PrimeCell® AXI Configurable Interconnect (PL300)

Revision: r0p1

Technical Reference Manual

ARM DDI 0354B

PrimeCell AXI Configurable Interconnect (PL300)
Technical Reference Manual

Release Information

The following changes have been made to this document.

Release history

Date	Issue	Confidentiality	Change
02 December 2004	A	Non-Confidential	First release for r0p0
26 October 2005	B	Confidential	Update for r0p1. Errata change to Figure 2-12 on page 2-19

Product Status

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

Web Address

`http://www.arm.com`

Contents

PrimeCell AXI Configurable Interconnect (PL300) Technical Reference Manual

List of Tables
PrimeCell AXI Configurable Interconnect (PL300) Technical Reference Manual

List of Figures
PrimeCell AXI Configurable Interconnect (PL300) Technical Reference Manual

Preface

This preface introduces the *PrimeCell AXI Configurable Interconnect (PL300) Technical Reference Manual*. It contains the following sections:

- *About this manual* on page x
- *Feedback* on page xv.

Note

You are strongly advised to read the AXI entries in the *Glossary* before reading this manual if you are not familiar with AXI interfaces and related terminology.

When referring to the size of an interconnect, this document always refers first to the number of slave interfaces, followed by the number of master interfaces. For example, a 3x4 interconnect interfaces to three masters and four slaves.

About this manual

This is the *Technical Reference Manual* for the *PrimeCell AXI Configurable Interconnect* (ACI).

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this manual, where:

r*n*	Identifies the major revision of the product.
p*n*	Identifies the minor revision or modification status of the product.

Intended audience

This manual is aimed at experienced hardware and software engineers who want to use the ACI in a *System-on-Chip* (SoC) design. Experience of working with ARM products or the ACI is not assumed.

Using this manual

This manual is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an overview of the ACI features and connections.

Chapter 2 *Functional Overview*

Read this chapter for a description of the major functional blocks of the ACI.

Appendix A *Signal Descriptions*

Read this appendix for a description of the ACI input and output signals.

Appendix B *Interface Attributes*

Read this appendix for a description of the ACI master and slave interface AXI attributes.

Glossary Read the Glossary for definitions of terms used in this manual.

Conventions

Conventions that this manual can use are described in:

- *Typographical*
- *Timing diagrams*
- *Signals* on page xii
- *Numbering* on page xiii.

Typographical

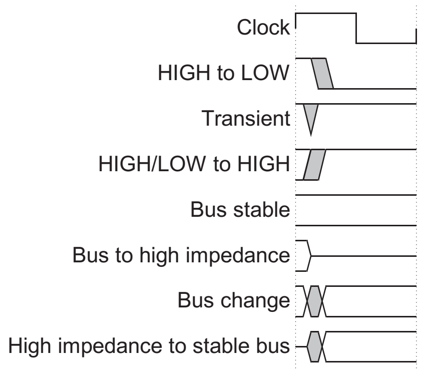
The typographical conventions are:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace` (`mono` underlined)	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
< and >	Angle brackets enclose replaceable terms for assembler syntax where they appear in code or code fragments. They appear in normal font in running text. For example: • `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>` • The Opcode_2 value selects which register is accessed.

Timing diagrams

The figure named *Key to timing diagram conventions* on page xii explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Key to timing diagram conventions

Signals

The signal conventions are:

Signal level	The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means HIGH for active-HIGH signals and LOW for active-LOW signals.
Lower-case n	Denotes an active-LOW signal.
Prefix A	Denotes global *Advanced eXtensible Interface* (AXI) signals:
Prefix AR	Denotes AXI read address channel signals.
Prefix AW	Denotes AXI write address channel signals.
Prefix B	Denotes AXI write response channel signals.
Prefix C	Denotes AXI low-power interface signals.
Prefix H	Denotes *Advanced High-performance Bus* (AHB) signals.
Prefix P	Denotes *Advanced Peripheral Bus* (APB) signals.
Prefix R	Denotes AXI read data channel signals.
Prefix W	Denotes AXI write data channel signals.

Numbering

The numbering convention is:

<size in bits>'<base><number>

This is a Verilog method of abbreviating constant numbers. For example:

- 'h7B4 is an unsized hexadecimal value.
- 'o7654 is an unsized octal value.
- 8'd9 is an eight-bit wide decimal value of 9.
- 8'h3F is an eight-bit wide hexadecimal value of `0x3F`. This is equivalent to b00111111.
- 8'b1111 is an eight-bit wide binary value of b00001111.

Further reading

This section lists publications by ARM Limited, and by third parties.

ARM Limited periodically provides updates and corrections to its documentation. See `http://www.arm.com` for current errata sheets, addenda, and the Frequently Asked Questions list.

ARM publications

This manual contains information that is specific to the ACI. See the following documents for other relevant information:

- *AMBA™ AXI Protocol v1.0 Specification* (ARM IHI 0022)
- *PrimeCell AXI Configurable Interconnect (PL300) Implementation Guide* (ARM DII 0121)
- *PrimeCell AXI Configurable Interconnect (PL300) Integration Manual* (ARM DII 0122)
- *PrimeCell Infrastructure AMBA 3 AXI File Reader Master (BP144) Technical Overview* (ARM DTO 0016)
- *PrimeCell Infrastructure AMBA 3 AXI Downsizer (BP131) Technical Overview* (ARM DTO 0018)
- *PrimeCell Infrastructure AMBA 3 AXI to AMBA 3 AHB Bridge (BP137) Technical Overview* (ARM DTO 0010)
- *PrimeCell Infrastructure AMBA 3 AXI to AMBA 3 APB Bridge (BP135) Technical Overview* (ARM DTO 0014)

- *PrimeCell Infrastructure AMBA 3 TrustZone Protection Controller (BP147) Technical Overview* (ARM DTO 0015).

Feedback

ARM Limited welcomes feedback on the PrimeCell ACI (PL300) and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier giving:

- the product name
- a concise explanation of your comments.

Feedback on this manual

If you have any comments on this manual, send email to `errata@arm.com` giving:

- the title
- the number
- the relevant page number(s) to which your comments apply
- a concise explanation of your comments.

ARM Limited also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the ACI. It contains the following sections:

- *About the ACI* on page 1-2
- *Features* on page 1-3
- *Product revisions* on page 1-4.

1.1 About the ACI

The ACI is a highly configurable RTL component that provides all of the infrastructure that you require to connect a number of AXI masters to a number of AXI slaves. This infrastructure is an integral part of an AXI-based system.

The block level RTL code is automatically configured from a system description file that is formatted in XML. A *File Reader Master* (FRM) based verification environment is configured at the same time that enables you to test the integrity and memory map of your specific configuration. See the *PrimeCell Infrastructure AMBA 3 AXI File Reader Master (BP144) Technical Overview* for more information.

The ACI enables rapid integration of an AXI-based system. It can be used in many different system configurations. Figure 1-1 shows the topology of the example interconnect, produced as part of the *Out Of Box* (OOB) procedures, connected to four AXI masters and five AXI slaves. The *PrimeCell AXI Configurable Interconnect (PL300) Implementation Guide* describes the OOB procedures.

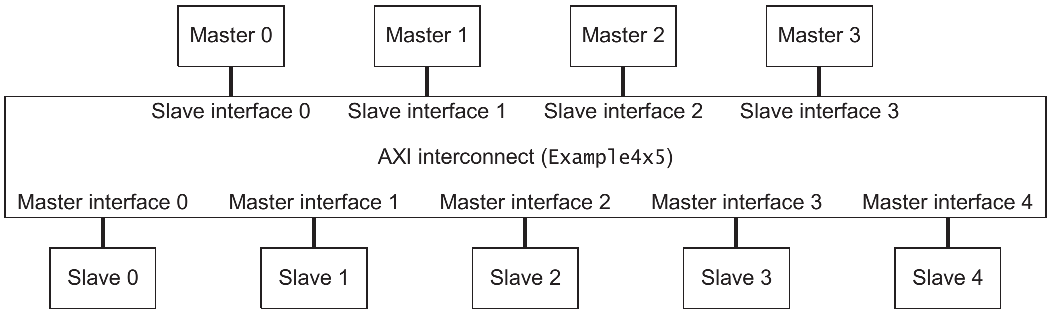


Figure 1-1 Example AXI interconnect

1.2 Features

The ACI features are:

- it is compliant with the *AMBA 3 AXI Protocol v1.0 Specification*
- it multiplexes and demultiplexes data and control information between connected masters and slaves
- it enforces the AXI ordering rules that govern the flow of data and control information on different channels
- it has a multi-layer capability to allow multiple masters to access different slaves simultaneously
- it produces a self-checking validation environment
- a global dynamic memory map
- deadlock avoidance
- TrustZone support
- out-of-order data support
- you can configure the following parameters:
  - number of master interfaces
  - names of master interfaces
  - number of slave interfaces
  - names of slave interfaces
  - the ID width of each slave interface
  - the write acceptance capability of each slave interface
  - the read acceptance capability of each slave interface
  - the cyclic dependency scheme for each slave interface
  - the write issuing capability of each master interface
  - the write interleave capability of each master interface
  - the memory map
  - the global data width of the interconnect
  - the global width of the transaction counters.

1.3 Product revisions

This section describes differences in functionality between product revisions of the ACI (PL300):

r0p0-r0p1 Contains the following differences in functionality:

- Now incorporates a cyclic dependency scheme that enables a master to have outstanding transactions to more than one slave. Each slave can be configured to one of three schemes, to optimize the ACI interconnect for the characteristics of the master.

Chapter 2
Functional Overview

This chapter describes the functions and operation of the ACI. It contains the following sections:

2.1 Functional description

The ACI configuration options and features are explained in more detail in the following sections:

- *Configurable interconnect options*
- *Native AXI support* on page 2-10
- *Multi-layer capability* on page 2-10
- *Fixed arbitration* on page 2-11
- *Combinatorial data paths* on page 2-11
- *Locked transfers* on page 2-11
- *TrustZone support* on page 2-12
- *TCL control script* on page 2-13.

2.1.1 Configurable interconnect options

You can configure the options described in:

- *Number of master and slave interfaces*
- *Data width*
- *Slave interface ID width* on page 2-3
- *Slave interface read and write acceptance capabilities* on page 2-4
- *Cyclic dependency schemes* on page 2-4
- *Outstanding write transactions* on page 2-7
- *Out-of-order data support* on page 2-7
- *Memory map* on page 2-10.

Number of master and slave interfaces

You can specify the number of master and slave interfaces that the interconnect is built with. There is no limit on the number of masters or slaves that the interconnect can support although you must be aware of the timing implications of a matrix with a large number of interfaces.

Data width

The ACI supports the AXI bus protocol with either 32-bit or 64-bit data paths. If you require a mix of data widths then you can integrate standard components such as the AXI Downsizer to the periphery of the interconnect to provide the necessary data width conversion. See the *PrimeCell Infrastructure AMBA 3 AXI Downsizer (BP131) Technical Overview* for more information.

Slave interface ID width

You can configure the ID width of each slave interface to a defined width. This enables you to connect masters with different ID widths. If the slave interface has an ID width of zero then the slave interface Verilog is configured to have zero ID ports.

The width of the master interface ID on the interconnect is *not* configurable. It is a function of the maximum slave interface ID width and the number of slave interfaces on the interconnect.

Figure 2-1 illustrates this with an example. It shows an interconnect with two masters:

- slave interface 0 has a configured ID width of two bits
- slave interface 1 has a configured ID width of four bits.

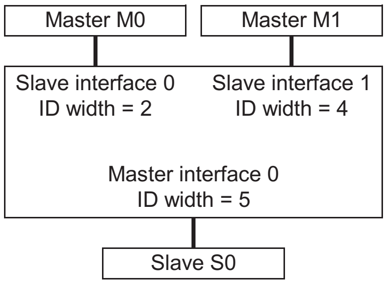


Figure 2-1 ID width

The master interface in Figure 2-1 has an ID width of five bits. This is derived automatically from the calculation:

(largest slave interface ID width) + (log$_2$(total number of slave interfaces)).

——— **Note** ———

You must configure your slaves to match the derived interconnect master interface ID width.

————————————

The interconnect appends bits to the ID to differentiate between the physical masters in the system. If necessary, the interconnect pads the incoming ID to equal the width of the master interface automatically.

For example, if master M1 in Figure 2-1 accesses slave S0 with an ID value of b1001 then one bit with a value of 1 is added to the slave interface ID least significant bit to indicate that this is from master M1. Therefore the slave receives a 5 bit ID with a value of b10011.

Another example is if master M0 in Figure 2-1 on page 2-3 accesses slave S0 with an ID value of b11 then one bit with a value of 0 is added to the slave interface ID least significant bit to indicate that this is from master M0. Two bits with a value of b00 are also added to the slave interface ID most significant bit to pad the ID width to 5 bits. Therefore the slave receives a 5 bit ID with a value of b00110.

Slave interface read and write acceptance capabilities

You can configure the maximum number of active read transactions a slave interface can accept. You are advised to set the read acceptance capability of the slave interface to match the read issuing capability of its attached master interface.

Similarly you can configure the maximum number of active write transactions a slave interface can accept. You are advised to set the write acceptance capability of the slave interface to match the write issuing capability of its attached master interface.

Cyclic dependency schemes

In any interconnect that is connected to a slave that reorders read data or write response signals, there is the potential for deadlock. To prevent this the ACI provides three cyclic dependency schemes that enables the slave interface to accept or stall a new transaction address. Each slave interface can be configured to one of the following cyclic dependency schemes:

- Single Slave
- Unique ID
- Hybrid

Each of these schemes are described in more detail in the following sections.

Single slave

In this configuration the ACI implements a deadlock prevention scheme that accepts or stalls a new transaction address based on the following rules:

- A master can initiate a transaction to any slave if the master has no outstanding transactions.
- If the master does have outstanding transactions then:
  - — a master can initiate a transaction to the same slave as the current outstanding transactions.

Figure 2-2 on page 2-5 shows these rules.

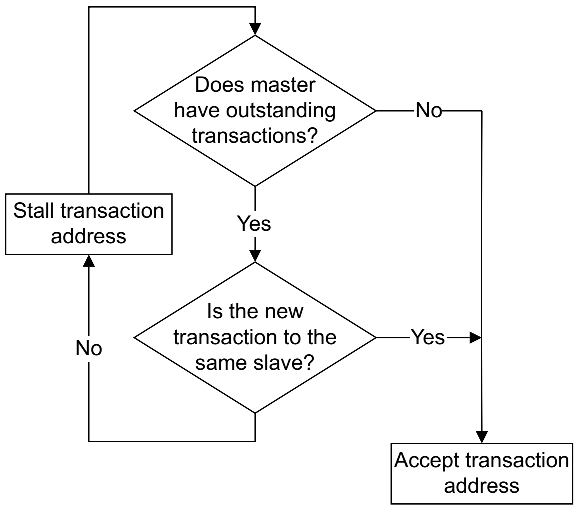

Does master have outstanding transactions?
No
Yes
Stall transaction address
Is the new transaction to the same slave?
No
Yes
Accept transaction address

Figure 2-2 Single slave scheme

This scheme adds minimal logic to the ACI interconnect, so this option has the least timing impact although it does not provide the flexibility of the two other schemes.

Unique ID

In this configuration the ACI implements a deadlock prevention scheme that accepts or stalls a new transaction address based on the following rules:

- A master can initiate a transaction to any slave if the master has no outstanding transactions.
- If the master does have outstanding transactions then:
  - a master can initiate a transaction to any slave but only if the transaction ID of the current transaction is unique, relative to current outstanding transactions.

    Note

    The unique ID scheme can only be used when the slave interface ID width is greater than `0`.

Figure 2-3 on page 2-6 shows these rules.

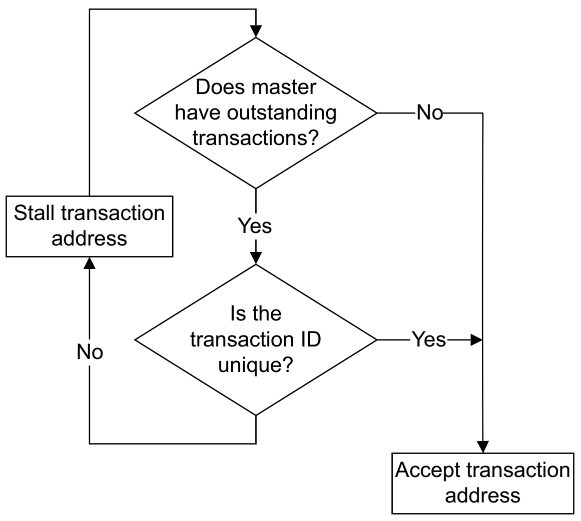

Does master have outstanding transactions?
No
Yes
Stall transaction address
Is the transaction ID unique?
No
Yes
Accept transaction address

Figure 2-3 Unique ID scheme

This scheme adds additional logic to the ACI interconnect, so this option provides a compromise between timing impact and transaction handling. The *PrimeCell AXI Configurable Interconnect (PL300) Implementation Guide* describes how to choose the appropriate cyclic dependency scheme.

Hybrid

In this configuration the ACI implements a deadlock prevention scheme that accepts or stalls a new transaction address based on the following rules:

- A master can initiate a transaction to any slave if the master has no outstanding transactions
- If the master does have outstanding transactions then:
  - a master can initiate a transaction to the same slave as the current outstanding transactions, or
  - a master can initiate a transaction to any slave but only if the transaction ID of the current transaction is unique, relative to current outstanding transactions.

  ——— **Note** ———

  The hybrid scheme can only be used when the slave interface ID width is greater than `0`.

  ———————————

Figure 2-4 shows these rules.

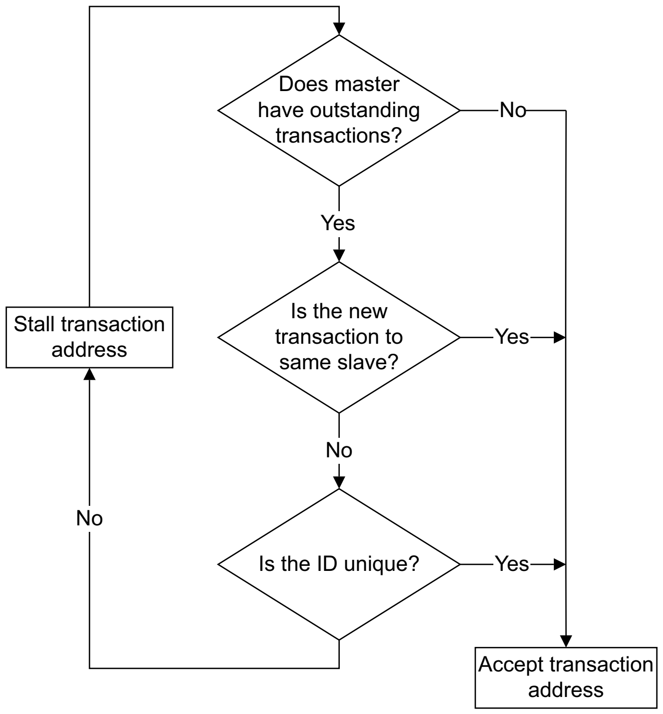

Does master have outstanding transactions?
No
Yes
Is the new transaction to same slave?
Yes
No
Is the ID unique?
Yes
No
Stall transaction address
Accept transaction address

Figure 2-4 Hybrid scheme

This scheme adds the greatest logic to the ACI interconnect, so this option has the most timing impact but it does provide the greatest flexibility in handling transactions compared with the two previous schemes.

Outstanding write transactions

You can configure the number of outstanding write addresses that each interconnect master interface is capable of issuing. You are advised to set the master interface write issuing capability to match the write acceptance capability of its attached slave.

Out-of-order data support

Out-of-order data is supported for read and write data as described in:

- *Write data* on page 2-8
- *Read data* on page 2-9.

Write data

Write data interleaving enables the interconnect to combine write data streams from different physical masters, to a single slave. This is useful because you can combine write data from a fast master with write data from a slow master and consequently increase the throughput of data across the interconnect.

The ACI supports write data interleaving and you can configure each master interface to support a slave with a different write interleave depth (see the *Glossary*). For slaves with an interleave depth of greater than one, such as the PrimeCell Dynamic Memory Controller (PL340), the interconnect can interleave write data from a number of different physical masters to maximize write data throughput and minimize latency for a given master.

——— **Note** ———

Interconnect slave interfaces have a fixed non-configurable write interleave depth of one. A connected master must not interleave write data to that interface and therefore its write interleaving capability must be equal to one.

You cannot interleave write data through cascaded interconnects. If you are using cascaded interconnects then you must ensure that the write interleaving capability of any master interface that connects to an interconnect slave interface is set to 1.

————————

As an example, Figure 2-5 shows an interconnect with two masters and one slave. The configured write interleave capability of master interface 0 is set at two to match the write interleave depth of slave 0.

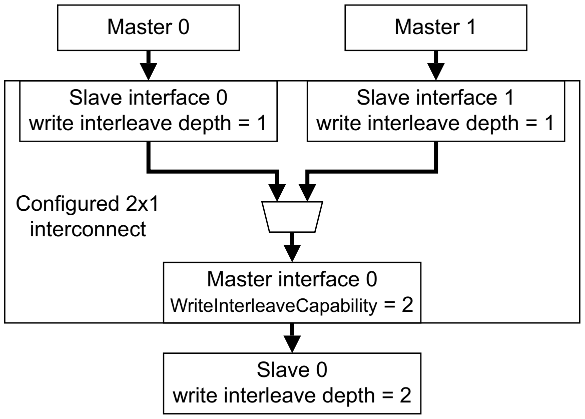

Master 0
Master 1
Slave interface 0
write interleave depth = 1
Slave interface 1
write interleave depth = 1
Configured 2x1
interconnect
Master interface 0
WriteInterleaveCapability = 2
Slave 0
write interleave depth = 2

Figure 2-5 Write data interleaving example

Assuming master 1 is capable of transmitting data faster than master 0, then Figure 2-6 shows the relative interleaved write transactions from master 0 and master 1 to slave 0.

In Figure 2-6:

WA	Signifies write address accepted.
WD	Signifies write data.
B	Signifies the write response.

Figure 2-6 shows that write data from master 1 is interleaved with write data from master 0. This decreases the time for both write transactions to complete.

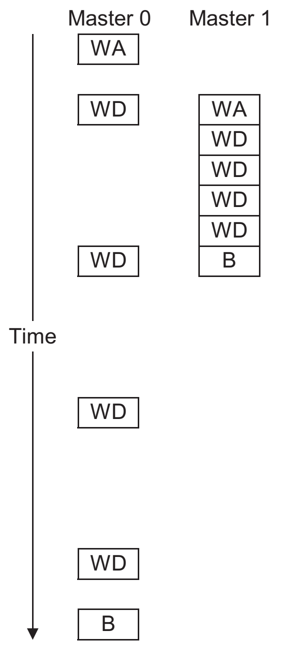


Figure 2-6 Write data interleaving

Read data

The ACI supports slaves that are capable of reordering read data. You do not have to configure the interconnect to support this feature.

A read transaction tracking scheme is implemented to ensure that each master is permitted to have outstanding read transactions to a single slave only. This ensures that deadlock does not occur in the interconnect when slaves reorder read data.

Memory map

The ACI allows you to define a memory map that assigns how the slaves are decoded. Each memory region is permitted to be non-contiguous.

You can use **REMAP** signals to switch dynamically between any number of pre-configured memory maps. See the *PrimeCell AXI Configurable Interconnect (PL300) Implementation Guide* for more information.

The interconnect provides a decode error response, DECERR, for any areas in the memory map that are not assigned to any slave.

An internal default slave is included in the design. When any slave attempts to access an area of memory that is undefined then the transfer is routed to the default slave. This accepts the transaction and returns a DECERR in either:

- **RRESP** for a read
- **BRESP** for a write. The entire **WDATA** burst is accepted normally for a write and it is only in the write response that the master is informed that there is an error.

Each address channel has a separate decoder so that read and write addresses can be accepted simultaneously. Each decoder uses the same memory map. The created HDL for the decoder block is partitioned into a separate file, `pl300_DecodeScheme_<name>.v`, to enable you to modify the RTL in the unlikely event that the XML memory map options do not meet your requirements. See the *PrimeCell AXI Configurable Interconnect (PL300) Implementation Guide* for more information and examples.

2.1.2 Native AXI support

The ACI is designed as a native AXI block. All bus routing and control logic supports the AXI protocol. If you require a mix of AMBA interfaces such as AHB or APB, then you can integrate the relevant bridges on the periphery of the interconnect to provide the necessary interface conversion. See the following for more information about AXI bridges:

- *PrimeCell Infrastructure AMBA 3 AXI to AMBA 3 AHB Bridge (BP137) Technical Overview*
- *PrimeCell Infrastructure AMBA 3 AXI to AMBA 3 APB Bridge (BP135) Technical Overview*.

2.1.3 Multi-layer capability

Figure 2-7 on page 2-11 shows masters and slaves attached to a multi-layer interconnect. The parallel capability of this interconnect enables master M1 to access one slave at the same as master M0 is accessing the other.

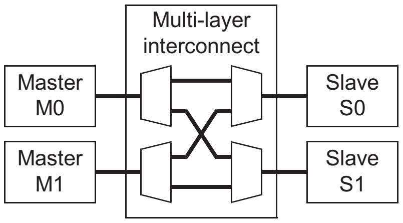


Figure 2-7 Multi-layer interconnect

The ACI supports a full multi-layer connection of all master and slave interfaces on all of the AXI channels.

Multi-layer interconnect enables parallel access paths between multiple masters and slaves in a system. This:

- increases data throughput
- decreases latency.

2.1.4 Fixed arbitration

The ACI supports a fixed address arbitration scheme. The priority is in the same order as the master interfaces are defined in the interconnect configuration script as described in the *PrimeCell AXI Configurable Interconnect (PL300) Implementation Guide*.

The scheme is partitioned as a separate module, `pl300_ArbiterScheme_<name>.v`, so that you can modify it if you require an alternative arbitration scheme.

A single arbiter arbitrates between the read and write address channels to minimize lock logic. This means that a slave can only accept simultaneous read and write transactions from the same master.

2.1.5 Combinatorial data paths

The ACI is designed with combinatorial data paths between the master and slave interfaces to minimize overall latency and provide you with more design flexibility. You can insert one or more register slices on the periphery of the interconnect for high frequency implementations. This breaks up the combinatorial paths at the expense of extra cycles of latency.

2.1.6 Locked transfers

Locked transfers are initiated by a master signaling with the **AWLOCK** or **ARLOCK** signal. To maintain data integrity throughout a locked sequence, the interconnect only accepts the locked transfer after the target slave has completed all outstanding transactions. The slave then becomes locked with only the locking master allowed access to it. The slave remains locked until the locking master completes an unlocked transfer to that slave.

Only one slave is locked in this implementation to enable masters to access other slaves during the locked period. This implies that a locked sequence must be within the same 4KB address region to guarantee that it is to a single slave. Locked sequences from ARM processors are always to a single address.

A locked transfer is only granted access by the arbiter when there are no outstanding transactions on the master interface. It is therefore advisable to connect masters to high priority slave interfaces if you intend them to perform locked transfers.

——— **Note** ———

To ensure that atomic accesses decode to the correct slave, you must make sure that the external values of **REMAP**, **TZPROT**, **AWPROT[1]**, and **ARPROT[1]** do not change during:

- a locked transfer sequence
- an exclusive access pair.

2.1.7 TrustZone support

TrustZone support enables the interconnect to reject, by signaling a DECERR response, non-secure transfers that access an area that has been assigned as secure. Each master interface is assigned a secure or non-secure attribute through its dedicated **TZPROT** inputs that can be tied off for a static configuration or driven by a security controller such as the *TrustZone Protection Controller* (TZPC) in a more dynamic system. See the *PrimeCell Infrastructure AMBA 3 TrustZone Protection Controller (BP147) Technical Overview* for more information.

Slaves that require both secure and non-secure areas within their own memory area must be labeled as non-secure and must implement TrustZone support locally.

——— **Caution** ———

You must use a secure software protocol before relying on any security settings that can be changed. This might include, but is not limited to:

- verifying that instructions to change the security settings are propagated across the interconnect to their final destination
- clearing any storage locations that can change the security status
- flushing caches and page tables

- stopping other masters.

2.1.8 TCL control script

The build process of the ACI is controlled by a TCL configuration script that enables you to:

- configure the source RTL
- configure the validation environment
- run the self-checking validation environment.

A unique directory is created each time the script is run and all automatically generated files are placed within this directory structure. See the *PrimeCell AXI Configurable Interconnect (PL300) Implementation Guide* for more information.

2.2 XML/XSL flow

An *eXtensible Mark-up Language* (XML) flow is at the core of the ACI. This enables you to define all of the configuration options and then build the required output files automatically. XML and its subset languages, *eXtensible Stylesheet Language* (XSL) and *eXtensible Schema Document* (XSD) are used because it is:

- the industry standard for data exchange
- compatible with other EDA tools
- flexible
- relatively simple to parse and process
- well supported by programmers and programming tools.

The flow consists of an XML parser that reads the input configuration and compares the XML input file against a *Document Type Definition* (DTD) to check that the configuration is legal.

An XSL processor then transforms the XML data into the format defined in the XSL stylesheet. This builds each of the Verilog logic and structural elements of the interconnect.

The XML parser and XSL processor are third party tools that are supplied with the ACI under a free software license. Figure 2-8 shows the flow.

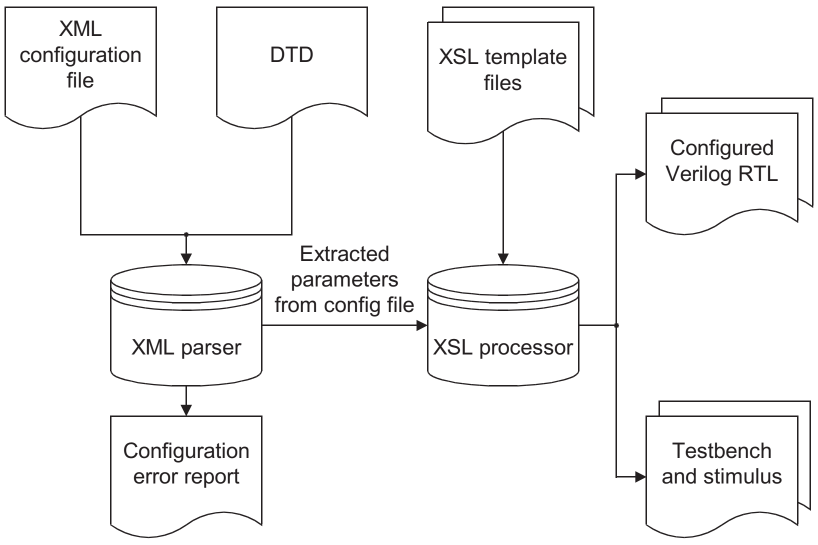


Figure 2-8 XML/XSL flow

2.3 Architecture

Figure 2-9 shows a simplified block diagram of a multi-layer interconnect.

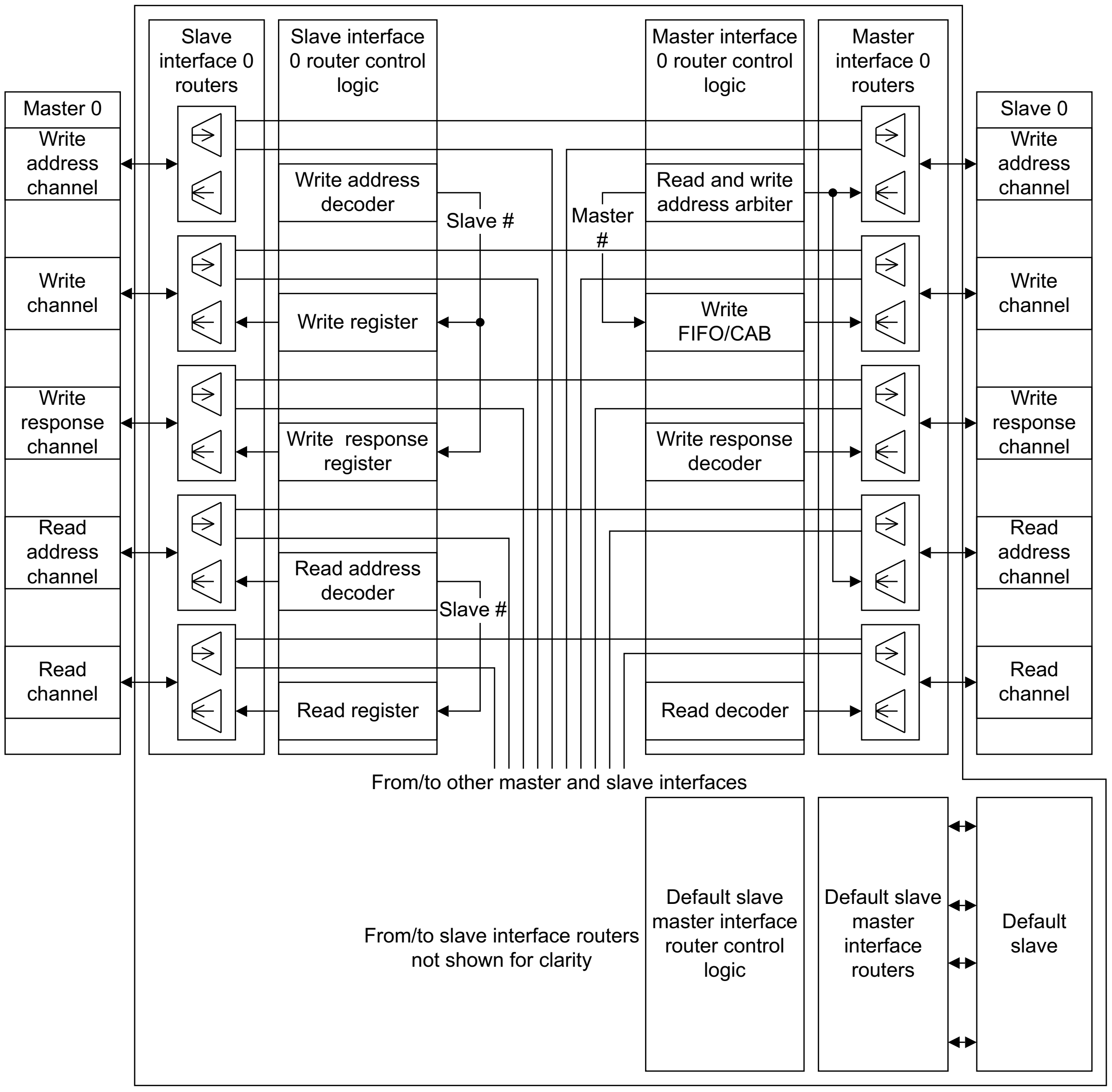


Figure 2-9 Simplified multi-layer interconnect architecture

The interconnect shown in Figure 2-9 on page 2-15 only shows a single master and single slave interface for clarity. Creating this interconnect configuration is inadvisable.

The architecture of the interconnect is highly modular with each of the routers and associated control logic partitioned on a per-channel basis.

The routers consist of multiplexing and demultiplexing elements that are based on the number of inputs and outputs. The select signal for each router is generated from a control block that is unique for each channel and interface. The control blocks store the routing information necessary to enforce the ordering constraints within the AXI protocol. They contain:

- arbiters
- decoders
- *Content Addressable Buffers* (CABs)
- FIFO elements.

Note

The interconnect does not buffer addresses or data. A slave that supports outstanding transactions must contain this storage locally.

2.4 Timing diagrams

This section provides timing diagrams in:

- *Address channels*
- *Write data channel* on page 2-18
- *Write response channel* on page 2-21
- *Read data channel* on page 2-22.

2.4.1 Address channels

The address channel timing diagrams are provided in:

- *Zero latency*
- *With latency* on page 2-18.

The diagrams provided in Figure 2-10 and Figure 2-11 on page 2-18 illustrate the timing for the write address channel. The timing for the read address channel is the same as that shown for these figures. The signal name prefixes change from **AW** to **AR**.

Zero latency

Figure 2-10 shows the write address channel for a transaction with zero latency. This zero latency is because the address channel arbiter has already granted the master.

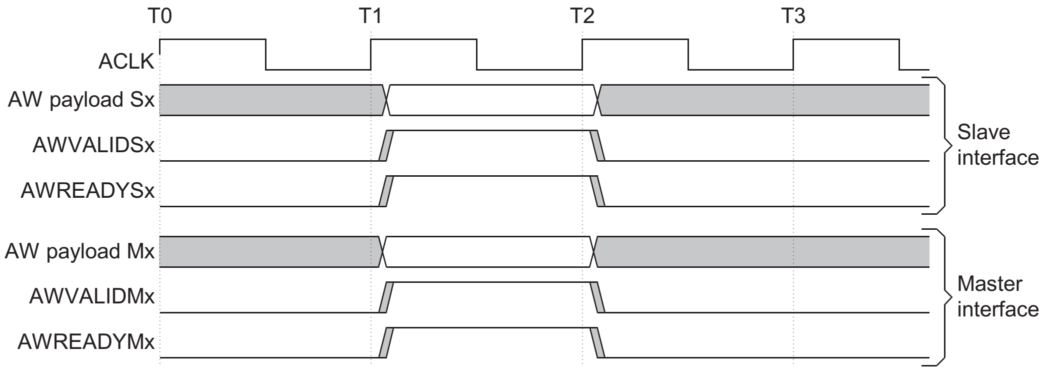


Figure 2-10 Address channel with zero latency

With latency

Figure 2-11 shows the write address channel for a transaction with latency. The one cycle latency at T1-T2 for **AWVALIDSx** and **AWVALIDMx** is caused by the address arbiter switching from the old master to the new one.

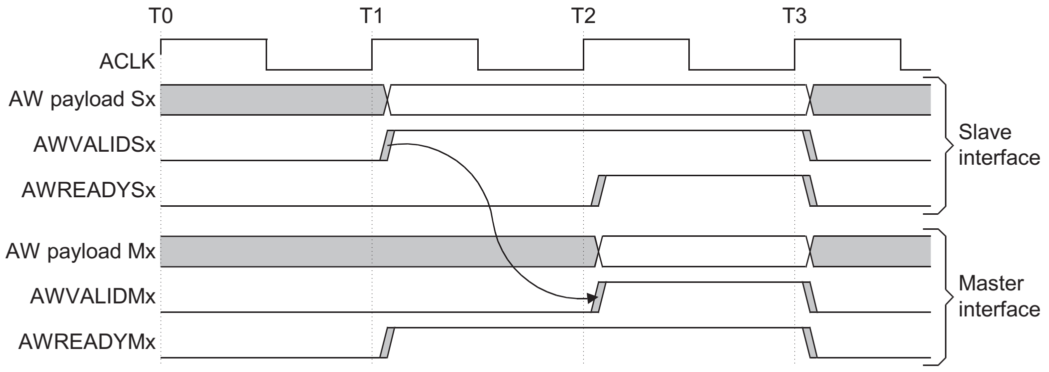


Figure 2-11 Address channel with latency

2.4.2 Write data channel

The write data channel timing diagrams are provided in:

- *Write interleave capability of one* on page 2-19
- *Write interleave capability greater than one* on page 2-20.

Write interleave capability of one

Figure 2-12 shows the write data channel to an interconnect master interface with a write interleave capability of one:

- the master sends the write address and data simultaneously at T1
- there is zero arbitration latency
- the slave has zero wait states
- the one cycle latency at T1-T2 for **WVALID** is the time for master number Mx to be pushed onto the FIFO that stores routing information
- there is zero latency for subsequent transfers after T3 because the **WDATA** route is set and can not be interleaved with other data, `WriteInterleaveCapability` = 1.

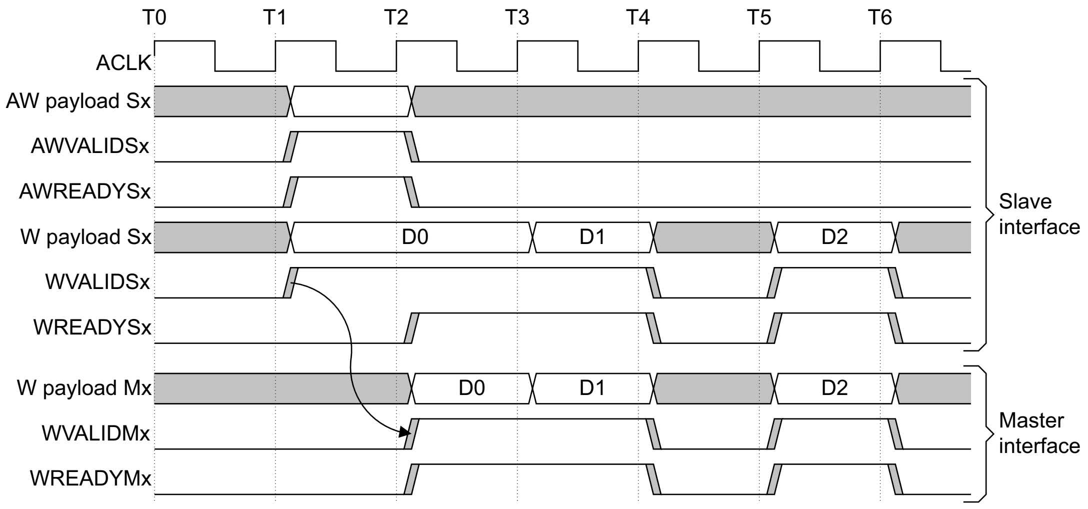


Figure 2-12 Write data channel with a write interleave capability of one

Write interleave capability greater than one

Figure 2-13 shows the write data channel to an interconnect master interface with a write interleave capability greater than one:

- the master sends the write address and data simultaneously at T1
- there is zero arbitration latency
- the slave has zero wait states
- the one cycle latency at T1-T2 for **WVALID** is the time for master number Mx to be pushed onto the FIFO that stores routing information
- the master drives **WVALIDSx** LOW at T3-T4 and loses its arbiter allocation
- the one cycle latency at T4-T5 is the time for the master to regain its arbiter allocation
- there is no latency between data items D1 and D2 because the master retains its arbiter allocation.

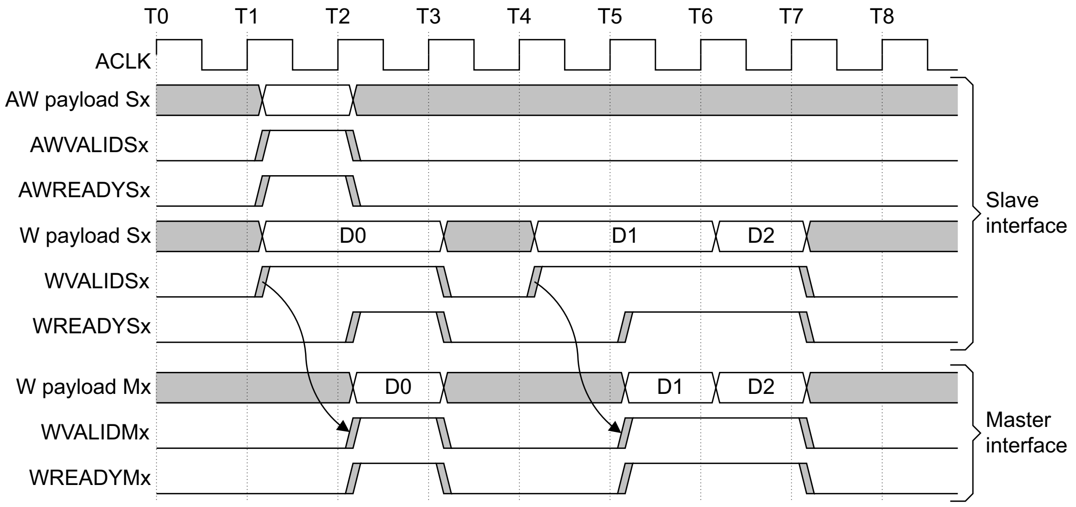


Figure 2-13 Write data channel with a write interleave capability greater than one

2.4.3 Write response channel

Figure 2-14 shows the write response channel:

- the write address and data are sent simultaneously at T1
- the slave has zero wait states
- there is zero latency between the channel master and slave interfaces.

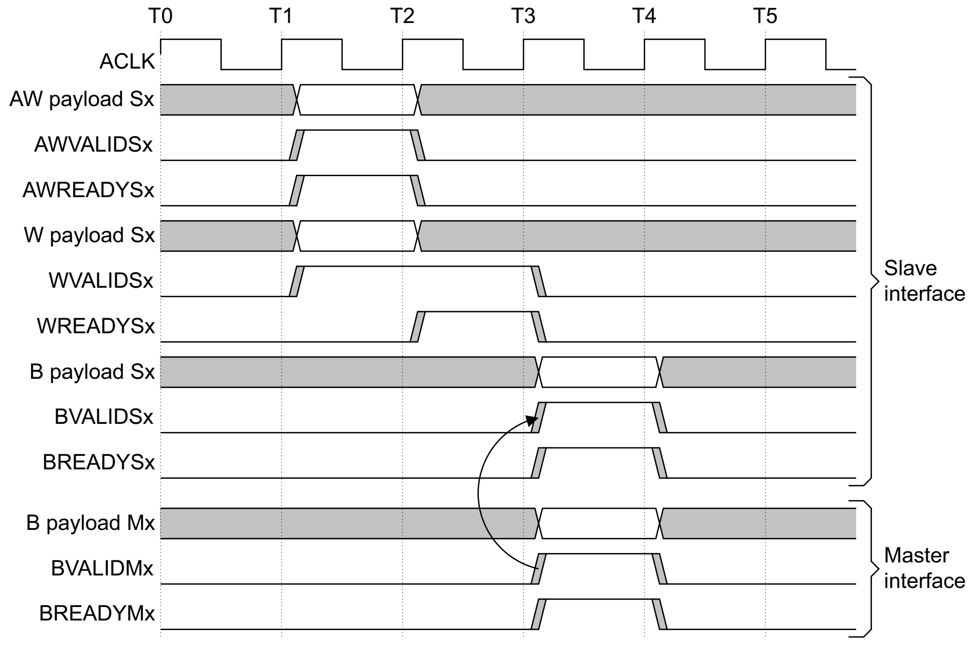


Figure 2-14 Write response channel

2.4.4 Read data channel

Figure 2-15 shows the read data channel. There is zero arbitration latency.

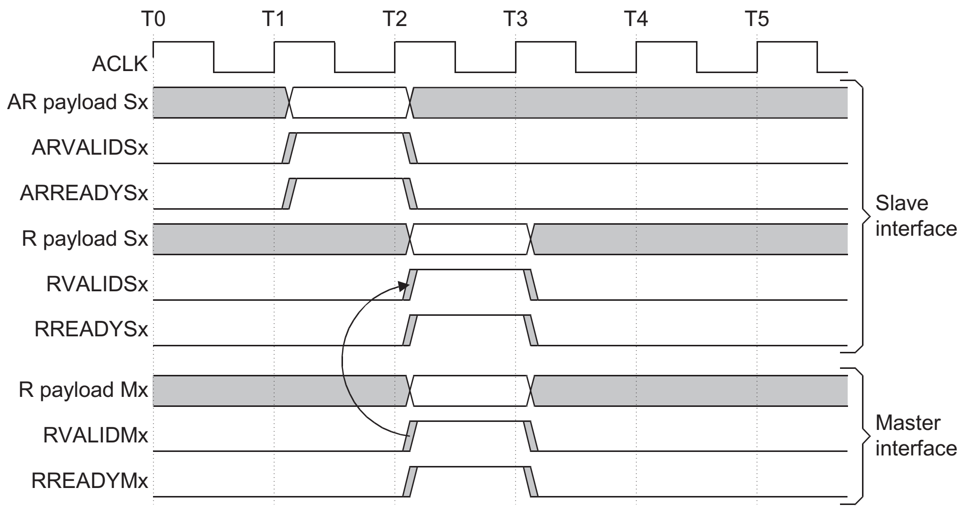


Figure 2-15 Read data channel

2.5 Physical characteristics

This section describes the ACI physical characteristics in:

- *Clock and reset*
- *AC characteristics*
- *Gate count*
- *Production test methodology*.

2.5.1 Clock and reset

The interconnect operates in a single clock and reset domain.

2.5.2 AC characteristics

The interconnect conforms to the following timing characteristics:

- external input delay of 20%
- external output delay of 20%.

The value is expressed as a percentage of the clock cycle.

Trial synthesis of the example4x5 interconnect suggests that timing can be met at 200MHz, using Synopsys Design Compiler with zero wire-load models.

Summary of the synthesized example4x5 interconnect:

- four slave interfaces and five master interfaces
- slave interface read and write acceptance capabilities of eight
- cyclic dependency schemes:
  - one single slave
  - two hybrids
  - one unique ID.
- 64-bit data width
- SAGE-HS TSMC CL013G library.

2.5.3 Gate count

The example4x5 interconnect with the constraints mentioned above has an estimated gate count of 30K NAND2x1 equivalents.

2.5.4 Production test methodology

The ACI is designed for full scan test insertion. Scan pins are not present at the top-level. No other dedicated test features are included.

Appendix A
Signal Descriptions

This appendix describes the signals that interface with the ACI. It contains the following sections:

A.1 AXI signals

This section describes the signal connections for a 1x1 64-bit interconnect with the ID width of the slave interfaces configured as two bits wide and a single remap alias.

The ACI uses standard AMBA AXI signals as described in the *AMBA AXI Protocol Specification*. The AXI channel signals described in this section are appended with:

- the letter **M** for signals that connect between the ACI master interface and the external slave
- the letter **S** for signals that connect between the ACI slave interface and the external master
- a numerical suffix that indicates the interface number.

The ACI signals are shown in:
- *Global*
- *Write channels* on page A-3
- *Read channels* on page A-4.

A.1.1 Global

Figure A-1 shows the ACI global signal connections for a configuration with one remap range and one master interface specified in the configuration script.

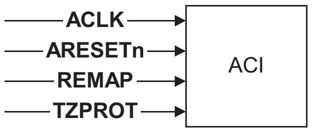


Figure A-1 Global signal connections

Note

The **REMAP** signal is not implemented if no remap range is specified in the configuration script. See the *PrimeCell AXI Configurable Interconnect (PL300) Implementation Guide*.

A.1.2 Write channels

Figure A-2 shows the signal connections for the ACI write channels.

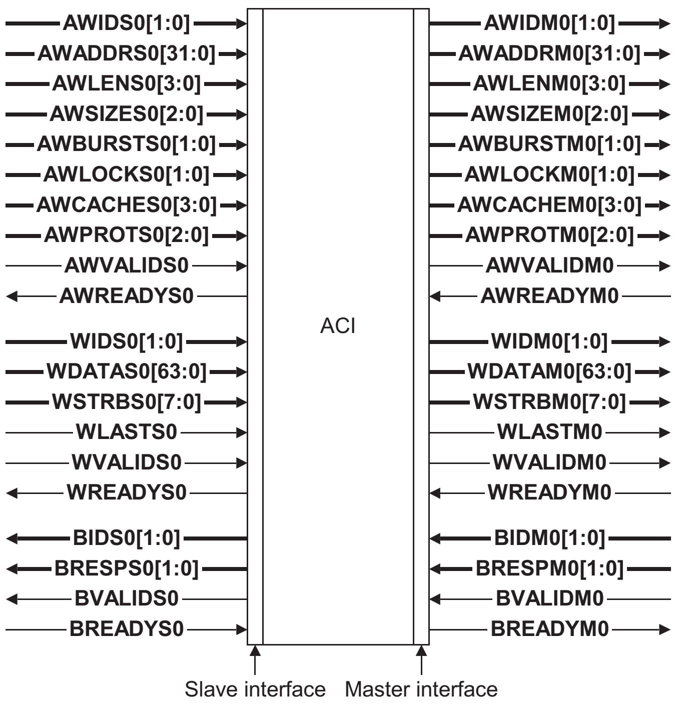


Figure A-2 Signal connections for write channels

Note

If `IdWidth`=0 is specified in the configuration file then no ID ports are implemented on the slave interface. See the *PrimeCell AXI Configurable Interconnect (PL300) Implementation Guide*.

A.1.3 Read channels

Figure A-3 shows the signal connections for the ACI read channels.

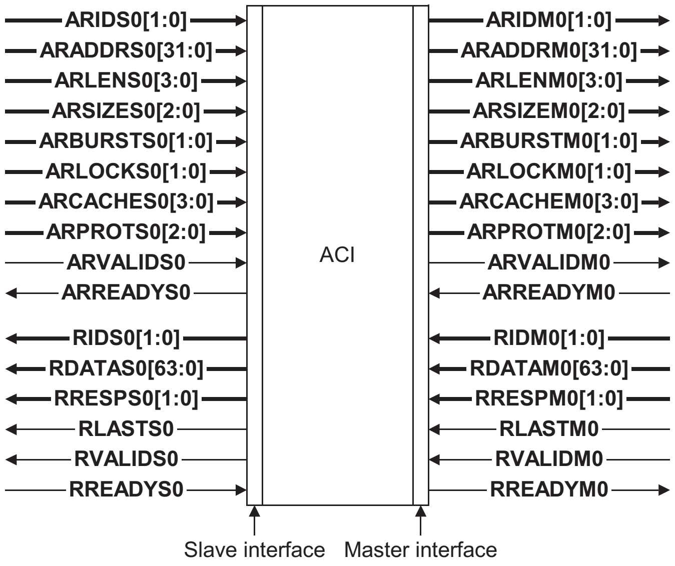


Figure A-3 Signal connections for read channels

Note

If `IdWidth=0` is specified in the configuration file then no ID ports are implemented on the slave interface. See the *PrimeCell AXI Configurable Interconnect (PL300) Implementation Guide*.

A.2 Non-AXI signals

Table A-1 lists signals that are present on the ACI but are not described in the *AMBA AXI Protocol Specification*.

Table A-1 Non-standard signals

Name	Type	Source	Description
REMAP	Input	Remap controller	This changes the memory map from that required during boot-up to that for normal operation: 0 = Remap memory alias or move is not active 1 = Remap memory alias or move is active. See *Memory map* on page 2-10.
TZPROT	Input	Security controller[a][b]	Protection indicator for master interfaces: 0 = secure 1 = non-secure. See *TrustZone support* on page 2-12.

a. You can tie this signal off at the interconnect for a static configuration.

b. This component is available from ARM Limited. See the *PrimeCell Infrastructure AMBA 3 TrustZone Protection Controller (BP147) Technical Overview* for more information.

Appendix B
Interface Attributes

This appendix describes the ACI master interface and ACI slave interface attributes. It contains the following sections:

B.1 Master interface

Table B-1 lists the ACI master interface attributes.

Table B-1 Master interface attributes

Attribute	Description	Value
Read ID capability	The maximum number of different **ARID** values that a master interface can generate for all active read transactions at any one time.	Master-dependent[a]
Read ID width	The number of bits in the **ARID** bus.	Indirectly configurable[b]
Read issuing capability	The maximum number of active read transactions that a master interface can generate.	Master-dependent[a]
Write ID capability	The maximum number of different **AWID** values that a master interface can generate for all active write transactions at any one time	Master-dependent[a]
Write ID width	The number of bits in the **AWID** and **WID** buses.	Indirectly configurable[b]
Write issuing capability	The maximum number of active write transactions that a master interface can generate.	Configurable[c]

a. Master-dependent means that the interface adopts the attribute value of the external masters that are connected through the interconnect.

b. Indirectly configurable means that this value is derived automatically when you run the ACI XML configuration file. An example of this is the master interface ID width as described in *Slave interface ID width* on page 2-3.

c. Configurable means that the value can be set when you edit the XML configuration file. See *Configurable interconnect options* on page 2-2 for more information.

B.2 Slave interface

Table B-2 lists the ACI slave interface attributes.

Table B-2 Slave interface attributes

Attribute	Description	Value
Read acceptance capability	The maximum number of active read transactions that a slave interface can accept.	Configurable[a]
Read data reorder depth	The number of active read transactions for which a slave interface may transmit data. This is counted from the earliest transaction.	Slave-dependent[b]
Write acceptance capability	The maximum number of active write transactions that a slave interface can accept.	Configurable[a]
Write interleave depth	The number of active write transactions for which the slave interface can receive data. This is counted from the earliest transaction.	1

a. Configurable means that the value can be set when you edit the XML configuration file. See *Configurable interconnect options* on page 2-2 for more information

b. Slave-dependent means that the interface adopts the attribute value of the external slaves that are connected through the interconnect.

Glossary

This glossary describes some of the terms used in ARM manuals.

Advanced eXtensible Interface (AXI)

A bus protocol that supports separate address/control and data phases, unaligned data transfers using byte strobes, burst-based transactions with only start address issued, separate read and write data channels to enable low-cost DMA, ability to issue multiple outstanding addresses, out-of-order transaction completion, and easy addition of register stages to provide timing closure.The AXI protocol also includes optional extensions to cover signaling for low-power operation.

AXI is targeted at high performance, high clock frequency system designs and includes a number of features that make it very suitable for high speed sub-micron interconnect.

Advanced High-performance Bus (AHB)

A bus protocol with a fixed pipeline between address/control and data phases. It only supports a subset of the functionality provided by the AMBA AXI protocol. The full AMBA AHB protocol specification includes a number of features that are not commonly required for master and slave IP developments and ARM Limited recommends only a subset of the protocol is usually used. This subset is defined as the AMBA AHB-Lite protocol.

See also Advanced Microcontroller Bus Architecture and AHB-Lite.

Advanced Microcontroller Bus Architecture (AMBA)

A family of protocol specifications that describe a strategy for the interconnect. AMBA is the ARM open standard for on-chip buses. It is an on-chip bus specification that details a strategy for the interconnection and management of functional blocks that make up a *System-on-Chip* (SoC). It aids in the development of embedded processors with one or more CPUs or signal processors and multiple peripherals. AMBA complements a reusable design methodology by defining a common backbone for SoC modules.

Advanced Peripheral Bus (APB)

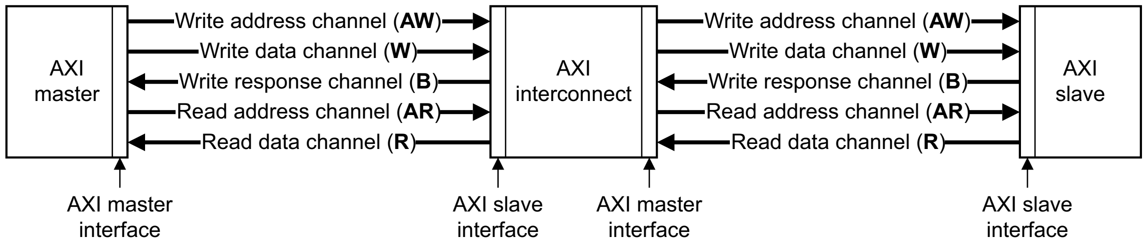
A simpler bus protocol than AXI and AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports. Connection to the main system bus is through a system-to-peripheral bus bridge that helps to reduce system power consumption.

AHB	*See* Advanced High-performance Bus.
AHB-Lite	A subset of the full AMBA AHB protocol specification. It provides all of the basic functions required by the majority of AMBA AHB slave and master designs, particularly when used with a multi-layer AMBA interconnect. In most cases, the extra facilities provided by a full AMBA AHB interface are implemented more efficiently by using an AMBA AXI protocol interface.
Aligned	A data item stored at an address that is divisible by the number of bytes that defines the data size is said to be aligned. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses that are divisible by four and two respectively.
AMBA	*See* Advanced Microcontroller Bus Architecture.
APB	*See* Advanced Peripheral Bus.
AXI	*See* Advanced eXtensible Interface.

AXI channel order and interfaces

The block diagram shows:

- the order in which AXI channel signals are described
- the master and slave interface conventions for AXI components.



AXI terminology

The following AXI terms are general. They apply to both masters and slaves:

Active read transaction

A transaction for which the read address has transferred, but the last read data has not yet transferred.

Active transfer

A transfer for which the **xVALID**[1] handshake has asserted, but for which **xREADY** has not yet asserted.

Active write transaction

A transaction for which the write address or leading write data has transferred, but the write response has not yet transferred.

Completed transfer

A transfer for which the **xVALID/xREADY** handshake is complete.

Payload The non-handshake signals in a transfer.

Transaction An entire burst of transfers, comprising an address, one or more data transfers and a response transfer (writes only).

Transmit An initiator driving the payload and asserting the relevant **xVALID** signal.

Transfer A single exchange of information. That is, with one **xVALID/xREADY** handshake.

The following AXI terms are master interface attributes. To obtain optimum performance, they must be specified for all components with an AXI master interface:

Combined issuing capability

The maximum number of active transactions that a master interface can generate. This is specified instead of write or read issuing capability for master interfaces that use a combined storage for active write and read transactions.

1. The letter **x** in the signal name denotes an AXI channel as follows:

AW	Write address channel.
W	Write data channel.
B	Write response channel.
AR	Read address channel.
R	Read data channel.

Read ID capability

The maximum number of different **ARID** values that a master interface can generate for all active read transactions at any one time.

Read ID width

The number of bits in the **ARID** bus.

Read issuing capability

The maximum number of active read transactions that a master interface can generate.

Write ID capability

The maximum number of different **AWID** values that a master interface can generate for all active write transactions at any one time.

Write ID width

The number of bits in the **AWID** and **WID** buses.

Write interleave capability

The number of active write transactions for which the master interface is capable of transmitting data. This is counted from the earliest transaction.

Write issuing capability

The maximum number of active write transactions that a master interface can generate.

The following AXI terms are slave interface attributes. To obtain optimum performance, they must be specified for all components with an AXI slave interface

Combined acceptance capability

The maximum number of active transactions that a slave interface can accept. This is specified instead of write or read acceptance capability for slave interfaces that use a combined storage for active write and read transactions.

Read acceptance capability

The maximum number of active read transactions that a slave interface can accept.

Read data reordering depth

The number of active read transactions for which a slave interface can transmit data. This is counted from the earliest transaction.

Write acceptance capability

The maximum number of active write transactions that a slave interface can accept.

Write interleave depth

The number of active write transactions for which the slave interface can receive data. This is counted from the earliest transaction.

Beat	Alternative word for an individual transfer within a burst. For example, an INCR4 burst comprises four beats. *See also* Burst.
Burst	A group of transfers to consecutive addresses. Because the addresses are consecutive, there is no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AMBA are controlled using signals to indicate the length of the burst and how the addresses are incremented. *See also* Beat.
Byte	An 8-bit data item.
Byte lane strobe	A signal that is used for unaligned or mixed-endian data accesses to determine which byte lanes are active in a transfer. One bit of this signal corresponds to eight bits of the data bus.
Unaligned	A data item stored at an address that is not divisible by the number of bytes that defines the data size is said to be unaligned. For example, a word stored at an address that is not divisible by four.