name )  [inline], [explicit]
```

Constructor.

Parameters

name	bridge name.

7.70.3 Member Function Documentation

7.70.3.1 kind()

```
template<typename STATE >
const char * amba_pv::signal_from_sc_bridge< STATE >::kind  [inline], [virtual]
```
Returns the kind string of this bridge.

7.70.4 Field Documentation

7.70.4.1 signal_in

```
template<typename STATE >
sc_core::sc_in<STATE> amba_pv::signal_from_sc_bridge< STATE >::signal_in
```
SystemC signal in.

7.70.4.2 signal_m

```
template<typename STATE >
signal_master_port<STATE> amba_pv::signal_from_sc_bridge< STATE >::signal_m
```
Master port to Signal.

7.71 amba_pv::signal_if< STATE > Class Template Reference

Signal interface.
`#include <signal/signal_if.h>`
Inherited by amba_pv::signal_master_port< STATE, N, POL >[virtual], amba_pv::signal_slave_base< STATE >[virtual], and amba_pv::signal_state_if< STATE >[virtual].

Public Member Functions

- virtual ~signal_if ()

  Virtual destructor.
- virtual void set_state (int export_id, const STATE &state)=0

  Transfers a signal state.

7.71.1 Detailed Description

template<typename STATE>
class amba_pv::signal_if< STATE >

Signal interface.
It is used to indicate changes in the state of side-band signals such as, for example, interrupts.
This interface is implemented by signal_master_port and must be implemented into slave modules inheriting from signal_slave_base.

Parameters

STATE	signal state type.

7.71.2 Constructor & Destructor Documentation

7.71.2.1 ~signal_if()

```
template<typename STATE >
virtual amba_pv::signal_if< STATE >::~signal_if ( )  [inline], [virtual]
```

Virtual destructor.

7.71.3 Member Function Documentation

7.71.3.1 set_state()

```
template<typename STATE >
virtual void amba_pv::signal_if< STATE >::set_state (
            int export_id,
            const STATE & state )  [pure virtual]
```

Transfers a signal state.
set_state() is used to indicate a change in the state of the signal.

Parameters

export_id	export identifier, for multi-export slave (index into bound interfaces on master side).
state	signal state.

Implemented in amba_pv::signal_master_port< bool, 0, sc_core::SC_ZERO_OR_MORE_BOUND >, amba_pv::signal_master_port<
amba_pv::signal_state_master_port< STATE, N, POL >, amba_pv::signal_slave_base< STATE >, and amba_pv::signal_state_slav

7.72 amba_pv::signal_master_port< STATE, N, POL > Class Template Reference

Signal port to be instantiated on the master side.
`#include <signal/signal_master_port.h>`
Inherits amba_pv::signal_if< STATE >, and sc_core::sc_port< nonblocking_transport_if< signal_request< STATE >, signal_response< STATE > >, 1, sc_core::SC_ONE_OR_MORE_BOUND >.

Public Member Functions

- signal_master_port ()
  Default constructor.
- signal_master_port (const char *)
  Parameterized constructor.
- virtual const char * kind () const
  Returns the kind string of this port.
- virtual void set_state (int, const STATE &)
  Transfers a signal state.
- void set_state (const STATE &)
  Transfers a signal state.

7.72.1 Detailed Description

template<typename STATE, int N = 1, sc_core::sc_port_policy POL = sc_core::SC_ONE_OR_MORE_BOUND>
class amba_pv::signal_master_port< STATE, N, POL >

Signal port to be instantiated on the master side.
This port is for use as a master port bound to one or more slave exports.
signal_master_port provides an implementation of the signal_if interface.

Parameters

STATE	signal state type.
N	number of bindings; defaults to 1.
POL	port binding policy; defaults to `sc_core::SC_ONE_OR_MORE_BOUND`.

7.72.2 Constructor & Destructor Documentation

7.72.2.1 signal_master_port() [1/2]

```
template<typename STATE , int N, sc_core::sc_port_policy POL>
amba_pv::signal_master_port< STATE, N, POL >::signal_master_port  [inline]
```

Default constructor.

7.72.2.2 signal_master_port() [2/2]

```
template<typename STATE , int N, sc_core::sc_port_policy POL>
amba_pv::signal_master_port< STATE, N, POL >::signal_master_port (
            const char * name )  [inline], [explicit]
```

Parameterized constructor.

Parameters

name	port name.

7.72.3 Member Function Documentation

7.72.3.1 kind()

```
template<typename STATE , int N, sc_core::sc_port_policy POL>
const char * amba_pv::signal_master_port< STATE, N, POL >::kind  [inline], [virtual]
```

Returns the kind string of this port.

7.72.3.2 set_state() [1/2]

```
template<typename STATE , int N, sc_core::sc_port_policy POL>
void amba_pv::signal_master_port< STATE, N, POL >::set_state (
            int index,
            const STATE & state )  [inline], [virtual]
```

Transfers a signal state.
This version of the method forward the user-layer set_state() call to the Signal core interface signal_transport_if.

Parameters

index	interface index (for ports bound more than once).
state	signal state.

Implements amba_pv::signal_if< STATE >.

7.72.3.3 set_state() [2/2]

```
template<typename STATE , int N, sc_core::sc_port_policy POL>
void amba_pv::signal_master_port< STATE, N, POL >::set_state (
             const STATE & state )  [inline]
```

Transfers a signal state.

Parameters

state	signal state.

7.73 amba_pv::signal_request< STATE > Class Template Reference

Signal request type.
`#include <signal/signal_request.h>`

Public Member Functions

- signal_request ()
  Default constructor.
- signal_request (const STATE &)
  Signal request constructor.
- signal_command get_command () const
  Returns the command of this request.
- void set_command (const enum signal_command)
  Sets the command of this request.
- STATE get_state () const
  Returns the signal state.
- void set_state (const STATE &)
  Sets the signal state.

7.73.1 Detailed Description

template<typename STATE>
class amba_pv::signal_request< STATE >

Signal request type.
This class is for use with the signal_transport_if and signal_state_transport_if core interfaces.

Parameters

STATE	signal state type.

7.73.2 Constructor & Destructor Documentation

7.73.2.1 signal_request() [1/2]

```
template<typename STATE >
amba_pv::signal_request< STATE >::signal_request  [inline]
```

Default constructor.

7.73.2.2 signal_request() [2/2]

```
template<typename STATE >
amba_pv::signal_request< STATE >::signal_request (
            const STATE & state )  [inline]
```

Signal request constructor.

Parameters

state	signal state.

7.73.3 Member Function Documentation

7.73.3.1 get_command()

```
template<typename STATE >
signal_command amba_pv::signal_request< STATE >::get_command  [inline]
```

Returns the command of this request.

7.73.3.2 set_command()

```
template<typename STATE >
void amba_pv::signal_request< STATE >::set_command (
            const enum signal_command command )  [inline]
```

Sets the command of this request.

Parameters

command	the command of this Signal request.

7.73.3.3 get_state()

```
template<typename STATE >
STATE amba_pv::signal_request< STATE >::get_state  [inline]
```

Returns the signal state.

7.73.3.4 set_state()

```
template<typename STATE >
void amba_pv::signal_request< STATE >::set_state (
            const STATE & state )  [inline]
```

Sets the signal state.

Parameters

state	signal state.

7.74 amba_pv::signal_response< STATE > Class Template Reference

Signal response type.
`#include <signal/signal_response.h>`

Public Member Functions

- signal_response ()
  Default constructor.
- signal_response (const STATE &)
  Signal response constructor.
- void set_state (const STATE &)
  Sets the signal state.
- STATE get_state () const
  Returns the signal state.

7.74.1 Detailed Description

template<typename STATE>
class amba_pv::signal_response< STATE >

Signal response type.
This class is for use with the signal_transport_if core interface.

Parameters

STATE	signal state type.

7.74.2 Constructor & Destructor Documentation

7.74.2.1 signal_response() [1/2]

```
template<typename STATE >
amba_pv::signal_response< STATE >::signal_response  [inline]
```

Default constructor.

7.74.2.2 signal_response() [2/2]

```
template<typename STATE >
amba_pv::signal_response< STATE >::signal_response (
            const STATE & state )  [inline]
```

Signal response constructor.

Parameters

state	signal state.

7.74.3 Member Function Documentation

7.74.3.1 set_state()

```
template<typename STATE >
void amba_pv::signal_response< STATE >::set_state (
            const STATE & state )  [inline]
```

Sets the signal state.

Parameters

state	signal state.

7.74.3.2 get_state()

```
template<typename STATE >
STATE amba_pv::signal_response< STATE >::get_state  [inline]
```

Returns the signal state.

7.75 amba_pv::signal_slave_base< STATE > Class Template Reference

Base class for all Signal slave modules.
`#include <signal/signal_slave_base.h>`
Inherits amba_pv::signal_transport_if< STATE >, and amba_pv::signal_if< STATE >.

Public Member Functions

- signal_slave_base (const std::string &)
  Constructor.
- std::string get_name () const
  Returns the name of this slave.

Protected Member Functions

- virtual void set_state (int, const STATE &)
  Transfers a signal state.
- virtual response_type nb_transport (int, const request_type &)
  Completes a transaction.

7.75.1 Detailed Description

template<typename STATE>
class amba_pv::signal_slave_base< STATE >

Base class for all Signal slave modules.

Note

signal_slave_base is not an `sc_module`.

Parameters

STATE	signal state type.

7.75.2 Constructor & Destructor Documentation

7.75.2.1 signal_slave_base()

```
template<typename STATE >
amba_pv::signal_slave_base< STATE >::signal_slave_base (
            const std::string & name )  [inline], [explicit]
```

Constructor.

Parameters

name	slave name.

7.75.3 Member Function Documentation

7.75.3.1 get_name()

```
template<typename STATE >
std::string amba_pv::signal_slave_base< STATE >::get_name  [inline]
```

Returns the name of this slave.

7.75.3.2 set_state()

```
template<typename STATE >
void amba_pv::signal_slave_base< STATE >::set_state (
            int export_id,
            const STATE & state )  [inline], [protected], [virtual]
```

Transfers a signal state.
This version of the method causes an error.
Implements amba_pv::signal_if< STATE >.

7.75.3.3 nb_transport()

```
template<typename STATE >
signal_response< STATE > amba_pv::signal_slave_base< STATE >::nb_transport (
            int export_id,
            const request_type & req )  [inline], [protected], [virtual]
```

Completes a transaction.
This version of the method translates this nb_transport() call into signal_if user-layer calls.
Implements amba_pv::signal_transport_if< STATE >.

7.76 amba_pv::signal_slave_export< STATE > Class Template Reference

Signal export to be instantiated on the slave side.
`#include <signal/signal_slave_export.h>`
Inherits amba_pv::nonblocking_transport_if< signal_request< STATE >, signal_response< STATE > >, amba_pv::signal_export_b
and sc_core::sc_export< nonblocking_transport_if< signal_request< STATE >, signal_response< STATE > >
>.

Public Member Functions

- signal_slave_export ()

  Default constructor.
- signal_slave_export (const char ∗, int=0)

  Constructor.
- virtual const char ∗ kind () const

  Returns the kind string of this export.
- void bind (signal_slave_export &)

  Binds an export to this export (hierarchical binding).
- void operator() (signal_slave_export &)

  Binds an export to this export (hierarchical binding).
- void bind (signal_transport_if< STATE > &)

  Binds the specified interface to this export.
- void operator() (signal_transport_if< STATE > &)

  Binds the specified interface to this export.

7.76.1 Detailed Description

template<typename STATE>
class amba_pv::signal_slave_export< STATE >

Signal export to be instantiated on the slave side.

Parameters

STATE	signal state type.

7.76.2 Constructor & Destructor Documentation

7.76.2.1 signal_slave_export() [1/2]

```
template<typename STATE >
amba_pv::signal_slave_export< STATE >::signal_slave_export  [inline]
```

Default constructor.

7.76.2.2 signal_slave_export() [2/2]

```
template<typename STATE >
amba_pv::signal_slave_export< STATE >::signal_slave_export (
            const char * name,
            int export_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	export name.
export↩ _id	export identifier (for multi-export slave). Defaults to 0.

7.76.3 Member Function Documentation

7.76.3.1 kind()

```
template<typename STATE >
const char * amba_pv::signal_slave_export< STATE >::kind  [inline], [virtual]
```

Returns the kind string of this export.

7.76.3.2 bind() [1/2]

```
template<typename STATE >
void amba_pv::signal_slave_export< STATE >::bind (
            signal_slave_export< STATE > & parent )  [inline]
```

Binds an export to this export (hierarchical binding).

Parameters

parent	export to bind to this export.

7.76.3.3 operator()() [1/2]

```
template<typename STATE >
void amba_pv::signal_slave_export< STATE >::operator() (
            signal_slave_export< STATE > & parent )  [inline]
```

Binds an export to this export (hierarchical binding).

Parameters

parent	export to bind to this export.

7.76.3.4 bind() [2/2]

```
template<typename STATE >
void amba_pv::signal_slave_export< STATE >::bind (
            signal_transport_if< STATE > & iface )  [inline]
```

Binds the specified interface to this export.

Parameters

iface	signal_transport_if interface to bind to this export.

7.76.3.5 operator()() [2/2]

```
template<typename STATE >
void amba_pv::signal_slave_export< STATE >::operator() (
            signal_transport_if< STATE > & iface )  [inline]
```

Binds the specified interface to this export.

Parameters

iface	signal_transport_if interface to bind to this export.

7.77 amba_pv::signal_state_from_sc_bridge< STATE > Class Template Reference

Generic sc_signal to SignalState bridge module.
`#include <signal/signal_bridges.h>`
Inherits sc_core::sc_module.

Public Member Functions

- signal_state_from_sc_bridge (const sc_core::sc_module_name &)
  Constructor.
- virtual const char * kind () const
  Returns the kind string of this bridge.

Data Fields

- sc_core::sc_in< STATE > signal_in
  SystemC signal in.
- signal_state_master_port< STATE > signal_m
  Master port to SignalState.

7.77.1 Detailed Description

template<typename STATE>
class amba_pv::signal_state_from_sc_bridge< STATE >

Generic sc_signal to SignalState bridge module.
The signal_state_from_sc_bridge class translates `sc_signal` events into SignalState transactions.

Parameters

STATE	signal state type.

7.77.2 Constructor & Destructor Documentation

7.77.2.1 signal_state_from_sc_bridge()

```
template<typename STATE >
amba_pv::signal_state_from_sc_bridge< STATE >::signal_state_from_sc_bridge (
            const sc_core::sc_module_name & name )  [inline], [explicit]
```

Constructor.

Parameters

name	bridge name.

7.77.3 Member Function Documentation

7.77.3.1 kind()

```
template<typename STATE >
const char * amba_pv::signal_state_from_sc_bridge< STATE >::kind  [inline], [virtual]
```
Returns the kind string of this bridge.

7.77.4 Field Documentation

7.77.4.1 signal_in

```
template<typename STATE >
sc_core::sc_in<STATE> amba_pv::signal_state_from_sc_bridge< STATE >::signal_in
```
SystemC signal in.

7.77.4.2 signal_m

```
template<typename STATE >
signal_state_master_port<STATE> amba_pv::signal_state_from_sc_bridge< STATE >::signal_m
```
Master port to SignalState.

7.78 amba_pv::signal_state_if< STATE > Class Template Reference

SignalState interface.
`#include <signal/signal_if.h>`
Inherits amba_pv::signal_if< STATE >.
Inherited by amba_pv::signal_state_master_port< STATE, N, POL > [virtual], and amba_pv::signal_state_slave_base< STATE

Public Member Functions

- virtual STATE get_state (int export_id, tlm::tlm_tag< STATE > *=NULL)=0

  Retrieves the signal state.

7.78.1 Detailed Description

template<typename STATE>
class amba_pv::signal_state_if< STATE >

SignalState interface.
It is used to indicate changes in the state of side-band signals such as, for example, interrupts and to retrieve the state.
This interface is implemented by signal_state_master_port and must be implemented into slave modules inheriting from signal_state_slave_base.

Parameters

STATE	signal state type.

7.78.2 Member Function Documentation

7.78.2.1 get_state()

```
template<typename STATE >
virtual STATE amba_pv::signal_state_if< STATE >::get_state (
            int export_id,
            tlm::tlm_tag< STATE > *  = NULL )  [pure virtual]
```

Retrieves the signal state.

Parameters

export_id	export identifier, for multi-export slave (index into bound interfaces on master side).

Returns

the signal state.

Implemented in amba_pv::signal_state_master_port< STATE, N, POL >, and amba_pv::signal_state_slave_base< STATE >.

7.79 amba_pv::signal_state_master_port< STATE, N, POL > Class Template Reference

SignalState port to be instantiated on the master side.
`#include <signal/signal_master_port.h>`
Inherits amba_pv::signal_state_if< STATE >, and sc_core::sc_port< nonblocking_transport_if< signal_request< STATE >, signal_response< STATE > >, 1, sc_core::SC_ONE_OR_MORE_BOUND >.

Public Member Functions

- signal_state_master_port ()
  Default constructor.
- signal_state_master_port (const char *)
  Parameterized constructor.
- virtual const char * kind () const
  Returns the kind string of this port.
- virtual void set_state (int, const STATE &)
  Transfers a signal state.
- void set_state (const STATE &)
  Transfers a signal state.
- virtual STATE get_state (int, tlm::tlm_tag< STATE > *=NULL)
  Retrieves the signal state.
- STATE get_state (tlm::tlm_tag< STATE > *=NULL)
  Retrieves the signal state.

7.79.1 Detailed Description

template<typename STATE, int N = 1, sc_core::sc_port_policy POL = sc_core::SC_ONE_OR_MORE_BOUND>
class amba_pv::signal_state_master_port< STATE, N, POL >

SignalState port to be instantiated on the master side.
This port is for use as a master port bound to one or more slave exports.
signal_state_master_port provides an implementation of the signal_state_if interface.

Parameters

STATE	signal state type.
N	number of bindings; defaults to 1.
POL	port binding policy; defaults to `sc_core::SC_ONE_OR_MORE_BOUND`.

7.79.2 Constructor & Destructor Documentation

7.79.2.1 signal_state_master_port() [1/2]

```
template<typename STATE , int N, sc_core::sc_port_policy POL>
amba_pv::signal_state_master_port< STATE, N, POL >::signal_state_master_port  [inline]
```

Default constructor.

7.79.2.2 signal_state_master_port() [2/2]

```
template<typename STATE , int N, sc_core::sc_port_policy POL>
amba_pv::signal_state_master_port< STATE, N, POL >::signal_state_master_port (
            const char * name )  [inline], [explicit]
```

Parameterized constructor.

Parameters

name	port name.

7.79.3 Member Function Documentation

7.79.3.1 kind()

```
template<typename STATE , int N, sc_core::sc_port_policy POL>
const char * amba_pv::signal_state_master_port< STATE, N, POL >::kind  [inline], [virtual]
```

Returns the kind string of this port.

7.79.3.2 set_state() [1/2]

```
template<typename STATE , int N, sc_core::sc_port_policy POL>
void amba_pv::signal_state_master_port< STATE, N, POL >::set_state (
            int index,
            const STATE & state )  [inline], [virtual]
```

Transfers a signal state.

This version of the method forwards the user-layer set_state() call to the SignalState core interface signal_state_transport_if.

Parameters

index	interface index (for ports bound more than once).
state	signal state.

Implements amba_pv::signal_if< STATE >.

7.79.3.3 set_state() [2/2]

```
template<typename STATE , int N, sc_core::sc_port_policy POL>
void amba_pv::signal_state_master_port< STATE, N, POL >::set_state (
            const STATE & state )  [inline]
```

Transfers a signal state.

Parameters

state	signal state.

7.79.3.4 get_state() [1/2]

```
template<typename STATE , int N, sc_core::sc_port_policy POL>
STATE amba_pv::signal_state_master_port< STATE, N, POL >::get_state (
            int index,
            tlm::tlm_tag< STATE > *  = NULL )  [inline], [virtual]
```

Retrieves the signal state.
This version of the method forwards the user-layer get_state() call to the Signal core interface signal_transport_if.

Parameters

index	interface index (for ports bound more than once).

Implements amba_pv::signal_state_if< STATE >.

7.79.3.5 get_state() [2/2]

```
template<typename STATE , int N, sc_core::sc_port_policy POL>
STATE amba_pv::signal_state_master_port< STATE, N, POL >::get_state (
            tlm::tlm_tag< STATE > *  = NULL )  [inline]
```

Retrieves the signal state.

7.80 amba_pv::signal_state_slave_base< STATE > Class Template Reference

Base class for all SignalState slave modules.
`#include <signal/signal_slave_base.h>`
Inherits amba_pv::signal_state_transport_if< STATE >, and amba_pv::signal_state_if< STATE >.

Public Member Functions

- signal_state_slave_base (const std::string &)
  Constructor.
- std::string get_name () const
  Returns the name of this slave.

Protected Member Functions

- virtual void set_state (int, const STATE &)
  Transfers a signal state.
- virtual STATE get_state (int, tlm::tlm_tag< STATE > *=NULL)
  Retrieves the signal state.
- virtual response_type nb_transport (int, const request_type &)
  Completes a transaction.

7.80.1 Detailed Description

template<typename STATE>
class amba_pv::signal_state_slave_base< STATE >

Base class for all SignalState slave modules.
signal_state_slave_base can be used instead of signal_slave_base. This enables combining signal_slave_export and signal_state_slave_export in the same `sc_module`.

Note

signal_state_slave_base is not an `sc_module`.

Parameters

STATE	signal state type.

7.80.2 Constructor & Destructor Documentation

7.80.2.1 signal_state_slave_base()

```
template<typename STATE >
amba_pv::signal_state_slave_base< STATE >::signal_state_slave_base (
            const std::string & name )  [inline], [explicit]
```

Constructor.

Parameters

name	slave name.

7.80.3 Member Function Documentation

7.80.3.1 get_name()

```
template<typename STATE >
std::string amba_pv::signal_state_slave_base< STATE >::get_name  [inline]
```

Returns the name of this slave.

7.80.3.2 set_state()

```
template<typename STATE >
void amba_pv::signal_state_slave_base< STATE >::set_state (
            int export_id,
            const STATE & state )  [inline], [protected], [virtual]
```

Transfers a signal state.
This version of the method causes an error.
Implements amba_pv::signal_if< STATE >.

7.80.3.3 get_state()

```
template<typename STATE >
STATE amba_pv::signal_state_slave_base< STATE >::get_state (
```

```
            int export_id,
            tlm::tlm_tag< STATE > *  = NULL )  [inline], [protected], [virtual]
```

Retrieves the signal state.
This version of the method causes an error.
Implements amba_pv::signal_state_if< STATE >.

7.80.3.4 nb_transport()

```
template<typename STATE >
signal_response< STATE > amba_pv::signal_state_slave_base< STATE >::nb_transport (
            int export_id,
            const request_type & req )  [inline], [protected], [virtual]
```

Completes a transaction.
This version of the method translates this nb_transport() call into signal_state_if user-layer calls.
Implements amba_pv::signal_transport_if< STATE >.

7.81 amba_pv::signal_state_slave_export< STATE > Class Template Reference

SignalState export to be instantiated on the slave side.
`#include <signal/signal_slave_export.h>`
Inherits amba_pv::nonblocking_transport_if< signal_request< STATE >, signal_response< STATE > >, amba_pv::signal_export_b
and sc_core::sc_export< nonblocking_transport_if< signal_request< STATE >, signal_response< STATE > >
>.

Public Member Functions

- signal_state_slave_export ()
  Default constructor.
- signal_state_slave_export (const char *, int=0)
  Constructor.
- virtual const char * kind () const
  Returns the kind string of this export.
- void bind (signal_state_slave_export &)
  Binds an export to this export (hierarchical binding).
- void operator() (signal_state_slave_export &)
  Binds an export to this export (hierarchical binding).
- void bind (signal_state_transport_if< STATE > &)
  Binds the specified interface to this export.
- void operator() (signal_state_transport_if< STATE > &)
  Binds the specified interface to this export.

7.81.1 Detailed Description

template<typename STATE>
class amba_pv::signal_state_slave_export< STATE >

SignalState export to be instantiated on the slave side.

Parameters

STATE	signal state type.

7.81.2 Constructor & Destructor Documentation

7.81.2.1 signal_state_slave_export() [1/2]

```
template<typename STATE >
amba_pv::signal_state_slave_export< STATE >::signal_state_slave_export  [inline]
```

Default constructor.

7.81.2.2 signal_state_slave_export() [2/2]

```
template<typename STATE >
amba_pv::signal_state_slave_export< STATE >::signal_state_slave_export (
            const char * name,
            int export_id = 0 )  [inline], [explicit]
```

Constructor.

Parameters

name	export name.
export_id	export identifier (for multi-export slave). Defaults to 0.

7.81.3 Member Function Documentation

7.81.3.1 kind()

```
template<typename STATE >
const char * amba_pv::signal_state_slave_export< STATE >::kind  [inline], [virtual]
```

Returns the kind string of this export.

7.81.3.2 bind() [1/2]

```
template<typename STATE >
void amba_pv::signal_state_slave_export< STATE >::bind (
            signal_state_slave_export< STATE > & parent )  [inline]
```

Binds an export to this export (hierarchical binding).

Parameters

parent	export to bind to this export.

7.81.3.3 operator()() [1/2]

```
template<typename STATE >
void amba_pv::signal_state_slave_export< STATE >::operator() (
            signal_state_slave_export< STATE > & parent )  [inline]
```

Binds an export to this export (hierarchical binding).

Parameters

Parameters

parent	export to bind to this export.

7.81.3.4 bind() [2/2]

```
template<typename STATE >
void amba_pv::signal_state_slave_export< STATE >::bind (
            signal_state_transport_if< STATE > & iface )  [inline]
```

Binds the specified interface to this export.

Parameters

iface	signal_state_transport_if interface to bind to this export.

7.81.3.5 operator()() [2/2]

```
template<typename STATE >
void amba_pv::signal_state_slave_export< STATE >::operator() (
            signal_state_transport_if< STATE > & iface )  [inline]
```

Binds the specified interface to this export.

Parameters

iface	signal_state_transport_if interface to bind to this export.

7.82 amba_pv::signal_state_to_sc_bridge< STATE > Class Template Reference

Generic SignalState to sc_signal bridge module.

`#include <signal/signal_bridges.h>`

Inherits amba_pv::signal_state_transport_if< STATE >, and sc_core::sc_module.

Public Member Functions

- signal_state_to_sc_bridge (const sc_core::sc_module_name &)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this bridge.

Data Fields

- signal_state_slave_export< STATE > signal_s

  SignalState slave export.
- sc_core::sc_out< STATE > signal_out

  Out sc_signal.

Protected Member Functions

- virtual response_type nb_transport (int, const request_type &)

  Bidirectional non-blocking transport.

7.82.1 Detailed Description

template<typename STATE>
class amba_pv::signal_state_to_sc_bridge< STATE >

Generic SignalState to sc_signal bridge module.
The signal_state_to_sc_bridge class translates SignalState transactions into `sc_signal` events.

Parameters

STATE	signal state type.

7.82.2 Constructor & Destructor Documentation

7.82.2.1 signal_state_to_sc_bridge()

```
template<typename STATE >
amba_pv::signal_state_to_sc_bridge< STATE >::signal_state_to_sc_bridge (
            const sc_core::sc_module_name & name )  [inline], [explicit]
```

Constructor.

Parameters

name	bridge name.

7.82.3 Member Function Documentation

7.82.3.1 kind()

```
template<typename STATE >
const char * amba_pv::signal_state_to_sc_bridge< STATE >::kind  [inline], [virtual]
```

Returns the kind string of this bridge.

7.82.3.2 nb_transport()

```
template<typename STATE >
signal_response< STATE > amba_pv::signal_state_to_sc_bridge< STATE >::nb_transport (
            int export_id,
            const request_type & req )  [inline], [protected], [virtual]
```

Bidirectional non-blocking transport.
This version of the method translates SignalState transactions into `sc_signal` events.
Implements amba_pv::signal_transport_if< STATE >.

7.82.4 Field Documentation

7.82.4.1 signal_s

```
template<typename STATE >
signal_state_slave_export<STATE> amba_pv::signal_state_to_sc_bridge< STATE >::signal_s
```

SignalState slave export.

7.82.4.2 signal_out

```
template<typename STATE >
sc_core::sc_out<STATE> amba_pv::signal_state_to_sc_bridge< STATE >::signal_out
```

Out sc_signal.

7.83 amba_pv::signal_state_transport_if< STATE > Class Template Reference

SignalState core interface.
`#include <signal/signal_core_ifs.h>`
Inherits amba_pv::signal_transport_if< STATE >.
Inherited by amba_pv::signal_state_slave_base< STATE >[virtual], and amba_pv::signal_state_to_sc_bridge< STATE >[vir

Additional Inherited Members

7.83.1 Detailed Description

template<typename STATE>
class amba_pv::signal_state_transport_if< STATE >

SignalState core interface.
This is an indirect tagged variant of the nonblocking_transport_if interface through signal_transport_if.

Parameters

STATE	signal state type.

7.84 amba_pv::signal_to_sc_bridge< STATE > Class Template Reference

Generic Signal to sc_signal bridge module.
`#include <signal/signal_bridges.h>`
Inherits amba_pv::signal_transport_if< STATE >, and sc_core::sc_module.

Public Member Functions

- signal_to_sc_bridge (const sc_core::sc_module_name &)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this bridge.

Data Fields

- signal_slave_export< STATE > signal_s

  Signal slave export.
- sc_core::sc_out< STATE > signal_out

  Out sc_signal.

Protected Member Functions

- virtual response_type nb_transport (int, const request_type &)

  Bidirectional non-blocking transport.

7.84.1 Detailed Description

template<typename STATE>
class amba_pv::signal_to_sc_bridge< STATE >

Generic Signal to sc_signal bridge module.
The signal_to_sc_bridge class translates Signal transactions into `sc_signal` events.

Parameters

STATE	signal state type.

7.84.2 Constructor & Destructor Documentation

7.84.2.1 signal_to_sc_bridge()

```
template<typename STATE >
amba_pv::signal_to_sc_bridge< STATE >::signal_to_sc_bridge (
            const sc_core::sc_module_name & name )  [inline], [explicit]
```

Constructor.

Parameters

name	bridge name.

7.84.3 Member Function Documentation

7.84.3.1 kind()

```
template<typename STATE >
const char * amba_pv::signal_to_sc_bridge< STATE >::kind  [inline], [virtual]
```

Returns the kind string of this bridge.

7.84.3.2 nb_transport()

```
template<typename STATE >
signal_response< STATE > amba_pv::signal_to_sc_bridge< STATE >::nb_transport (
            int export_id,
            const request_type & req )  [inline], [protected], [virtual]
```

Bidirectional non-blocking transport.
This version of the method translates Signal transactions into `sc_signal` events.
Implements amba_pv::signal_transport_if< STATE >.

7.84.4 Field Documentation

7.84.4.1 signal_s

```
template<typename STATE >
signal_slave_export<STATE> amba_pv::signal_to_sc_bridge< STATE >::signal_s
```

Signal slave export.

7.84.4.2 signal_out

```
template<typename STATE >
sc_core::sc_out<STATE> amba_pv::signal_to_sc_bridge< STATE >::signal_out
```

Out sc_signal.

7.85 amba_pv::signal_transport_if< STATE > Class Template Reference

Signal core interface.
`#include <signal/signal_core_ifs.h>`
Inherits sc_core::sc_interface.
Inherited by amba_pv::signal_slave_base< STATE >[virtual], amba_pv::signal_state_transport_if< STATE >[virtual], and amba_pv::signal_to_sc_bridge< STATE >[virtual].

Public Member Functions

- virtual response_type nb_transport (int export_id, const request_type &req)=0

  Bidirectional non-blocking transport.

7.85.1 Detailed Description

template<typename STATE>
class amba_pv::signal_transport_if< STATE >

Signal core interface.
This is a tagged variant of the nonblocking_transport_if interface.

Parameters

STATE	signal state type.

7.85.2 Member Function Documentation

7.85.2.1 nb_transport()

```
template<typename STATE >
virtual response_type amba_pv::signal_transport_if< STATE >::nb_transport (
            int export_id,
            const request_type & req )  [pure virtual]
```

Bidirectional non-blocking transport.

Parameters

export_id	export identifier (index into bound interfaces on master side).
req	signal request.

Returns

the signal response.

Implemented in amba_pv::signal_to_sc_bridge< STATE >, amba_pv::signal_state_to_sc_bridge< STATE >, amba_pv::signal_slave_base< STATE >, and amba_pv::signal_state_slave_base< STATE >.

7.86 tlmx::tlmx_blocking_snoop_if< TRANS > Class Template Reference

TLMX blocking snoop transaction interface.
`#include <tlmx/tlmx_bw_ifs.h>`
Inherits sc_core::sc_interface.

Public Member Functions

- virtual void b_snoop (TRANS &trans, sc_core::sc_time &t)=0

  Blocking snoop.

7.86.1 Detailed Description

template<typename TRANS = tlm::tlm_generic_payload>
class tlmx::tlmx_blocking_snoop_if< TRANS >

TLMX blocking snoop transaction interface.
This interface is used for the backward path.

Parameters

TRANS	transaction type; defaults to `tlm::tlm_generic_payload`.

7.86.2 Member Function Documentation

7.86.2.1 b_snoop()

```
template<typename TRANS = tlm::tlm_generic_payload>
virtual void tlmx::tlmx_blocking_snoop_if< TRANS >::b_snoop (
            TRANS & trans,
            sc_core::sc_time & t )  [pure virtual]
```

Blocking snoop.

Parameters

trans	transaction.
t	timing annotation.

7.87 tlmx::tlmx_bw_transport_if< TYPES > Class Template Reference

TLMX combined backward interface.
`#include <tlmx/tlmx_bw_ifs.h>`
Inherits tlm::tlm_bw_nonblocking_transport_if< TYPES::tlm_payload_type, TYPES::tlm_phase_type >, tlmx::tlmx_blocking_snoop_if
tlm::tlm_bw_direct_mem_if, and tlmx::tlmx_snoop_dbg_if< TYPES::tlm_payload_type >.

Additional Inherited Members

7.87.1 Detailed Description

template<typename TYPES = tlm::tlm_base_protocol_types>
class tlmx::tlmx_bw_transport_if< TYPES >

TLMX combined backward interface.

Parameters

TYPES	protocol traits class; defaults to `tlm::tlm_base_protocol_types`.

7.88 tlmx::tlmx_has_get_protocol_types< BASE, TYPES, typename > Struct Template Reference

Wrapper around tlm_base_(intiator|target)_socket.
`#include <tlmx/tlmx_has_get_protocol_types.h>`
Inherits BASE.

7.88.1 Detailed Description

template<typename BASE, typename TYPES, typename = void>
struct tlmx::tlmx_has_get_protocol_types< BASE, TYPES, typename >

Wrapper around tlm_base_(intiator|target)_socket.

Note

`tlmx_has_get_protocol_types` is a wrapper around BASE type. If BASE type doesn't declare get↩ _protocol_types as a virtual member function, tlmx_has_get_protocol_types is a mere wrapper of BASE. If BASE type declares get_protocol_types as a virtual member, tlmx_has_get_protocol_types implements the (pure) virtual function. There are base classes which inherit from tlm_base_socket_if class in TLM. tlm↩ _base_socket_if declares tlmx_has_get_protocol_types as a pure virtual in Accelera systemc version 2.3.2 and above but doesn't declare this function in earlier versions and also not found in the SystemC IEEE 1666-2011 standard. The following SFINAE based approach helps to deal with this in a systemc implementation agnostic way.

Parameters

BASE	could be `tlm::tlm_base_initiator_socket` or `tlm::tlm_base_target_socket`
TYPES	protocol traits class; defaults to `tlm::tlm_base_protocol_types`.

7.89 tlmx::tlmx_initiator_socket< BUSWIDTH, TYPES, N, POL > Class Template Reference

TLMX initiator socket.
`#include <tlmx/tlmx_initiator_socket.h>`
Inherits tlmx::tlmx_has_get_protocol_types< tlm::tlm_base_initiator_socket< 32, tlm::tlm_fw_transport_if< tlm::tlm_base_protocol_t

Public Member Functions

- tlmx_initiator_socket ()

  Default constructor.

- tlmx_initiator_socket (const char *name)

  Constructor.
- virtual const char * kind () const

  Returns the kind string of this socket.

7.89.1 Detailed Description

template<unsigned int BUSWIDTH = 32, typename TYPES = tlm::tlm_base_protocol_types, int N = 1, sc_core::sc_port_policy POL = sc_core::SC_ONE_OR_MORE_BOUND>
class tlmx::tlmx_initiator_socket< BUSWIDTH, TYPES, N, POL >

TLMX initiator socket.
Principal initiator full-duplex socket, parameterized with protocol traits class. This full-duplex socket is for use as an initiator socket bound to one or more target full-duplex sockets.

Note

The current implementation of tlmx_initiator_socket inherits from TLM 2.0 `tlm::tlm_base_initiator_socket`.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024; defaults to 64.
TYPES	protocol traits class; defaults to `tlm::tlm_base_protocol_types`.
N	number of bindings. Defaults to 1.
POL	port binding policy. Defaults to `sc_core::SC_ONE_OR_MORE_BOUND`.

7.89.2 Constructor & Destructor Documentation

7.89.2.1 tlmx_initiator_socket() [1/2]

```
template<unsigned int BUSWIDTH, typename TYPES , int N, sc_core::sc_port_policy POL>
tlmx::tlmx_initiator_socket< BUSWIDTH, TYPES, N, POL >::tlmx_initiator_socket  [inline]
```

Default constructor.

7.89.2.2 tlmx_initiator_socket() [2/2]

```
template<unsigned int BUSWIDTH, typename TYPES , int N, sc_core::sc_port_policy POL>
tlmx::tlmx_initiator_socket< BUSWIDTH, TYPES, N, POL >::tlmx_initiator_socket (
            const char * name )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.

7.89.3 Member Function Documentation

7.89.3.1 kind()

```
template<unsigned int BUSWIDTH, typename TYPES , int N, sc_core::sc_port_policy POL>
const char * tlmx::tlmx_initiator_socket< BUSWIDTH, TYPES, N, POL >::kind  [inline], [virtual]
```

Returns the kind string of this socket.
Reimplemented in amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >, amba_pv::ext::amba_pv_ace_base_
and amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL >.

7.90 tlmx::tlmx_snoop_dbg_if< TRANS > Class Template Reference

TLMX snoop debug transport interface.
`#include <tlmx/tlmx_bw_ifs.h>`
Inherits sc_core::sc_interface.

Public Member Functions

- virtual unsigned int snoop_dbg (TRANS &trans)=0

  Debug access to an initiator.

7.90.1 Detailed Description

template<typename TRANS = tlm::tlm_generic_payload>
class tlmx::tlmx_snoop_dbg_if< TRANS >

TLMX snoop debug transport interface.
This interface is used for the backward path.

Parameters

TRANS	transaction type; defaults to `tlm::tlm_generic_payload`.

7.90.2 Member Function Documentation

7.90.2.1 snoop_dbg()

```
template<typename TRANS = tlm::tlm_generic_payload>
virtual unsigned int tlmx::tlmx_snoop_dbg_if< TRANS >::snoop_dbg (
            TRANS & trans )  [pure virtual]
```

Debug access to an initiator.
This use the same path as the b_snoop() interface. This debug access must be performed without any of the delays, waits, event notifications or side effects associated with a regular transaction. This debug access is, therefore, non-intrusive.

Parameters

trans	transaction.

Returns

number of bytes read or written or, if error, 0.

7.91 tlmx::tlmx_target_socket< BUSWIDTH, TYPES, N, POL > Class Template Reference

TLMX target socket.
`#include <tlmx/tlmx_target_socket.h>`
Inherits tlmx::tlmx_has_get_protocol_types< tlm::tlm_base_target_socket< 32, tlm::tlm_fw_transport_if< tlm::tlm_base_protocol_typ

Public Member Functions

- tlmx_target_socket ()
  Default constructor.
- tlmx_target_socket (const char *name)
  Constructor.
- virtual const char * kind () const
  Returns the kind string of this socket.

7.91.1 Detailed Description

template<unsigned int BUSWIDTH = 32, typename TYPES = tlm::tlm_base_protocol_types, int N = 1, sc_core::sc_port_policy POL = sc_core::SC_ONE_OR_MORE_BOUND>
class tlmx::tlmx_target_socket< BUSWIDTH, TYPES, N, POL >

TLMX target socket.
Principal target full-duplex socket, parameterized with protocol traits class. This full-duplex socket is for use as a target socket bound to one or more initiator full-duplex sockets.

Note

The current implementation of tlmx_target_socket inherits from TLM 2.0 `tlm::tlm_base_target_socket`.

Parameters

BUSWIDTH	bus width in bits as one of 8, 16, 32, 64, 128, 256, 512, or 1024; defaults to 64.
TYPES	protocol traits class; defaults to `tlm::tlm_base_protocol_types`.
N	number of bindings; defaults to 1.
POL	port binding policy; defaults to `sc_core::SC_ONE_OR_MORE_BOUND`.

7.91.2 Constructor & Destructor Documentation

7.91.2.1 tlmx_target_socket() [1/2]

```
template<unsigned int BUSWIDTH, typename TYPES , int N, sc_core::sc_port_policy POL>
tlmx::tlmx_target_socket< BUSWIDTH, TYPES, N, POL >::tlmx_target_socket  [inline]
```
Default constructor.

7.91.2.2 tlmx_target_socket() [2/2]

```
template<unsigned int BUSWIDTH, typename TYPES , int N, sc_core::sc_port_policy POL>
```

```
tlmx::tlmx_target_socket< BUSWIDTH, TYPES, N, POL >::tlmx_target_socket (
            const char * name )  [inline], [explicit]
```

Constructor.

Parameters

name	socket name.

7.91.3 Member Function Documentation

7.91.3.1 kind()

```
template<unsigned int BUSWIDTH, typename TYPES , int N, sc_core::sc_port_policy POL>
const char * tlmx::tlmx_target_socket< BUSWIDTH, TYPES, N, POL >::kind  [inline], [virtual]
```

Returns the kind string of this socket.

Reimplemented in amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >, amba_pv::ext::amba_pv_ace_base_s and amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL >.

7.92 amba_pv::amba_pv_trans_pool::transaction_allocator Class Reference

AMBA-PV transaction allocator.

```
#include <user/amba_pv_mm.h>
```

Public Types

- typedef amba_pv_transaction * pointer

  Convenience definitions of pointers and references to AMBA-PV transaction.

Public Member Functions

- virtual ~transaction_allocator ()

  Destructor.
- virtual pointer allocate (amba_pv_trans_pool *pool)

  Factory method.
- virtual pointer allocate (amba_pv_trans_pool *pool, unsigned int length, unsigned int size, const amba_pv_control *ctrl, amba_pv_burst_t burst)

  Factory method.
- virtual void deallocate (pointer trans)

  Deallocates an AMBA-PV transaction.

7.92.1 Detailed Description

AMBA-PV transaction allocator.

This class implements the default allocator for the transactions in the pool. Derived classes could implement their own memory management while the transactions are still handled by the pool.

7.92.2 Member Typedef Documentation

7.92.2.1 pointer

```
typedef amba_pv_transaction* amba_pv::amba_pv_trans_pool::transaction_allocator::pointer
```

Convenience definitions of pointers and references to AMBA-PV transaction.

7.92.3 Constructor & Destructor Documentation

7.92.3.1 ~transaction_allocator()

```
virtual amba_pv::amba_pv_trans_pool::transaction_allocator::~transaction_allocator ( )  [inline],
[virtual]
```

Destructor.

Forces derived classes to have a virtual destructor for safety.

7.92.4 Member Function Documentation

7.92.4.1 allocate() [1/2]

```
virtual pointer amba_pv::amba_pv_trans_pool::transaction_allocator::allocate (
            amba_pv_trans_pool * pool )  [inline], [virtual]
```

Factory method.

Allocates and returns a new AMBA-PV transaction.

Parameters

pool	pointer to pool owning new transaction.

7.92.4.2 allocate() [2/2]

```
virtual pointer amba_pv::amba_pv_trans_pool::transaction_allocator::allocate (
            amba_pv_trans_pool * pool,
            unsigned int length,
            unsigned int size,
            const amba_pv_control * ctrl,
            amba_pv_burst_t burst )  [inline], [virtual]
```

Factory method.

Allocates and returns a new AMBA-PV transaction. It will also allocate a new associated AMBA-PV extension.

Parameters

pool	pointer to pool owning new transaction.
length	transaction burst length as in [1-256].
size	transaction burst size in bytes as one of [1, 2, 4, 8, 16, 32, 64, 128].
ctrl	optional AMBA 4 control information (set to `NULL` if unused).
burst	transaction burst type as one of `AMBA_PV_INCR`, `AMBA_PV_FIXED`, `AMBA_PV_WRAP`.

7.92.4.3 deallocate()

```
virtual void amba_pv::amba_pv_trans_pool::transaction_allocator::deallocate (
            pointer trans )  [inline], [virtual]
```

Deallocates an AMBA-PV transaction.

Parameters

trans	pointer to transaction to be deleted.

Chapter 8

File Documentation

8.1 amba_pv.h File Reference

AMBA-PV main header file.

8.1.1 Detailed Description

AMBA-PV main header file.

8.2 bus/amba_pv_atomic.h File Reference

AMBA-PV additional information for atomic transaction.

Data Structures

- class amba_pv::amba_pv_atomic

  Provides atomic transaction information used by AMBA AXI buses.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

Enumerations

- enum amba_pv::amba_pv_atomic_op_t {
  amba_pv::AMBA_PV_NONATOMIC ,
  amba_pv::AMBA_PV_ATOMICSTORE ,
  amba_pv::AMBA_PV_ATOMICLOAD ,
  amba_pv::AMBA_PV_ATOMICSWAP ,
  amba_pv::AMBA_PV_ATOMICCOMPARE }

  Atomic transaction type.
- enum amba_pv::amba_pv_atomic_subop_t {
  amba_pv::AMBA_PV_ATOMIC_ADD ,
  amba_pv::AMBA_PV_ATOMIC_BIT_CLEAR ,
  amba_pv::AMBA_PV_ATOMIC_EXCLUSIVE_OR ,
  amba_pv::AMBA_PV_ATOMIC_BIT_SET ,
  amba_pv::AMBA_PV_ATOMIC_SIGNED_MAX ,
  amba_pv::AMBA_PV_ATOMIC_SIGNED_MIN ,
  amba_pv::AMBA_PV_ATOMIC_UNSIGNED_MAX ,
  amba_pv::AMBA_PV_ATOMIC_UNSIGNED_MIN }

  Atomic transaction operation type.

- enum amba_pv::amba_pv_atomic_endianness_t {
  amba_pv::AMBA_PV_LITTLE_ENDIAN ,
  amba_pv::AMBA_PV_BIG_ENDIAN }

  Atomic operation endianness.

Functions

- std::string amba_pv::amba_pv_atomic_op_string (const amba_pv_atomic_op_t op)

  converts an atomic op enum to a human readable string

8.2.1 Detailed Description

AMBA-PV additional information for atomic transaction.

8.3 bus/amba_pv_atomic_subop_impl.h File Reference

AMBA-PV far atomics operation type implemention.

Data Structures

- struct amba_pv::atomic_subop_impl::do_add

  A functor for in-place addition operation.
- struct amba_pv::atomic_subop_impl::do_bit_clear

  A functor for in-place bit clear operation.
- struct amba_pv::atomic_subop_impl::do_xor

  A functor for in-place exclusive or operation.
- struct amba_pv::atomic_subop_impl::do_bit_set

  A functor for in-place bit set operation.
- struct amba_pv::atomic_subop_impl::do_signed_max

  A functor for in-place signed max operation.
- struct amba_pv::atomic_subop_impl::do_signed_min

  A functor for in-place signed min operation.
- struct amba_pv::atomic_subop_impl::do_unsigned_max

  A functor for in-place unsigned max operation.
- struct amba_pv::atomic_subop_impl::do_unsigned_min

  A functor for in-place unsigned min operation.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.
- namespace amba_pv::atomic_subop_impl

  AMBA-PV atomic operation type implementation namespace.

8.3.1 Detailed Description

AMBA-PV far atomics operation type implemention.

8.4 bus/amba_pv_atomic_utils.h File Reference

AMBA-PV far atomics utility functions.

Data Structures

- class amba_pv::amba_pv_atomic_utils

  An utility class that offers the implementation of executing an atomic transaction.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

Functions

- void amba_pv::swap_bytes (unsigned char *const data, const size_t size)

  Swaps the bytes on a block of memory based on a specified size.

8.4.1 Detailed Description

AMBA-PV far atomics utility functions.

8.5 bus/amba_pv_attributes.h File Reference

Additional user-defined attributes.

Data Structures

- class amba_pv::amba_pv_attributes

  Provides support for additional user-defined attributes.
- class amba_pv::amba_pv_attributes::const_attribute_ref

  A `const` reference to a specific attribute in a map of attributes that is not accessed until it is required.
- class amba_pv::amba_pv_attributes::attribute_ref

  A reference to a specific attribute in a map of attributes that is not accessed until it is required.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.5.1 Detailed Description

Additional user-defined attributes.

8.6 bus/amba_pv_control.h File Reference

AMBA-PV additional control information.

Data Structures

- class amba_pv::amba_pv_control

  Provides support for additional control information used by the AMBA buses.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

Enumerations

- enum amba_pv::amba_pv_snoop_t {
  amba_pv::AMBA_PV_READ_NO_SNOOP ,
  amba_pv::AMBA_PV_READ_ONCE ,
  amba_pv::AMBA_PV_READ_CLEAN ,
  amba_pv::AMBA_PV_READ_NOT_SHARED_DIRTY ,
  amba_pv::AMBA_PV_READ_SHARED ,
  amba_pv::AMBA_PV_READ_UNIQUE ,
  amba_pv::AMBA_PV_CLEAN_UNIQUE ,
  amba_pv::AMBA_PV_CLEAN_SHARED ,
  amba_pv::AMBA_PV_CLEAN_INVALID ,
  amba_pv::AMBA_PV_MAKE_UNIQUE ,
  amba_pv::AMBA_PV_MAKE_INVALID ,
  amba_pv::AMBA_PV_WRITE_NO_SNOOP ,
  amba_pv::AMBA_PV_WRITE_UNIQUE ,
  amba_pv::AMBA_PV_WRITE_LINE_UNIQUE ,
  amba_pv::AMBA_PV_WRITE_BACK ,
  amba_pv::AMBA_PV_WRITE_CLEAN ,
  amba_pv::AMBA_PV_EVICT ,
  amba_pv::AMBA_PV_BARRIER ,
  amba_pv::AMBA_PV_DVM_COMPLETE ,
  amba_pv::AMBA_PV_DVM_MESSAGE }

  Snoop type.
- enum amba_pv::amba_pv_domain_t {
  amba_pv::AMBA_PV_NON_SHAREABLE ,
  amba_pv::AMBA_PV_INNER_SHAREABLE ,
  amba_pv::AMBA_PV_OUTER_SHAREABLE ,
  amba_pv::AMBA_PV_SYSTEM }

  Domain type.
- enum amba_pv::amba_pv_bar_t {
  amba_pv::AMBA_PV_RESPECT_BARRIER ,
  amba_pv::AMBA_PV_MEMORY_BARRIER ,
  amba_pv::AMBA_PV_IGNORE_BARRIER ,
  amba_pv::AMBA_PV_SYNCHRONISATION_BARRIER }

  Barrier type.
- enum amba_pv::amba_pv_service_req_t {
  amba_pv::AMBA_PV_NO_SERVICE_REQUEST ,
  amba_pv::AMBA_PV_PCIE_SERVICE ,
  amba_pv::AMBA_PV_FAR_ATOMIC_SERVICE }

  Service Request type.
- enum amba_pv::amba_pv_physical_address_space_t {
  amba_pv::AMBA_PV_SECURE_PAS ,
  amba_pv::AMBA_PV_NON_SECURE_PAS ,
  amba_pv::AMBA_PV_ROOT_PAS ,
  amba_pv::AMBA_PV_REALM_PAS ,
  amba_pv::AMBA_PV_SYSTEM_AGENT_PAS ,
  amba_pv::AMBA_PV_NON_SECURE_PROTECTED_PAS ,
  amba_pv::AMBA_PV_NA6_PAS ,
  amba_pv::AMBA_PV_NA7_PAS }

  Physical Address Space type.
- enum amba_pv::amba_pv_mmuflow_t {
  amba_pv::AMBA_PV_MMUFLOW_STALL ,
  amba_pv::AMBA_PV_MMUFLOW_ATST ,
  amba_pv::AMBA_PV_MMUFLOW_NoStall ,
  amba_pv::AMBA_PV_MMUFLOW_PRI }

  AxMMUFLOW encodings.

Functions

- std::string amba_pv::amba_pv_snoop_read_string (amba_pv_snoop_t snoop, amba_pv_domain_t domain, amba_pv_bar_t bar)

  Returns the text string representation of the specified read snoop type.
- std::string amba_pv::amba_pv_snoop_write_string (amba_pv_snoop_t snoop, amba_pv_domain_t domain, amba_pv_bar_t bar)

  Returns the text string representation of the specified snoop type for write transactions.
- std::string amba_pv::amba_pv_domain_string (amba_pv_domain_t domain)

  Returns the text string representation of the specified domain type.
- std::string amba_pv::amba_pv_bar_string (amba_pv_bar_t bar)

  Returns the text string representation of the specified bar type.

8.6.1 Detailed Description

AMBA-PV additional control information.

8.7 bus/amba_pv_dvm.h File Reference

AMBA-PV additional information for DVM messages.

Data Structures

- class amba_pv::amba_pv_dvm

  Provides DVM message information used by the AMBA ACE buses.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

Enumerations

- enum amba_pv::amba_pv_dvm_message_t {
  amba_pv::AMBA_PV_TLB_INVALIDATE ,
  amba_pv::AMBA_PV_BRANCH_PREDICTOR_INVALIDATE ,
  amba_pv::AMBA_PV_PHYSICAL_INSTRUCTION_CACHE_INVALIDATE ,
  amba_pv::AMBA_PV_VIRTUAL_INSTRUCTION_CACHE_INVALIDATE ,
  amba_pv::AMBA_PV_SYNC ,
  amba_pv::AMBA_PV_HINT }

  DVM Message type.
- enum amba_pv::amba_pv_dvm_os_t {
  amba_pv::AMBA_PV_HYPERVISOR_OR_GUEST ,
  amba_pv::AMBA_PV_EL3 ,
  amba_pv::AMBA_PV_GUEST ,
  amba_pv::AMBA_PV_HYPERVISOR }

  DVM message Guest OS or hypervisor type.
- enum amba_pv::amba_pv_dvm_security_t {
  amba_pv::AMBA_PV_SECURE_AND_NON_SECURE ,
  amba_pv::AMBA_PV_SECURE_ONLY ,
  amba_pv::AMBA_PV_NON_SECURE_ONLY }

  DVM message security type.
- enum amba_pv::amba_pv_dvm_stage_t {
  amba_pv::AMBA_PV_DVM_V7 ,
  amba_pv::AMBA_PV_STAGE_1_ONLY ,
  amba_pv::AMBA_PV_STAGE_2_ONLY }

DVM message Staged Invalidation.

Functions

- std::string amba_pv::amba_pv_dvm_message_string (amba_pv_dvm_message_t message_type)

  Returns the text string representation of the specified DVM message type.
- std::string amba_pv::amba_pv_dvm_os_string (amba_pv_dvm_os_t os)

  Returns the text string representation of the specified DVM Guest OS or hypervisor type.
- std::string amba_pv::amba_pv_dvm_security_string (amba_pv_dvm_security_t security)

  Returns the text string representation of the specified DVM security type.

8.7.1 Detailed Description

AMBA-PV additional information for DVM messages.

8.8 bus/amba_pv_extension.h File Reference

AMBA-PV extension class.

Data Structures

- class amba_pv::amba_pv_extension

  AMBA-PV extension class.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

Enumerations

- enum amba_pv::amba_pv_burst_t {
  amba_pv::AMBA_PV_FIXED ,
  amba_pv::AMBA_PV_INCR ,
  amba_pv::AMBA_PV_WRAP }

  Burst type.

Functions

- std::string amba_pv::amba_pv_burst_string (amba_pv_burst_t burst)

  Returns the text string representation of the specified burst type.
- std::string amba_pv::amba_pv_snoop_string (amba_pv_snoop_t snoop)

  Returns the text string representation of the specified snoop type.
- sc_dt::uint64 amba_pv::amba_pv_address (const sc_dt::uint64 &addr, unsigned int length, unsigned int size, amba_pv_burst_t burst, unsigned int n)

  Computes the address of a transfer in a burst.

8.8.1 Detailed Description

AMBA-PV extension class.

8.9 bus/amba_pv_response.h File Reference

AMBA-PV response class.

Data Structures

- class amba_pv::amba_pv_response

  AMBA-PV response class.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

Enumerations

- enum amba_pv::amba_pv_resp_t {
  amba_pv::AMBA_PV_INCOMPLETE ,
  amba_pv::AMBA_PV_OKAY ,
  amba_pv::AMBA_PV_EXOKAY ,
  amba_pv::AMBA_PV_SLVERR ,
  amba_pv::AMBA_PV_DECERR }

  AMBA-PV response type.

Functions

- std::string amba_pv::amba_pv_resp_string (amba_pv_resp_t resp)

  Returns the text string representation of the specified AMBA-PV response.
- amba_pv_resp_t amba_pv::amba_pv_resp_from_tlm (tlm::tlm_response_status response_status, bool is_exclusive=false)

  Translates the specified TLM 2.0 response status value into an AMBA-PV response.
- tlm::tlm_response_status amba_pv::amba_pv_resp_to_tlm (amba_pv_resp_t resp, bool is_exclusive=false)

  Translates the specified AMBA-PV response value into a TLM 2.0 response status.

8.9.1 Detailed Description

AMBA-PV response class.

8.10 core/amba_pv_core_ifs.h File Reference

AMBA-PV core transaction interfaces.

Data Structures

- class amba_pv::amba_pv_fw_transport_if

  AMBA-PV core transaction interface.
- class amba_pv::amba_pv_bw_transport_if

  AMBA-PV core transaction interface.
- class amba_pv::amba_pv_bw_snoop_if

  AMBA-PV core additional transaction interface for ACE.
- class amba_pv::amba_pv_ace_bw_transport_if

  AMBA-PV core additional transaction interface for ACE.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.10.1 Detailed Description

AMBA-PV core transaction interfaces.

8.11 core/amba_pv_ext_core_ifs.h File Reference

AMBA-PV core transaction interfaces.

Data Structures

- class amba_pv::ext::amba_pv_fw_transport_if

  AMBA-PV core transaction interface.
- class amba_pv::ext::amba_pv_bw_transport_if

  AMBA-PV core transaction interface.
- class amba_pv::ext::amba_pv_ace_bw_transport_if

  AMBA-PV ACE core transaction interface.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.
- namespace amba_pv::ext

  Extensions namespace.

8.11.1 Detailed Description

AMBA-PV core transaction interfaces.

8.12 core/amba_pv_types.h File Reference

AMBA-PV protocol types.

Data Structures

- struct amba_pv::amba_pv_protocol_types

  AMBA-PV protocol types.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

Typedefs

- typedef amba_pv_protocol_types::tlm_payload_type amba_pv::amba_pv_transaction

  AMBA-PV transaction type.

8.12.1 Detailed Description

AMBA-PV protocol types.

8.13 models/amba_pv_ace_protocol_checker.h File Reference

AMBA-PV protocol checker model.

Data Structures

- class amba_pv::amba_pv_ace_protocol_checker< BUSWIDTH >
  AMBA-PV ACE protocol checker model.

Namespaces

- namespace amba_pv
  AMBA-PV namespace.

8.13.1 Detailed Description

AMBA-PV protocol checker model.

8.14 models/amba_pv_ace_simple_probe.h File Reference

AMBA-PV ACE simple probe model.

Data Structures

- class amba_pv::amba_pv_ace_simple_probe< BUSWIDTH >
  AMBA-PV ACE simple probe model.

Namespaces

- namespace amba_pv
  AMBA-PV namespace.

8.14.1 Detailed Description

AMBA-PV ACE simple probe model.

8.15 models/amba_pv_address_map.h File Reference

AMBA-PV address mapping information related structures.

Data Structures

- class amba_pv::amba_pv_address_region
  AMBA-PV address region structure.
- class amba_pv::amba_pv_address_map
  AMBA-PV address mapping information structure.

Namespaces

- namespace amba_pv
  AMBA-PV namespace.

8.15.1 Detailed Description

AMBA-PV address mapping information related structures.

8.16 models/amba_pv_bridges.h File Reference

TLM 2.0 BP - AMBA-PV bridge modules.

Data Structures

- class amba_pv::amba_pv_from_tlm_bridge< BUSWIDTH >
  TLM 2.0 BP to AMBA-PV bridge module.
- class amba_pv::amba_pv_to_tlm_bridge< BUSWIDTH >
  AMBA-PV to TLM 2.0 BP bridge module.

Namespaces

- namespace amba_pv
  AMBA-PV namespace.

8.16.1 Detailed Description

TLM 2.0 BP - AMBA-PV bridge modules.

8.17 models/amba_pv_decoder.h File Reference

AMBA-PV bus decoder model.

Data Structures

- class amba_pv::amba_pv_decoder< BUSWIDTH, NUMMASTERS, NUMSLAVES >
  AMBA-PV bus decoder model.

Namespaces

- namespace amba_pv
  AMBA-PV namespace.

8.17.1 Detailed Description

AMBA-PV bus decoder model.

8.18 models/amba_pv_exclusive_monitor.h File Reference

AMBA-PV exclusive monitor model.

Data Structures

- class amba_pv::amba_pv_exclusive_monitor< BUSWIDTH >
  AMBA-PV exclusive monitor model.

Namespaces

- namespace amba_pv
  AMBA-PV namespace.

8.18.1 Detailed Description

AMBA-PV exclusive monitor model.

8.19 models/amba_pv_heap_allocator.h File Reference

heap memory allocator.

Data Structures

- class amba_pv::amba_pv_heap_allocator

  AMBA-PV heap memory allocator.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.19.1 Detailed Description

heap memory allocator.

8.20 models/amba_pv_memory.h File Reference

AMBA-PV advanced memory model.

Data Structures

- class amba_pv::amba_pv_memory< BUSWIDTH, ALLOCATOR >

  AMBA-PV advanced memory model.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.20.1 Detailed Description

AMBA-PV advanced memory model.

8.21 models/amba_pv_memory_base.h File Reference

AMBA-PV memory model base class.

Data Structures

- class amba_pv::amba_pv_memory_base< BUSWIDTH >

  AMBA-PV memory model base class.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.21.1 Detailed Description

AMBA-PV memory model base class.

8.22 models/amba_pv_protocol_checker.h File Reference

AMBA-PV protocol checker model.

Data Structures

- class amba_pv::amba_pv_protocol_checker< BUSWIDTH >

  AMBA-PV protocol checker model.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.22.1 Detailed Description

AMBA-PV protocol checker model.

8.23 models/amba_pv_protocol_checker_base.h File Reference

AMBA-PV protocol checker base model.

Data Structures

- class amba_pv::amba_pv_protocol_checker_base< BUSWIDTH >

  AMBA-PV protocol checker base model.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

Enumerations

- enum amba_pv::amba_pv_protocol_t {
  amba_pv::AMBA_PV_APB ,
  amba_pv::AMBA_PV_AHB ,
  amba_pv::AMBA_PV_AXI ,
  amba_pv::AMBA_PV_AXI3 ,
  amba_pv::AMBA_PV_AXI4_LITE ,
  amba_pv::AMBA_PV_AXI4 ,
  amba_pv::AMBA_PV_ACE_LITE ,
  amba_pv::AMBA_PV_ACE ,
  amba_pv::AMBA_PV_AXI5 }

  AMBA procotol checks type.

8.23.1 Detailed Description

AMBA-PV protocol checker base model.

8.24 models/amba_pv_simple_memory.h File Reference

AMBA-PV simple memory model.

Data Structures

- class amba_pv::amba_pv_simple_memory< BUSWIDTH >

  AMBA-PV simple memory model.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.24.1 Detailed Description

AMBA-PV simple memory model.

8.25 models/amba_pv_simple_probe.h File Reference

AMBA-PV simple probe model.

Data Structures

- class amba_pv::amba_pv_simple_probe< BUSWIDTH >

  AMBA-PV simple probe model.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.25.1 Detailed Description

AMBA-PV simple probe model.

8.26 models/amba_pv_simple_probe_base.h File Reference

AMBA-PV simple probe base model.

Data Structures

- class amba_pv::amba_pv_simple_probe_base< BUSWIDTH >

  AMBA-PV simple probe base model.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.26.1 Detailed Description

AMBA-PV simple probe base model.

8.27 signal/signal_bridges.h File Reference

Generic sc_signal - Signal(State) bridge modules.

Data Structures

- class amba_pv::signal_from_sc_bridge< STATE >

  Generic sc_signal to Signal bridge module.
- class amba_pv::signal_to_sc_bridge< STATE >

  Generic Signal to sc_signal bridge module.
- class amba_pv::signal_state_from_sc_bridge< STATE >

  Generic sc_signal to SignalState bridge module.
- class amba_pv::signal_state_to_sc_bridge< STATE >

  Generic SignalState to sc_signal bridge module.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.27.1 Detailed Description

Generic sc_signal - Signal(State) bridge modules.

8.28 signal/signal_core_ifs.h File Reference

Signal core interfaces.

Data Structures

- class amba_pv::nonblocking_transport_if< REQ, RSP >

  Non-blocking transport core interface.
- class amba_pv::signal_transport_if< STATE >

  Signal core interface.
- class amba_pv::signal_state_transport_if< STATE >

  SignalState core interface.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.28.1 Detailed Description

Signal core interfaces.

8.29 signal/signal_if.h File Reference

Signal and SignalState interfaces.

Data Structures

- class amba_pv::signal_if< STATE >

  Signal interface.
- class amba_pv::signal_state_if< STATE >

  SignalState interface.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.29.1 Detailed Description

Signal and SignalState interfaces.

8.30 signal/signal_master_port.h File Reference

Signal and SignalState ports to be instantiated on the master side.

Data Structures

- class amba_pv::signal_master_port< STATE, N, POL >

  Signal port to be instantiated on the master side.
- class amba_pv::signal_state_master_port< STATE, N, POL >

  SignalState port to be instantiated on the master side.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.30.1 Detailed Description

Signal and SignalState ports to be instantiated on the master side.

8.31 signal/signal_request.h File Reference

Signal request type.

Data Structures

- class amba_pv::signal_request< STATE >

  Signal request type.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

Enumerations

- enum amba_pv::signal_command {
  amba_pv::SIGNAL_SET ,
  amba_pv::SIGNAL_GET }

  Signal request command type.

8.31.1 Detailed Description

Signal request type.

8.32 signal/signal_response.h File Reference

Signal response type.

Data Structures

- class amba_pv::signal_response< STATE >
  Signal response type.

Namespaces

- namespace amba_pv
  AMBA-PV namespace.

8.32.1 Detailed Description

Signal response type.

8.33 signal/signal_slave_base.h File Reference

Base class for all Signal and SignalState slave modules.

Data Structures

- class amba_pv::signal_slave_base< STATE >
  Base class for all Signal slave modules.
- class amba_pv::signal_state_slave_base< STATE >
  Base class for all SignalState slave modules.

Namespaces

- namespace amba_pv
  AMBA-PV namespace.

8.33.1 Detailed Description

Base class for all Signal and SignalState slave modules.

8.34 signal/signal_slave_export.h File Reference

Signal and SignalState exports to be instantiated on the slave side.

Data Structures

- class amba_pv::signal_export_base
  Signal export base class.
- class amba_pv::signal_slave_export< STATE >
  Signal export to be instantiated on the slave side.
- class amba_pv::signal_state_slave_export< STATE >
  SignalState export to be instantiated on the slave side.

Namespaces

- namespace amba_pv
  AMBA-PV namespace.

8.34.1 Detailed Description

Signal and SignalState exports to be instantiated on the slave side.

8.35 sockets/amba_pv_ace_master_socket.h File Reference

AMBA-PV ACE socket to be instantiated on the master side.

Data Structures

- class amba_pv::amba_pv_ace_master_socket< BUSWIDTH >
  AMBA-PV ACE socket to be instantiated on the master side.

Namespaces

- namespace amba_pv
  AMBA-PV namespace.

8.35.1 Detailed Description

AMBA-PV ACE socket to be instantiated on the master side.

8.36 sockets/amba_pv_ace_slave_socket.h File Reference

AMBA-PV ACE socket to be instantiated on the slave side.

Data Structures

- class amba_pv::amba_pv_ace_slave_socket< BUSWIDTH >
  AMBA-PV ACE socket to be instantiated on the slave side.

Namespaces

- namespace amba_pv
  AMBA-PV namespace.

8.36.1 Detailed Description

AMBA-PV ACE socket to be instantiated on the slave side.

8.37 sockets/amba_pv_ext_ace_master_socket.h File Reference

AMBA-PV ACE socket to be instantiated on the master side.

Data Structures

- class amba_pv::ext::amba_pv_ace_base_master_socket< BUSWIDTH, N, POL >
  AMBA-PV ACE base master socket.
- class amba_pv::ext::amba_pv_ace_master_socket< BUSWIDTH, N, POL >
  AMBA-PV ACE socket to be instantiated on the master side.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.
- namespace amba_pv::ext

  Extensions namespace.

8.37.1 Detailed Description

AMBA-PV ACE socket to be instantiated on the master side.

8.38 sockets/amba_pv_ext_ace_slave_socket.h File Reference

AMBA-PV ACE socket to be instantiated on the slave side.

Data Structures

- class amba_pv::ext::amba_pv_ace_base_slave_socket< BUSWIDTH, N, POL >

  AMBA-PV base slave socket.
- class amba_pv::ext::amba_pv_ace_slave_socket< BUSWIDTH, N, POL >

  AMBA-PV ACE socket to be instantiated on the slave side.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.
- namespace amba_pv::ext

  Extensions namespace.

8.38.1 Detailed Description

AMBA-PV ACE socket to be instantiated on the slave side.

8.39 sockets/amba_pv_ext_master_socket.h File Reference

AMBA-PV socket to be instantiated on the master side.

Data Structures

- class amba_pv::ext::amba_pv_base_master_socket< BUSWIDTH, N, POL >

  AMBA-PV base master socket.
- class amba_pv::ext::amba_pv_master_socket< BUSWIDTH, N, POL >

  AMBA-PV socket to be instantiated on the master side.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.
- namespace amba_pv::ext

  Extensions namespace.

8.39.1 Detailed Description

AMBA-PV socket to be instantiated on the master side.

8.40 sockets/amba_pv_ext_slave_socket.h File Reference

AMBA-PV socket to be instantiated on the slave side.

Data Structures

- class amba_pv::ext::amba_pv_base_slave_socket< BUSWIDTH, N, POL >

  AMBA-PV base slave socket.
- class amba_pv::ext::amba_pv_slave_socket< BUSWIDTH, N, POL >

  AMBA-PV socket to be instantiated on the slave side.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.
- namespace amba_pv::ext

  Extensions namespace.

8.40.1 Detailed Description

AMBA-PV socket to be instantiated on the slave side.

8.41 sockets/amba_pv_master_socket.h File Reference

AMBA-PV socket to be instantiated on the master side.

Data Structures

- class amba_pv::amba_pv_master_socket< BUSWIDTH >

  AMBA-PV socket to be instantiated on the master side.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.41.1 Detailed Description

AMBA-PV socket to be instantiated on the master side.

8.42 sockets/amba_pv_slave_socket.h File Reference

AMBA-PV socket to be instantiated on the slave side.

Data Structures

- class amba_pv::amba_pv_slave_socket< BUSWIDTH >

  AMBA-PV socket to be instantiated on the slave side.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.42.1 Detailed Description

AMBA-PV socket to be instantiated on the slave side.

8.43 sockets/amba_pv_snoop_socket.h File Reference

AMBA-PV slave socket used to implement the upstream ACE snoop interface.

Data Structures

- class amba_pv::amba_pv_snoop_socket< BUSWIDTH >

  AMBA-PV slave socket used to implement the upstream ACE snoop interface.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.43.1 Detailed Description

AMBA-PV slave socket used to implement the upstream ACE snoop interface.

8.44 sockets/amba_pv_socket_array.h File Reference

AMBA-PV socket array class.

Data Structures

- class amba_pv::amba_pv_socket_array< SOCKET >

  AMBA-PV socket array class.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.44.1 Detailed Description

AMBA-PV socket array class.

8.45 sockets/amba_pv_socket_base.h File Reference

AMBA-PV socket base class.

Data Structures

- class amba_pv::amba_pv_socket_base

  AMBA-PV socket base class.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.45.1 Detailed Description

AMBA-PV socket base class.

8.46 tlmx/tlmx_bw_ifs.h File Reference

TLM 2.0 extended transaction interfaces.

Data Structures

- class tlmx::tlmx_blocking_snoop_if< TRANS >
  TLMX blocking snoop transaction interface.
- class tlmx::tlmx_snoop_dbg_if< TRANS >
  TLMX snoop debug transport interface.
- class tlmx::tlmx_bw_transport_if< TYPES >
  TLMX combined backward interface.

Namespaces

- namespace tlmx
  TLMX namespace.

8.46.1 Detailed Description

TLM 2.0 extended transaction interfaces.

8.47 tlmx/tlmx_has_get_protocol_types.h File Reference

A utility header to deal with non standard conformant Accelera specific socket interface - get_protocol_types.

Data Structures

- struct tlmx::tlmx_has_get_protocol_types< BASE, TYPES, typename >
  Wrapper around tlm_base_(intiator|target)_socket.

Namespaces

- namespace tlmx
  TLMX namespace.

8.47.1 Detailed Description

A utility header to deal with non standard conformant Accelera specific socket interface - get_protocol_types.

8.48 tlmx/tlmx_initiator_socket.h File Reference

TLM 2.0 extended initiator socket.

Data Structures

- class tlmx::tlmx_initiator_socket< BUSWIDTH, TYPES, N, POL >
  TLMX initiator socket.

Namespaces

- namespace tlmx

  TLMX namespace.

8.48.1 Detailed Description

TLM 2.0 extended initiator socket.

8.49 tlmx/tlmx_target_socket.h File Reference

TLM 2.0 extended target socket.

Data Structures

- class tlmx::tlmx_target_socket< BUSWIDTH, TYPES, N, POL >

  TLMX target socket.

Namespaces

- namespace tlmx

  TLMX namespace.

8.49.1 Detailed Description

TLM 2.0 extended target socket.

8.50 user/amba_pv_ace_master_base.h File Reference

Base class for all AMBA-PV ACE master modules.

Data Structures

- class amba_pv::amba_pv_ace_master_base

  Base class for all AMBA-PV ACE master modules.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.50.1 Detailed Description

Base class for all AMBA-PV ACE master modules.

8.51 user/amba_pv_ext_ace_master_base.h File Reference

Base class for all AMBA-PV ACE master modules.

Data Structures

- class amba_pv::ext::amba_pv_ace_master_base

  Base class for all AMBA-PV ACE master modules.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.
- namespace amba_pv::ext

  Extensions namespace.

8.51.1 Detailed Description

Base class for all AMBA-PV ACE master modules.

8.52 user/amba_pv_ext_ace_slave_base.h File Reference

Base class for all AMBA-PV ACE slave modules.

Data Structures

- class amba_pv::ext::amba_pv_ace_slave_base

  Base class for all AMBA-PV ACE slave modules.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.
- namespace amba_pv::ext

  Extensions namespace.

8.52.1 Detailed Description

Base class for all AMBA-PV ACE slave modules.

8.53 user/amba_pv_ext_master_base.h File Reference

Base class for all AMBA-PV master modules.

Data Structures

- class amba_pv::ext::amba_pv_master_base

  Base class for all AMBA-PV master modules.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.
- namespace amba_pv::ext

  Extensions namespace.

8.53.1 Detailed Description

Base class for all AMBA-PV master modules.

8.54 user/amba_pv_ext_slave_base.h File Reference

Base class for all AMBA-PV slave modules.

Data Structures

- class amba_pv::ext::amba_pv_slave_base< BUSWIDTH >

  Base class for all AMBA-PV slave modules.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.
- namespace amba_pv::ext

  Extensions namespace.

8.54.1 Detailed Description

Base class for all AMBA-PV slave modules.

8.55 user/amba_pv_if.h File Reference

AMBA-PV user-layer transaction interface.

Data Structures

- class amba_pv::amba_pv_if< BUSWIDTH >

  AMBA-PV user-layer transaction interface.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.55.1 Detailed Description

AMBA-PV user-layer transaction interface.

8.56 user/amba_pv_master_base.h File Reference

Base class for all AMBA-PV master modules.

Data Structures

- class amba_pv::amba_pv_master_base

  Base class for all AMBA-PV master modules.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.56.1 Detailed Description

Base class for all AMBA-PV master modules.

8.57 user/amba_pv_mm.h File Reference

AMBA-PV transaction memory manager classes.

Data Structures

- class amba_pv::amba_pv_trans_pool

  AMBA-PV transaction pool.
- class amba_pv::amba_pv_trans_pool::transaction_allocator

  AMBA-PV transaction allocator.
- class amba_pv::amba_pv_trans_ptr

  AMBA-PV transaction smart pointer.
- class amba_pv::amba_pv_trans_lock

  AMBA-PV transaction lock wrapper.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.57.1 Detailed Description

AMBA-PV transaction memory manager classes.

8.58 user/amba_pv_slave_base.h File Reference

Base class for all AMBA-PV slave modules.

Data Structures

- class amba_pv::amba_pv_slave_base< BUSWIDTH >

  Base class for all AMBA-PV slave modules.

Namespaces

- namespace amba_pv

  AMBA-PV namespace.

8.58.1 Detailed Description

Base class for all AMBA-PV slave modules.