Arm® CoreLink™ XHB-500 Bridge Technical Reference Manual

Revision: r0p0

AXI5 to AHB5 bridge and AHB5 to AXI5 bridge

arm

101375_0000_00_en

Arm® CoreLink™ XHB-500 Bridge Technical Reference Manual

AXI5 to AHB5 bridge and AHB5 to AXI5 bridge

Release Information

Document History

Issue	Date	Confidentiality	Change
0000-00	30 November 2018	Non-Confidential	First release for r0p0 BET.

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is for a Beta product, that is a product under development.

Web Address

http://www.arm.com

Contents

Arm® CoreLink™ XHB-500 Bridge Technical Reference Manual AXI5 to AHB5 bridge and AHB5 to AXI5 bridge

Preface

This preface introduces the *Arm® CoreLink™ XHB-500 Bridge Technical Reference Manual AXI5 to AHB5 bridge and AHB5 to AXI5 bridge*.

It contains the following:

About this book

This book describes the functionality of the bridges in the Arm® CoreLink™ XHB-500 product. It also provides the signal descriptions.

Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

r*m* Identifies the major revision of the product, for example, r1.

p*n* Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This book is written for system designers and programmers who are designing or programming a *System on Chip* (SoC) that uses the XHB-500.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction
This chapter introduces the XHB-500 bridges.

Chapter 2 Functional Description, AXI5 to AHB5 bridge
This chapter describes the functionality of the AXI5 to AHB5 bridge.

Chapter 3 Functional Description, AHB5 to AXI5 bridge
This chapter describes the functionality of the AHB5 to AXI5 bridge.

Chapter 4 Programmers Model
This chapter describes the programmers model.

Appendix A Signal Descriptions
This appendix describes the interface signals of each XHB-500 bridge.

Appendix B Revisions
This appendix describes the technical changes between released issues of this book.

Glossary

The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the *Arm® Glossary* for more information.

Typographic conventions

italic
Introduces special terminology, denotes 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.

`mono`space
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>
: Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS
: Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The following figure 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.

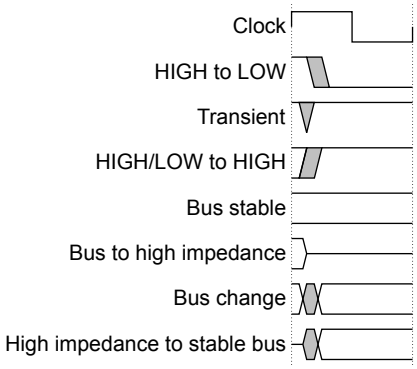

Figure 1 Key to timing diagram conventions

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.
- LOW for active-LOW signals.

Lowercase n
: At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This book contains information that is specific to this product. See the following documents for other relevant information.

Arm publications

- *Arm® AMBA® AXI and ACE Protocol Specification* (Arm IHI 0022).
- *Arm® AMBA® 5 AHB Protocol Specification* (Arm IHI 0033).
- *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces* (Arm IHI 0068).
- *Arm® CoreLink™ AXI4 to AHB-Lite XHB-400 Bridge Technical Reference Manual* (Arm DDI 0523).

The following confidential books are only available to licensees or require registration with Arm:

- *Arm® CoreLink™ XHB-500 Bridge Configuration and Integration Manual* (Arm 101376).

Feedback

Feedback on this product

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

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to *errata@arm.com*. Give:

- The title *Arm CoreLink XHB-500 Bridge Technical Reference Manual AXI5 to AHB5 bridge and AHB5 to AXI5 bridge*.
- The number 101375_0000_00_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1
Introduction

This chapter introduces the XHB-500 bridges.

It contains the following sections:

1.1 About the XHB-500 bridges

The XHB-500 product provides an AMBA AXI5 to AHB5 bridge and an AHB5 to AXI5 bridge.

The AXI5 to AHB5 bridge translates AXI5 transactions into the corresponding AHB transfers. The bridge has an AXI5 slave interface and an AHB5 master interface.

The AHB5 to AXI5 bridge translates AHB5 transfers into the corresponding AXI transactions. The bridge has an AHB5 slave interface and an AXI5 master interface.

AXI5 to AHB5 bridge overview

The AXI5 to AHB5 is a low-latency bridge that performs no transaction buffering.

The following figure shows the interfaces of the AXI5 to AHB5 bridge.

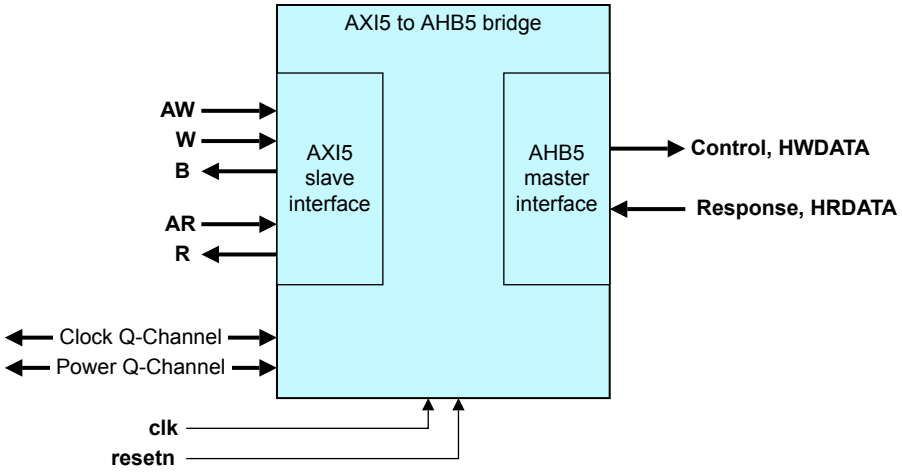

Figure 1-1 AXI5 to AHB5 interfaces

The main features are:

- Single power domain.
- Single clock domain.
- Configurable data width.
- AXI5 slave interface features:
  - AXI5 protocol support.
  - AXI4 protocol support.
  - Fixed address width.
  - Registered or unregistered interface.
  - Single Exclusive accesses. Exclusive bursts are not supported.
  - Unaligned accesses.
  - Conversion of sparse write transactions, when the `HWSTRB_ENABLE` configuration parameter is set to OFF.
  - Supports all burst types.
- AHB5 master interface features:
  - AHB5 support.
  - AHB-Lite support, which requires several signals to be tied off.
  - Fixed address width.
  - Registered or unregistered interface.
  - Exclusive accesses. For AHB-Lite, extra glue logic is required.
  - No support for locked transfers.
  - Write strobe support using the **hwstrb** signal, when the `HWSTRB_ENABLE` configuration parameter is set to ON. The **hwstrb** signal is not present in the *Arm® AMBA® 5 AHB Protocol Specification*.
- Q-Channel interface for clock control.
- Q-Channel interface for power control.

The bridge does not support endian conversion.

AHB5 to AXI5 bridge overview

The AHB5 to AXI5 is a low-latency bridge.

The following figure shows the interfaces of the AHB5 to AXI5 bridge.

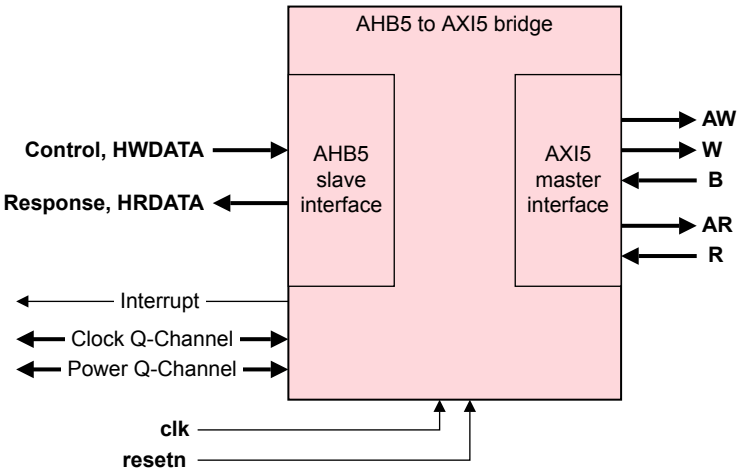

Figure 1-2 AHB5 to AXI5 interfaces

The main features are:

- Single power domain.
- Single clock domain.
- Configurable data width.
- AHB5 slave interface features:
  - AHB5 protocol support.
  - Fixed address width.
  - Registered or unregistered interface.
  - Support for early write response.
  - Supports all burst types.
- AXI5 master interface features:
  - AXI5 support.
  - Fixed address width.
  - Registered or unregistered interface.
  - RAW hazard checking for early write response.
- Buffered write error interrupt.
- Q-Channel interface for clock control.
- Q-Channel interface for power control.

The bridge does not support endian conversion.

1.2 Compliance

The XHB-500 Bridge complies with the following specifications:

- *Arm® AMBA® AXI and ACE Protocol Specification*.
- *Arm® AMBA® 5 AHB Protocol Specification*.
- *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces*.

This *Technical Reference Manual* (TRM) complements the protocol specifications. The TRM does not duplicate information from these sources.

1.3 Product documentation

Documentation that is provided with this product includes a *Technical Reference Manual* (TRM) and a *Configuration and Integration Manual* (CIM), together with protocol information.

For relevant protocol information that relates to this product, see *Additional reading* on page 8.

The XHB-500 documentation is as follows:

Technical Reference Manual

The TRM describes the functionality and the effects of functional options on the behavior of the XHB-500 bridges. It is required at all stages of the design flow. The choices that are made in the design flow can mean that some behaviors that the TRM describes are not relevant.

The TRM complements protocol specifications and relevant external standards. It does not duplicate information from these sources.

Configuration and Integration Manual

The CIM describes:

- The available build configuration options.
- How to configure the *Register Transfer Level* (RTL) with the build configuration options.
- How to integrate the XHB-500 bridges into a SoC.
- How to implement the XHB-500 bridges into your design.
- The processes to validate the configured design.

The CIM is a confidential book that is only available to licensees.

1.4 Product revisions

This section describes the differences in functionality between product revisions:

r0p0 First release.

Chapter 2
Functional Description, AXI5 to AHB5 bridge

This chapter describes the functionality of the AXI5 to AHB5 bridge.

It contains the following sections:

2.1 AMBA bus properties

The AMBA protocols define multiple property types that indicate the capabilities of a device.

The following table lists the AXI5 properties of the AXI5 to AHB5 bridge.

Table 2-1 AXI5 properties

AXI5 property	Value	Comment
Ordered_Write_Observation	TRUE	Improved support for the Producer/Consumer ordering model.
Multi_Copy_Atomicity	TRUE	Support for multi-copy atomicity.
Atomic_Transactions	FALSE	-
Check_Type	FALSE	-
Poison	FALSE	-
QoS_Accept	FALSE	AXI4 QoS is supported, but QoS_Accept signaling is not supported.
Trace_Signals	FALSE	-
Loopback_Signals	TRUE	-
Wakeup_Signals	TRUE	Q-Channel activity is generated from the **awakeup** input signal.
Untranslated_Transactions	FALSE	-
NSAccess_Identifiers	TRUE	Supported by using the **hnsaid[3:0]** sideband signals on the AHB5 master interface.

The following table lists the AHB5 properties of the AXI5 to AHB5 bridge.

Table 2-2 AHB5 properties

AHB5 property	Value	Comment
Extended_Memory_Types	True	**hprot** value is generated from **axcache**, **axdomain**, and **axprot**.
Secure_Transfers	True	**axprot[1]**, the Secure bit, is converted to **hnonsec**.
Endian	False	AXI uses a byte-invariant bus, both of its types are directly convertible to AHB. AHB word-invariant endianness is not supported.
Stable_Between_Clock	False	Not supported.
Exclusive_Transfers	True	All AHB Exclusive Transfers are supported, but AXI Exclusive bursts are not supported.
Multi_Copy_Atomicity	True	Pass-through, no buffering.

2.2 Burst conversions

The AXI5 to AHB5 bridge converts the AXI transactions into AHB transfers. To indicate the type of AHB transfer, the bridge sets the value of the **hburst**, **hsize**, and **htrans** signals.

hburst[2:0] mapping

For AXI transfers with a length that matches natural AHB bursts such as 4, 8, or 16, the AXI5 to AHB5 uses fixed-length bursts.

For AXI transfers with a length of 1 and for fixed address transfers, the bridge uses AHB bursts of type SINGLE.

For other transfer lengths, the bridge uses AHB INCR bursts.

The burst boundary for AXI is 4KB, but in AHB it is 1KB. Therefore, if an AXI burst crosses the 1KB boundary, then the bridge sets the AHB burst type to INCR and at the crossing point it changes the transfer type from SEQ to NONSEQ.

The following table shows the mapping of AXI5 burst types to AHB5 burst types.

Table 2-3 AXI5 burst type to AHB5 burst type mapping

AXI burst type & length	hburst[2:0]	Conditions
FIXED, any length	SINGLE	Never crosses the 1KB border, but can be sparse.
INCR, length 1		
INCR, length 4	INCR4	The mapping is possible only when all the following conditions apply: • The burst does not cross 1KB border. • For writes, **awsparse** is LOW or **hwstrb** is in use. • A read is not unaligned Non-modifiable. • AXI transaction size is not 1024. • AXI burst type is not WRAP length 16.
INCR, length 8	INCR8	
INCR, length 16	INCR16	
WRAP, length 4	WRAP4	
WRAP, length 8	WRAP8	
WRAP, length 16	WRAP16	
Any	INCR	This mapping occurs when any of the following conditions occur: • The incoming AXI burst is of a type and length combination that is not previously listed. • The burst crosses a 1KB border. • For writes, **awsparse** is HIGH. • An AXI WRAP burst that has a length of 2. • An AXI WRAP burst has a length of 16. • An AXI data width of 1024 bits. • The read is unaligned Non-modifiable. If the bridge splits a burst, then it issues multiple INCR bursts.

hsize[2:0] mapping

The bridge sets **hsize[2:0]** to the value of **axsize[2:0]** unless it has to split an AXI transaction to multiple AHB transfers when handling sparse writes or Non-modifiable reads. In these situations, the bridge sets **hsize[2:0]** to the largest possible aligned size for the next AHB transfer.

htrans[1:0] mapping

The **htrans[1:0]** signal indicates the transfer type and whether a beat continues or splits a burst. The following table lists **htrans[1:0]** settings for various conditions.

Table 2-4 htrans[1:0] mapping

Conditions for htrans[1:0] mapping	htrans[1:0]	Comment
Default	IDLE	-
An AXI burst is in progress but the relevant AXI channel (W, R, or B) is not ready. No AHB burst is in progress.		
All AXI beat strobes are LOW.		
First beat of an AXI transaction.	NONSEQ	-
hburst == SINGLE		-
1KB border crossing beat.		-
Sparse AXI beat.		NONSEQ for all AHB transfers belonging to the AXI beats.
Previous AXI beat was sparse.		
Wrap-back beat.		Wrap-back beat of an AHB-INCR-converted AXI WRAP transaction. See the *Any* on page 2-19 row in the **hburst[2:0]** mapping table.
AHB burst is in progress, but the relevant AXI channel (W, R, or B) is not ready.	BUSY	**hburst** != SINGLE
Any other cases.	SEQ	-

2.3 1KB boundary crossing

The AHB protocol requires that bursts do not cross 1KB boundaries. Since AXI5 transactions can cross a 1KB boundary, the AXI5 to AHB5 bridge must ensure that a transfer does not cross a 1KB boundary.

If an AXI transaction crosses a 1KB boundary, the bridge converts the transaction to undefined length INCR bursts, and **htrans** is set to NONSEQ on the first address after crossing the 1KB boundary.

2.4 Protection control translation

For protection control, the AXI5 to AHB5 bridge maps the **axcache** and **axprot** AXI signals to the **hprot** and **hnonseq** AHB signals.

The following table shows how the AXI5 to AHB5 bridge converts the protection control from AXI5 to AHB5.

Table 2-5 Protection control mapping

AHB signals	AHB bit number, function	Value is derived from AXI signal:
hprot	Bit[6], Shareable	**axdomain**. The **hprot[6]** signal is set to: • 0 if **axdomain** == `0b00` or `0b11`. A Non-shareable or System transaction translates to a Non-shareable AHB transfer. • 1 if **axdomain** == `0b01` or `0b10`. An Inner Shareable or Outer Shareable transaction translates to a Shareable AHB transfer. **ardomain** and **awdomain** do not exist in the AXI protocol. The ACE protocol uses a concept that is called shareability domains and the **axdomain** signals indicate the shareability domain of a transaction. See the *Arm® AMBA® AXI and ACE Protocol Specification* for more information.
	Bit[5], Allocate	**arcache[2]** for read transactions. **awcache[3]** for write transactions.
	Bit[4], Lookup	**axcache[3:2]** The **hprot[4]** signal is set to: • 0 if **axcache[3:2]** == `0b00`. • 1 if **axcache[3:2]** != `0b00`.
	Bit[3], Modifiable	**axcache[1]**
	Bit[2], Bufferable	**axcache[0]** The **hprot[2]** signal is set to: • **axcache[0]** for all transactions, except for a Normal Non-cacheable Bufferable transaction. • !**axcache[0]** for a Normal Non-cacheable Bufferable transaction (**axcache[3:0]** == `0b0011`).
	Bit[1], Privileged	**axprot[0]**
	Bit[0], Data/Opcode	!**axprot[2]**
hnonsec	Bit[0], Secure	**axprot[1]**

2.5 Exclusive and locked accesses

The AXI protocol supports Exclusive bursts but the AHB protocol only supports single (length 1) Exclusive accesses. Therefore, if the AXI5 to AHB5 bridge receives an AXI Exclusive burst, then it translates the burst to normal (Non-exclusive) AHB transfers. If the bridge receives a single AXI Exclusive transaction, then it translates the transaction to an Exclusive AHB transfer.

The AXI5 to AHB5 bridge does not support single sparse Exclusive writes, because splitting the write transaction would create an Exclusive AHB burst. As the preceding Exclusive read might have been answered with HEXOKAY, the bridge always responds with SLVERR for a single sparse Exclusive write.

Note

The bridge returns SLVERR because although OKAY is a valid Exclusive response, an OKAY response could cause the AXI master to repeat the Exclusive write indefinitely.

The bridge uses the **axid** values to identify which AXI master issues an Exclusive access. For the AHB transfer, the bridge copies the **axid** value to **hmaster**.

The following table shows the AXI5 to AHB5 Exclusive transaction mapping.

Table 2-6 AXI5 to AHB5 bridge Exclusive transaction mapping

AXI Exclusive access type	AHB transfer
AXI Exclusive read.	Exclusive AHB transfers, for a single AXI transaction.
	Normal AHB transfers, for a burst AXI transaction.
AXI non-sparse Exclusive write.	Exclusive AHB transfers, for a single AXI transaction.
	Normal AHB transfers, for a burst AXI transaction.
AXI sparse Exclusive write.	Normal (SLVERR) AHB transfer, for a single AXI transaction.
	Normal AHB transfers, for a burst AXI transaction.

Comparison to the XHB-400 behavior

The previous generation AXI to AHB bridge is the CoreLink XHB-400, which supports the AMBA 3 AHB-Lite protocol. The AHB-Lite protocol does not support Exclusive accesses, so to support AHB-Lite Exclusive transfers, the XHB-400 includes the **EXREQ** and **EXRESP** signals. If necessary, you can add external glue logic to the AXI5 to AHB5 bridge, so that the logic generates **EXREQ** and **EXRESP**, with the limitation that Exclusive bursts are not supported. See the *Arm® CoreLink™ XHB-500 Bridge Configuration and Integration Manual* for more information about the glue logic.

Locked transfers for AXI Non-modifiable transactions

The AXI protocol requires that Non-modifiable transactions must not be split into multiple transactions or merged with other transactions. However, the AXI5 to AHB5 bridge must split sparse writes and unaligned Non-modifiable reads to generate multiple AHB bursts. The bridge issues these AHB bursts as Non-modifiable with **hmastlock** HIGH, and it allows no arbitration during the AXI burst. Asserting **hmastlock** ensures that the bridge or an AHB interconnect does not separate the bursts because they belong to the same lock sequence. The bridge inserts an IDLE transfer after the lock sequence.

While an AXI burst can cross the 1K address range, the AHB protocol requires that all transfers in a locked sequence are to the same slave address region. If a Non-modifiable AXI burst crosses a 1K

address boundary, the bridge generates Non-modifiable AHB bursts with **hmastlock** LOW, and it responds with SLVERR. In this scenario, the bridge allows arbitration.

Note

Although not recommended by Arm, the bridge can issue transactions that start, end, or contain only locked IDLE transfers, depending on the incoming AXI transaction or the AXI response channel availability.

2.6 Sparse writes

Although the AHB protocol does not support write strobes, the AXI5 to AHB5 bridge provides two methods to handle an AXI sparse write transaction. During implementation of the bridge, the setting of the `HWSTRB_ENABLE` configuration option controls which method the bridge implements.

`HWSTRB_ENABLE` == OFF

When `HWSTRB_ENABLE == OFF`, the bridge uses the **awsparse** and **wstrb** inputs to control how it issues AHB write transfers that contain sparse data. **awsparse** is a sideband signal that an AXI master can assert for any write transaction, including unaligned writes, that might contain sparse beats. When **awsparse** is HIGH, the bridge checks the **wstrb** strobe lanes for every AXI beat, and splits the beat into several AHB transfers if byte lanes are omitted. To issue the minimum number of transfers, the bridge uses the largest possible aligned transfer size. The mapping process does not insert any delay cycles to the transaction. The bridge propagates all bursts as INCR transfers, to optimize for non-sparse burst translations that **awsparse** might pessimistically indicate.

The bridge gives priority to reads over writes. Therefore, a read burst always executes to completion, but the bridge can stall a sparse write transaction when it receives a read burst. Sparse write transactions continue after the read burst completes. See *2.11 Read and write transaction scheduling* on page 2-31.

Note

- To ensure that **awsparse** passes through AXI interconnects, you can use an **awuser** signal.
- If the master does not assert **awsparse** signal for a write transaction that has a beat with a sparse **wstrb** value, or the write transaction is unaligned, then the bridge indicates an error condition by setting **bresp** to SLVERR.
- If the AXI master interface does not provide an **awsparse** output, then the bridge **awsparse** must be tied HIGH.

`HWSTRB_ENABLE` == ON

When `HWSTRB_ENABLE == ON`, the AHB master interface includes a write strobe sideband signal, **hwstrb**, and the AXI slave interface does not have an **awsparse** input. For a write transaction, the AXI5 to AHB5 bridge copies the **wstrb** value to **hwstrb**. Any AHB slave that supports write strobes, can use **hwstrb** to select the byte lanes that hold valid data.

The bridge aligns any unaligned AXI addresses and uses the strobes to convey the unalignment information, and it translates the bursts as if they were not sparse. See *2.2 Burst conversions* on page 2-19 for more information.

2.7 Address alignment

The AXI and AHB protocols handle transaction addresses differently, so the AXI5 to AHB5 bridge must perform address alignment. The AHB protocol does not support unaligned transfers.

In AXI, the master issues only the starting address for the whole transaction and it is the responsibility of the slave to increment the address for the transaction beats. The AXI address can be unaligned to the boundary that **axsize[2:0]** determines, although it is only the first beat that is unaligned, the remaining beats are aligned. In AHB, the master calculates and issues all beat addresses in a burst and each beat is aligned.

Therefore, the AXI5 to AHB5 calculates aligned AHB addresses for all beats from the incoming single AXI address. The bridge can either align the address or split the AXI transaction into several aligned AHB beats. The bridge also calculates wrap addresses.

When the bridge receives:

An unaligned write transaction
The bridge treats the transaction as a sparse transaction, with the restriction that the address (**awaddr[31:0]**) and **wstrb** signals must be consistent. Since the first beat is always sparse, the AXI master must drive **awsparse** HIGH if the bridge is configured with `HWSTRB_ENABLE == OFF`.

Modifiable unaligned read transaction
The bridge aligns the address, so the first beat of the burst contains more data than the AXI master expects. The AXI master can ignore the extra data.

Non-modifiable unaligned read transaction
A Non-modifiable read transaction is not permitted to perform a speculative read of Device memory, so:

- If the AXI burst type is INCR or WRAP, then the bridge splits the first AXI transfer into several aligned AHB beats, transferring the largest possible aligned beats. The bridge transfers the remainder of the transaction as an undefined length AHB INCR burst.
- For AXI FIXED bursts, the bridge splits the transaction so that all the AHB beats are aligned.

2.8 User signals

The AXI5 to AHB5 bridge supports four AXI User signals and three AHB User signals. The signal width of the address User signals, the read User signals, and the write User signals can be configured to different values.

The following table shows how these signals are mapped between the interfaces.

Table 2-7 User sideband signals mapping

AXI5 slave interface		AHB5 master interface			Comment
Signal	**Direction**	**Signal**	**Direction**	**Phase**	
aruser	Input	**hauser**	Output	Address	Read transfers. The same **hauser** value is output on each beat that is associated with an AXI5 transaction.
awuser					Write transfers. The same **hauser** value is output on each beat that is associated with an AXI5 transaction.
wuser	Input	**hwuser**	Output	Data	Only valid on write transfers. For non-sparse writes, the bridge separately translates each beat, and the respective **hwuser** equals **wuser** for all beats. If a transfer is sparse, one AXI transfer might translate to multiple AHB beats, so the **wuser** value repeats on **hwuser**.
ruser	Output	**hruser**	Input		Only valid on read transfers. The value might differ on each beat. The last **hruser** is propagated when multiple AHB transfers are accumulated for the first beat of an unaligned Non-modifiable read.

Note

The AXI5 to AHB5 does not provide a **buser** signal because the AHB protocol does not have a write response User signal.

2.9 Q-Channels

The AXI5 to AHB5 bridge provides two Q-Channel interfaces. One Q-Channel is intended for clock control and the other Q-Channel for power control.

Both Q-Channels implement the low-power interfaces that the *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces* describes.

The Q-Channels deny quiescent requests when there is ongoing activity or outstanding transfers. Incoming transactions are halted when the bridge is in quiescent state.

A synchronizer is always present on the **pwr_qreqn** input of the power Q-Channel. The presence of a synchronizer on the **clk_qreqn** input of the clock Q-Channel is configurable, by using the `QCLK_SYNC_EN` parameter.

2.10 Register slices

The bridge offers flexibility in the timing of its interfaces with configurable register slices on each slave and master interface.

AXI register slices

Each AXI channel has a register slice. During configuration, you can configure a register slice to operate in one of the following modes:

BYPASS The register slice is bypassed and inserts zero latency.

FORWARD The forward register slice provides timing isolation in only the forward direction, that is, the **valid** signal and the payload of that AXI channel. The register slice inserts 1 **clk** cycle of latency.

REVERSE The reverse register slice provides timing isolation in only the reverse direction, that is, the **ready** signal and the payload of that AXI channel. The register slice inserts a minimum of zero **clk** cycle of latency and a maximum of 1 **clk** cycle of latency.

FULL The full register slice provides timing isolation in the forward and reverse directions. The register slice inserts a minimum of 1 **clk** cycle of latency and a maximum of:

- 2 **clk** cycles of latency on the AR, AW, and W channels.
- 1 **clk** cycle of latency on the R and B channels.

AHB register slices

During configuration, you can add the following register slices:

CNTRL The control register slice provides timing isolation for the address phase and **hwdata** paths.

RDATA This register slice provides timing isolation for **hrdata** and **hready** paths.

The bridge core logic detects an **hready** falling edge one **clk** cycle later than it occurs, so temporary storage is included on the AHB output signals to hold the values during the transient cycle.

Latency calculations for read transfers

If no register slices are enabled, then the bridge can potentially perform read transfers with no added latency. However, due to the pipeline behavior of AHB, there is always 1 cycle of latency.

When there are no added wait states, the minimum and maximum read latencies are:

$read\_latency_{(min)}$ = 1 + lat(AXI_AR) + AHB_REG_CNTRL + AHB_REG_RDATA(addr) + AHB_REG_RDATA(resp) + lat(AXI_R)

= 1 + 0 + 0 + 0 + 0 + 0

= 1 **clk** cycle of latency.

$read\_latency_{(max)}$ = 1 + lat(AXI_AR) + AHB_REG_CNTRL + AHB_REG_RDATA(addr) + AHB_REG_RDATA(resp) + lat(AXI_R)

= 1 + 2 + 1 + 1 + 2 + 1

= 8 **clk** cycles of latency. This latency occurs for a fully registered bridge, that is, the AXI register slices set to FULL and the AHB read data and control register slices enabled.

Wait states occur when the slave is not ready or the read data is delayed.

Latency calculations for write transfers

The AXI protocol requires that the B channel write response must always follow the last W channel write transfer, so there is always 1 **clk** cycle of latency.

The AW channel latency is excluded, because it does not delay the write data channel.

When there are no added wait states, the minimum and maximum write latencies are:

$write\_latency_{(min)}$ = 1 + lat(max(AXI_AW, AXI_W)) + AHB_REG_CNTRL + AHB_REG_WDATA(addr) + AHB_REG_WDATA(resp) + lat(AXI_B)

= 1 + 0 + 0 + 0 + 0 + 0

= 1 **clk** cycle of latency.

$write\_latency_{(max)}$ = 1 + lat(max(AXI_AW, AXI_W)) + AHB_REG_CNTRL + AHB_REG_WDATA(addr) + AHB_REG_WDATA(resp) + lat(AXI_B)

= 1 + 2 + 1 + 1 + 2 + 1

= 8 **clk** cycles of latency. This latency occurs for a fully registered bridge, that is, the AXI register slices set to FULL and the AHB write data and control register slices enabled.

AXI transaction acceptance capabilities

The acceptance capabilities depend on whether the AR or AW channels are registered. The combined acceptance also depends on the setting of the `HWSTRB_ENABLE` parameter.

The following table lists the AXI acceptance capabilities.

Table 2-8 AXI acceptance capabilities

Capability	Value	Notes
Read acceptance.	1 + *AR acceptance*	The value of *AR acceptance* is: 0, when AR channel register slice is in BYPASS. 1, when AR channel register slice is in FORWARD or REVERSE. 2, when AR channel register slice is in FULL.
Write acceptance.	1 + *AW acceptance*	The value of *AW acceptance* is: 0, when AW channel register slice is in BYPASS. 1, when AW channel register slice is in FORWARD or REVERSE. 2, when AW channel register slice is in FULL.
Combined acceptance.	1 + *AR acceptance* + *AW acceptance*	When `HWSTRB_ENABLE == ON`.
	2 + *AR acceptance* + *AW acceptance*	When `HWSTRB_ENABLE == OFF`.

2.11 Read and write transaction scheduling

The bridge contains a transaction scheduler because the AXI5 to AHB5 bridge can receive an AXI read transaction and a write transaction in the same **clk** cycle.

To improve processor performance, the bridge contains arbiter logic that gives priority to read transactions but it also prevents back-to-back read transactions from stalling write transactions indefinitely.

When an incoming read transaction arrives, the arbiter logic can:

- Interrupt a progressing sparse write transaction, for a duration of 7 reads, and then the sparse write can continue.
- Stall an incoming write transaction for a duration of 7 reads, before the bridge starts processing the write transaction. The write transaction can be sparse or non-sparse.

This 7:1 read-write accept policy can prevent a system livelock or starvation issue. The arbitration occurs after an AXI transaction completes or after a sparse write beat.

Note

The arbitration logic is defined from the point of view of the bridge core and not the AXI slave interface. Therefore, the arbitration point is observable externally, when the input stage (AW, AR, W channel) register slices are configured as BYPASS.

Arbitration of a sparse write transaction

The following figure shows the arbitration of a sparse write transaction (**awsparse** is HIGH).

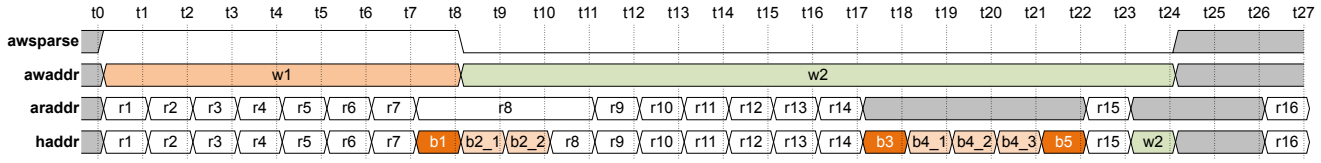

Figure 2-1 Arbitration of a sparse write transaction

In the figure:

- r{num} or w{num} represents the incoming, full AXI transactions.
- b{num} represents a non-sparse AHB write beat, which is shown in orange, for the w1 write transaction.
- b{num}_{piece} represents a sparse AHB write beat, which is shown in light orange, for the w1 write transaction.

In the figure, at time:

t7 After 7 back-to-back reads, the bridge processes the w1 transaction.

t8-t10 The bridge issues a non-sparse write beat, b1, and splits the b2 sparse write beat into the b2_1 and b2_2 transfers.

t10 Arbitration occurs after the sparse beat. The bridge processes read transactions, since reads have a higher priority than writes.

t17 After 7 back-to-back reads, the bridge processes the w1 transaction.

t18-t21 The bridge issues non-sparse write beat, b3, and splits the b4 sparse write beat into the b4_1, b4_2, and b4_3 transfers.

t22 Arbitration occurs after sparse beat b4. The w1 transaction continues because there is no incoming read transaction that has priority.

Write transaction w1 completes with beat b5.

t23 Arbitration occurs, read transaction r15 has priority over the write transaction, w2.

t24 Arbitration occurs, the bridge processes the w2 transaction because there are no incoming read transactions.

t25-t26 The bridge has no incoming transactions.

Arbitration when write transaction is either non-sparse or `HWSTRB_ENABLE` == ON

The bridge does not interrupt any non-sparse write transactions because it might translate into a non-interruptible fixed-length AHB burst such as INCR4. Incoming AXI non-sparse writes always complete before the next arbitration occurs.

The following figure shows the arbitration of a non-sparse write transaction (**awsparse** is LOW). The r{num} and w{num} identifiers represent full AXI transactions.

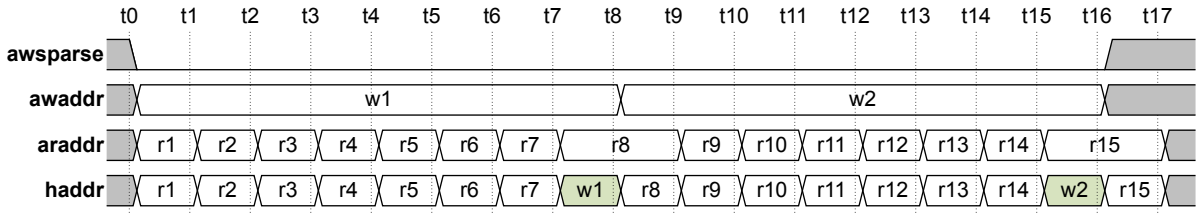

Figure 2-2 Arbitration of a non-sparse write transaction

2.12 Response scheduling, response FIFO

The AXI5 to AHB5 bridge contains AXI response FIFOs for the R and B channels, because the AHB master cannot extend the AHB data phases. After the bridge issues the AHB address phase, the bridge must be able to store the incoming AHB data phase response in the next **clk** cycle, even if the destination AXI channel is unavailable (**rready** or **bready** is LOW). The response FIFO depth depends on the presence of AHB register slices.

The bridge accepts incoming AXI transactions only when the AHB slave interface is ready to accept or provide data (**hready** is HIGH) and the AXI response channel FIFO is ready to accept data. If either of these conditions is not met, the bridge stalls burst propagation by setting the respective **axready**, **wready**, **rvalid**, or **bvalid** signal LOW.

For an AXI write transaction, the bridge must provide a single write response. However, for AHB writes, the bridge receives a write response for each AHB beat. Therefore, the bridge accumulates all the write beat responses, belonging to the burst, and returns the most serious response (SLVERR > OKAY > HEXOKAY).

For an AXI read transaction, the bridge provides the response with each AHB beat. However, when the bridge splits up the first beat of an unaligned access, then the bridge accumulates all the read beat responses, belonging to the burst, and returns the most serious response (SLVERR > OKAY > HEXOKAY).

AXI transaction responses

The **rresp** and **bresp** signals convey the transaction response for the R channel and B channel, respectively. To generate the response, the bridge uses the AHB slave response and the internal bridge error state. An internal bridge error occurs when either:

- The bridge receives an unsupported single, sparse Exclusive write transaction.
- The bridge detects inconsistent use of the **awsparse** signal. For example, if **awsparse** is LOW and a write strobe bit, **wstrb[(DATA_WIDTH/8)−1:0]**, is LOW.

If an internal bridge error occurs, then the bridge still propagates the transaction to the AHB interface and then it issues an SLVERR response on the **bresp** signal.

——— Caution ———

Inconsistent use of the **awsparse** signal might cause memory corruption.

———————————

For AXI transactions that translate to Non-exclusive AHB accesses, the AXI response is either OKAY or SLVERR. The SLVERR response can originate either from the AHB slave or the bridge.

For AXI transactions that translate to Exclusive AHB accesses, the AXI response can be EXOKAY, OKAY (which indicates an Exclusive fail), or SLVERR. In this case, all responses originate from the AHB slave signals **hresp** and **hexokay**.

Chapter 3
Functional Description, AHB5 to AXI5 bridge

This chapter describes the functionality of the AHB5 to AXI5 bridge.

It contains the following sections:

3.1 AMBA bus properties

The AMBA protocols define multiple property types that indicate the capabilities of a device.

The following table lists the AHB5 properties of the AHB5 to AXI5 bridge.

Table 3-1 AHB5 properties

AHB5 property	Value	Comment
Extended_Memory_Types	True	Some extra **hprot** bits that can be translated, are used to generate AXI signals. See *3.3 Protection control translation* on page 3-37.
Secure_Transfers	True	**hnonsec** is copied to **axprot[1]**.
Endian	False	Pass-through. No built-in endian adaptation, it functions as a simple bridge.
Stable_Between_Clock	False	Not supported.
Exclusive_Transfers	True	Exclusive Transfers are translated.
Multi_Copy_Atomicity	True	Pass-through, no buffering. Early write response does not affect this capability.

The following table lists the AXI5 properties of the AHB5 to AXI5 bridge.

Table 3-2 AXI5 properties

AXI5 property	Value	Comment
Ordered_Write_Observation	TRUE	Improved support for the Producer/Consumer ordering model.
Multi_Copy_Atomicity	TRUE	Support for multi-copy atomicity.
Atomic_Transactions	FALSE	-
Check_Type	FALSE	-
Poison	FALSE	-
QoS_Accept	FALSE	AXI4 QoS is supported, but QoS_Accept signaling is not supported.
Trace_Signals	FALSE	-
Loopback_Signals	FALSE	-
Wakeup_Signals	TRUE	Generated from Q-Channel and AHB activity.
Untranslated_Transactions	FALSE	-
NSAccess_Identifiers	TRUE	Supported by using the **hnsaid[3:0]** sideband signals on the AHB5 slave interface.

3.2 Burst translation

For most of the AHB burst types, the AHB5 to AXI5 bridge can perform a simple translation to create the AXI burst. However, the bridge requires extra logic to cope with undefined length bursts and early burst termination.

Undefined length bursts

The bridge converts Modifiable undefined length incremental bursts to transactions of burst length 1 or burst length 4. The `INCR_BURST_CONV` Verilog parameter controls the length of the AXI burst.

If `INCR_BURST_CONV` is set to a burst length of four and the AHB master prematurely stops the burst, then the bridge still completes the four bursts, except:

- For reads, the bridge discards the extra read data.
- For writes, the bridge sets **wstrb** LOW.

Early burst termination

If an AHB read burst is terminated early, the bridge completes the remainder of the burst but it discards the extra read data.

If an AHB write burst is terminated early, the bridge sets the byte strobes to zero and it discards the extra write response. If the discarded write response contains an error, the bridge pulses the **buf_write_error_irq** interrupt signal for 1 **clk** cycle.

Note

- An AHB burst might terminate prematurely because:
  - The AHB master receives an ERROR response so it terminates the burst early.
  - The interconnect terminates the burst early.
  - The bridge promotes an undefined length burst to INCR4 and the ending of the AHB burst does not coincide with the AXI burst length.
- Early burst termination can also cause issues when the bridge is accessing Device memory. Therefore, to prevent the bridge from performing speculative reads of Device memory, the bridge converts Non-modifiable transfers to single AXI transactions.

3.3 Protection control translation

For protection control, the AHB5 to AXI5 bridge maps the **hprot** and **hnonseq** AHB signals to the **axprot** and **axcache** AXI signals. The bridge also maps **hprot[6]** to **axdomain[1:0]**.

The following table shows how the AHB5 to AXI5 bridge generates the AXI5 signals from the AHB5 signals.

Table 3-3 AHB5 to AXI5 protection control mapping

AXI signals	AXI bit number, function	Value is derived from AHB signal:
arprot[2:0] and **awprot[2:0]**	Bit[2], Opcode/Data	!**hprot[0]**
	Bit[1], Secure	**hnonsec**
	Bit[0], Privileged	**hprot[1]**
arcache[3:0]	Bit[3], Other Allocate	Either: • **hprot[4]** for Non-exclusive accesses. • **arcache[3]** is set LOW for Exclusive accesses.
	Bit[2], Allocate	Either: • **hprot[5]** for Non-exclusive accesses. • **arcache[2]** is set LOW for Exclusive accesses.
	Bit[1], Modifiable	**hprot[3]**
	Bit[0], Bufferable	**hprot[2]**
awcache[3:0]	Bit[3], Allocate	Either: • **hprot[5]** for Non-exclusive accesses. • **awcache[3]** is set LOW for Exclusive accesses.
	Bit[2], Other Allocate	Either: • **hprot[4]** for Non-exclusive accesses. • **awcache[2]** is set LOW for Exclusive accesses.
	Bit[1], Modifiable	**hprot[3]**
	Bit[0], Bufferable	**hprot[2]**
ardomain[1:0] and **awdomain[1:0]**	Bits[1:0], shareability domain	**hprot[6]**. The AXI signal is set to: • `0b11` when **hprot[6]** is LOW. • `0b01` when **hprot[6]** is HIGH. **ardomain[1:0]** and **awdomain[1:0]** do not exist in the AXI protocol. The ACE protocol uses a concept that is called shareability domains and the **axdomain[1:0]** signals indicate the shareability domain of a transaction. See the *Arm® AMBA® AXI and ACE Protocol Specification* for more information.

3.4 Response generation

When an AHB slave is idle, it always asserts **hreadyout** and sets **hresp** LOW, which indicates no errors. Therefore, the bridge always accepts a transfer if there is no pending data phase transaction. In the data phase:

- For reads, the bridge asserts **hreadyout** when the read data is available.
- For writes, the bridge asserts **hreadyout** when it accepts the write data.

Error responses

For reads, the bridge transfers any AXI error responses that occur.

For writes, the bridge returns an OKAY response for each AHB beat except for the final beat, when it returns the response from the AXI B channel.

AHB early burst termination

AHB allows early termination of bursts. If a burst is broken on AHB, the bridge still completes the AXI transaction, but it discards read data and masks off write data, and it discards any AXI error responses during the transaction.

If the bridge receives a write error response, then it pulses the **buf_write_error_irq** interrupt signal for 1 **clk** cycle. The bridge generates an interrupt because the error might relate to write beats that occur before the AHB master signals the early termination of a burst.

3.5 Exclusive and locked accesses

The AHB5 to AXI5 bridge can translate AHB5 Exclusive accesses to AXI5 Exclusive accesses. The bridge sets AXI ID to the value of the **hmaster** signal.

For Exclusive accesses, the bridge sets **AxCACHE[3:2]** LOW because the AXI protocol does not permit an Exclusive access to Cacheable memory.

Locked accesses

The bridge does not support locked AHB transfers because the AXI4 and AXI5 protocols do not support locked transactions.

The `AHB_LOCK_RESP` configuration parameter controls how the bridge behaves when it receives a locked AHB transfer. If the bridge receives a locked AHB transfer (**hmastlock** is HIGH), then it either:

- Ignores the lock and forwards the transfer as if **hmastlock** is LOW.
- Blocks the transfer and responds with an ERROR response.

3.6 QoS, region, and NSAID signaling

The AXI protocol supports QoS, region, and NSAID signaling. Although the AHB protocol does not support these signals, the AHB5 to AXI5 bridge has QoS, region, and NSAID signals on the AHB slave interface.

QoS signaling

To support *Quality of Service* (QoS) signaling, the bridge provides the **hqos[3:0]** signal, which is not included in the AHB5 protocol.

The **hqos[3:0]** sideband signal is active during the address phase and the bridge routes its value to the appropriate read or write address channel, on **arqos[3:0]** and **awqos[3:0]**, respectively.

The bridge does not support QoS Accept signaling, which was introduced in the AXI5 protocol.

Region identifier signaling

To support region identifier signaling, the bridge provides the **hregion[3:0]** signal, which is not included in the AHB5 protocol.

The **hregion[3:0]** sideband signal is active during the address phase and the bridge routes its value to the appropriate read or write address channel, on **arregion[3:0]** and **awregion[3:0]**, respectively.

NSAID signaling

To support *Non-secure Access Identifier* (NSAID) signaling, the bridge provides the **hnsaid[3:0]** signal, which is not included in the AHB5 protocol.

The **hnsaid[3:0]** sideband signal is active during the address phase and the bridge routes its value to the appropriate read or write address channel, on **arnsaid[3:0]** and **awnsaid[3:0]**, respectively.

3.7 User signals

The AHB5 to AXI5 bridge supports three AHB User signals and four AXI User signals. The signal width of the address User signals, the read User signals, and the write User signals can be configured to different values.

The following table shows how these signals are mapped between the interfaces.

Table 3-4 User signals mapping

AHB5 slave interface			AXI5 master interface		Comment
Signal	**Direction**	**Phase**	**Signal**	**Direction**	
hauser	Input	Address	**aruser**	Output	Read transactions.
			awuser	Output	Write transactions.
hwuser	Input	Data	**wuser**	Output	For AHB, only valid when **hwdata** is valid. For AXI, only valid when **wdata** is valid.
hruser	Output		**ruser**	Input	Only valid on read transactions.

Note

The AHB5 to AXI5 does not provide a **buser** signal because the AHB protocol does not have a write response User signal.

3.8 Q-Channels

The AHB5 to AXI5 bridge provides two Q-Channel interfaces. One Q-Channel is intended for clock control and the other Q-Channel for power control.

Both Q-Channels implement the low-power interfaces that the *AMBA® Low Power Interface Specification, Arm® Q-Channel and P-Channel Interfaces* describes.

The Q-Channels deny quiescent requests when there is ongoing activity or outstanding transactions. Incoming transfers are halted when the bridge is in quiescent state.

If the bridge receives a wakeup request on the power channel, then it asserts **clk_qactive** because the bridge requires the clock during the wakeup process.

A synchronizer is always present on the **pwr_qreqn** input of the power Q-Channel. The presence of a synchronizer on the **clk_qreqn** input of the clock Q-Channel is configurable, by using the `QCLK_SYNC_EN` parameter.

3.9 Early write response and RAW hazard

An early write response is when the AHB5 to AXI5 bridge provides the AHB5 write response before it receives the AXI write response.

The AHB5 to AXI5 bridge provides an early write response for AHB write transfers that are Bufferable (**hprot[2]** is HIGH) and Non-shareable (**hprot[6]** is LOW).

If the bridge responds OKAY to an AHB buffered write and then later the AXI response is ERROR, the bridge pulses the **buf_write_error_irq** interrupt signal for 1 **clk** cycle, when **irq_en** is HIGH. Setting the interrupt enable, **irq_en**, signal LOW, disables the generation of **buf_write_error_irq** interrupts.

RAW hazard

To avoid *Read After Write* (RAW) hazards, the bridge stores the last 4 AXI write addresses that are waiting for write responses. The bridge stalls any read with an address that matches the same 4K region, until the write response arrives. If there are 4 writes waiting for an AXI write response, the bridge does not provide an early write response for a 5th write until one of the previous 4 writes completes.

3.10 Register slices

The bridge offers flexibility in the timing of its interfaces with configurable register slices on each slave and master interface.

AHB register slices

During configuration, you can add the following register slices:

CNTRL The control register slice provides timing isolation for the address phase paths.

RDATA This register slice provides timing isolation for the **hrdata** path, the response, and the write response.

WDATA This register slice provides timing isolation for the **hwdata** path.

A register slice includes the corresponding User signal, that is, **hauser**, **hruser**, or **hwuser**.

AXI register slices

Each AXI channel can include a register slice. During configuration, you can configure a register slice to operate in one of the following modes:

BYPASS A register slice is not instantiated on the AXI channel.

FORWARD The forward register slice inserts 1 **clk** cycle of latency and provides timing isolation in only the forward direction, that is, the **valid** signal and the payload of that AXI channel.

REVERSE The reverse register slice inserts 0-1 **clk** cycle of latency and provides timing isolation in only the reverse direction, that is, the **ready** signal and the payload of that AXI channel.

FULL The full register slice inserts 1-2 **clk** cycles of latency and provides timing isolation in the forward and reverse directions.

Latency calculations for read transfers

If no register slices are enabled, then the bridge can potentially perform read transfers with no added latency. If all registering is enabled the read latency is 4-6 **clk** cycles, depending on whether the connected slave has **arready** and **rready** HIGH.

When there are no added wait states, the read latency is:

- read_latency = AHB_REG_CNTRL + lat(AXI_AR) + lat(AXI_R) + AHB_REG_RDATA

Wait states occur when the slave is not ready or the read data is delayed.

Latency calculations for write transfers

A write transfer can only have zero latency if the bridge can provide an early write response. Otherwise, the AXI protocol requires that the B channel write response must always follow the last W channel write transfer, which results in one extra **clk** cycle.

The AW channel latency is excluded, because it does not delay the write data channel.

If the bridge can provide an early write response, then for a single beat write the latency is:

- write_latency_ewr = AHB_REG_CNTRL + AHB_REG_WDATA + lat(AXI_W) + AHB_REG_RDATA

If the bridge cannot provide an early write response, the latency is:

- write_latency_noewr_last = AHB_REG_CNTRL + AHB_REG_WDATA + lat(AXI_W) +1 + lat(AXI_B) + AHB_REG_RDATA

If all registering is enabled on both AHB and AXI, then the minimum write latency is 6 **clk** cycles (5 wait states are inserted). The maximum write latency is 8 **clk** cycles.

Chapter 4
Programmers Model

This chapter describes the programmers model.

It contains the following section:

4.1 About the programmers model

The XHB-500 has no programmable registers so the AXI5 to AHB5 bridge and AHB5 to AXI5 bridge are transparent to a programmer.

Appendix A
Signal Descriptions

This appendix describes the interface signals of each XHB-500 bridge.

It contains the following sections:

A.1 AXI5 to AHB5 bridge signals

The bridge has signals for an AXI5 slave interface and an AHB5 master interface. The bridge also has Q-Channel and sideband signals.

The following table lists the clock and reset signals.

Table A-1 Clock and reset signals

Signal	Direction	Description
clk	Input	Clock
resetn	Input	Active-LOW reset. Reset can go LOW asynchronously but must go HIGH synchronously.

The following table lists the AXI5 slave interface signals.

Table A-2 AXI5 slave interface signals

Signal	Direction	Description
AW channel signals:		
awvalid	Input	Write address valid signal.
awaddr[31:0]	Input	Write address signal.
awdomain[1:0]	Input	Used for creating an AHB shareable protection control bit for read transactions.
awburst[1:0]	Input	Write burst type signal.
awid[ID_WIDTH−1:0]	Input	Write request ID signal.
awlen[7:0]	Input	Write burst length signal.
awsize[2:0]	Input	Write burst size signal.
awlock	Input	Write lock type signal.
awprot[2:0]	Input	Write protection type signal.
awready	Output	Write address ready signal.
awcache[3:0]	Input	Indicates how transactions are required to progress through a system.
awregion[3:0]	Input	Permits a single physical interface on a slave to be used for multiple logical interfaces.
awnsaid[3:0]	Input	Provides extra access controls for writes to Non-secure memory locations.
awloop[LB_WIDTH−1:0]	Input	Value to return on the B channel loopback signal, **bloop**.
awqos[3:0]	Input	QoS identifier.
awuser[USER_AX_WIDTH−1:0]	Input	User-defined signal.
AR channel signals:		
arvalid	Input	Read address valid signal.
araddr[31:0]	Input	Read address signal.
ardomain[1:0]	Input	Used for creating an AHB shareable protection control bit for read transactions.
arburst[1:0]	Input	Read burst type signal.
arid[ID_WIDTH−1:0]	Input	Read request ID signal.

Table A-2 AXI5 slave interface signals (continued)

Signal	Direction	Description
arlen[7:0]	Input	Read address burst length signal.
arsize[2:0]	Input	Read burst size signal.
arlock	Input	Read lock type signal.
arprot[2:0]	Input	Read protection type signal.
arready	Output	Read address ready signal.
arcache[3:0]	Input	Indicates how transactions are required to progress through a system.
arregion[3:0]	Input	Permits a single physical interface on a slave to be used for multiple logical interfaces.
arnsaid[3:0]	Input	Provides extra access controls for reads to Non-secure memory locations.
arloop[LB_WIDTH−1:0]	Input	Value to return on the R channel loopback signal, **rloop**.
arqos[3:0]	Input	QoS identifier.
aruser[USER_AX_WIDTH−1:0]	Input	User-defined signal.
W channel signals:		
wvalid	Input	Write data valid signal.
wlast	Input	Indicates last transfer in a write burst.
wstrb[(DATA_WIDTH/8)−1:0]	Input	Write byte lane strobes.
wdata[DATA_WIDTH−1:0]	Input	Write data signal.
wuser[USER_W_WIDTH−1:0]	Input	User-defined signal.
wready	Output	Write data ready signal.
R channel signals:		
rvalid	Output	Read data valid signal.
rid[ID_WIDTH−1:0]	Output	Read data ID.
rlast	Output	Indicates last transfer in read data.
rdata[DATA_WIDTH−1:0]	Output	Read data.
ruser[USER_R_WIDTH−1:0]	Output	User-defined signal.
rresp[1:0]	Output	Read data response.
rready	Input	Read data ready signal.
rloop[LB_WIDTH−1:0]	Output	Return path for the AR channel loopback signal, **arloop**.
B channel signals:		
bvalid	Output	Read data valid signal.
bid[ID_WIDTH−1:0]	Output	Read data ID signal.
bresp[1:0]	Output	Write response signal.
bready	Input	Write response ready signal.
bloop[LB_WIDTH−1:0]	Output	Return path for the AW channel loopback signal, **awloop**.

The following table lists the low-power and sideband signals on the AXI5 slave interface.

Table A-3 AXI5 slave interface low-power signal and sideband signal

Signal	Direction	Description
Low-power signals:		
awakeup	Input	Indicates that the AXI master is initiating activity on this interface.
Sideband signals:		
awsparse	Input	Set this signal HIGH, if an AXI burst might use sparse writes strobes. This signal is not present in the AXI5 protocol.

The following table lists the AHB5 master interface signals.

Table A-4 AHB5 master interface signals

Signal	Direction	Description
hnonsec	Output	Security level, asserted to indicate a Non-secure transfer.
haddr[ADDR_WIDTH−1:0]	Output	Transfer address.
htrans[1:0]	Output	Transfer type.
hsize[2:0]	Output	Transfer size.
hwrite	Output	Write transfer.
hready	Input	Transfer completion indicator.
hprot[6:0]	Output	Protection control.
hburst[2:0]	Output	Transfer burst length.
hmastlock	Output	Locked sequence indicator.
hwdata[DATA_WIDTH−1:0]	Output	Write data.
hexcl	Output	Exclusive Transfer indicator.
hmaster[ID_WIDTH−1:0]	Output	Master identifier.
hrdata[DATA_WIDTH−1:0]	Input	Read data.
hresp	Input	Slave response.
hexokay	Input	Exclusive okay.
hauser[USER_AX_WIDTH−1:0]	Output	Address channel User signals.
hruser[USER_R_WIDTH−1:0]	Input	Read data channel User signals.
hwuser[USER_W_WIDTH−1:0]	Output	Write data channel User signals.

The following table lists the sideband signals on the AHB5 master interface.

Table A-5 AHB5 master interface sideband signals

Signal	Direction	Description
hqos[3:0]	Output	QoS signal
hregion[3:0]	Output	Region identifier signal
hnsaid[3:0]	Output	*Non-secure Access Identifier* (NSAID) signal
hwstrb[(DATA_WIDTH/8)−1:0]	Output	Replicates the **wstrb** content, when the `HWSTRB_ENABLE` parameter is set to ON.

The following table lists the Q-Channel signals.

Table A-6 Q-Channel signals for the AXI5 to AHB5 bridge

Signal	Direction	Description
Clock control Q-Channel signals:		
clk_qreqn	Input	This signal indicates when the controller issues a quiescence entry or exit request to the bridge. The `QCLK_SYNC_EN` parameter controls whether this input includes a 2-stage synchronizer.
clk_qacceptn	Output	This signal indicates when the bridge accepts the quiescence request.
clk_qdeny	Output	This signal indicates when the bridge denies the quiescence request.
clk_qactive	Output	This signal indicates when the bridge is active or it is requesting to exit from quiescence.
Power control Q-Channel signals:		
pwr_qreqn	Input	This signal indicates when the controller issues a quiescence entry or exit request to the bridge. The input contains a 2-stage synchronizer, so the signal can transition asynchronously.
pwr_qacceptn	Output	This signal indicates when the bridge accepts the quiescence request.
pwr_qdeny	Output	This signal indicates when the bridge denies the quiescence request.
pwr_qactive	Output	This signal indicates when the bridge is active or it is requesting to exit from quiescence.

A.2 AHB5 to AXI5 bridge signals

The bridge has signals for an AHB5 slave interface and an AXI5 master interface. The bridge also has Q-Channel, interrupt, and sideband signals.

The following table lists the clock and reset signals.

Table A-7 Clock and reset signals

Signal	Direction	Description
clk	Input	Clock
resetn	Input	Active-LOW reset. Reset can go LOW asynchronously but must go HIGH synchronously.

The following table shows the AHB5 slave interface signals. For more information about the AMBA AHB5 signals, see the *Arm® AMBA® 5 AHB Protocol Specification*.

Table A-8 AHB5 slave interface signals

Signal	Direction	Description
hsel	Input	This signal selects the AHB5 slave interface.
hnonsec	Input	Security level, asserted to indicate a Non-secure transfer.
haddr[ADDR_WIDTH−1:0]	Input	Transfer address.
htrans[1:0]	Input	Transfer type.
hsize[2:0]	Input	Transfer size.
hwrite	Input	Write transfer.
hready	Input	Transfer completion indicator.
hprot[6:0]	Input	Protection control.
hburst[2:0]	Input	Transfer burst length.
hmastlock	Input	Locked sequence indicator.
hwdata[DATA_WIDTH−1:0]	Input	Write data.
hexcl	Input	Exclusive Transfer indicator.
hmaster[ID_WIDTH−1:0]	Input	Master identifier.
hrdata[DATA_WIDTH−1:0]	Output	Read data.
hreadyout	Output	Slave ready.
hresp	Output	Slave response.
hexokay	Output	Exclusive okay.
hauser[USER_AX_WIDTH−1:0]	Input	Address channel User signals.
hruser[USER_R_WIDTH−1:0]	Output	Read data channel User signals.
hwuser[USER_W_WIDTH−1:0]	Input	Write data channel User signals.

The following table shows the sideband signals for the AHB5 slave interface. These signals are not present in the AHB5 protocol.

Table A-9 AHB5 slave interface sideband signals

Signal	Direction	Description
hqos[3:0]	Input	QoS signal
hregion[3:0]	Input	Region identifier signal
hnsaid[3:0]	Input	*Non-secure Access Identifier* (NSAID) signal

The following table shows the interrupt signal, and its enable signal, for the AHB5 to AXI5 bridge.

Table A-10 Interrupt signals for the AHB5 to AXI5 bridge

Signal	Direction	Description
buf_write_error_irq	Output	Active-HIGH pulse interrupt. If the bridge receives an ERROR response for a buffered write, then this signal goes HIGH for 1 **clk** cycle. See *3.9 Early write response and RAW hazard* on page 3-43. If the bridge receives an ERROR response for an early terminated write, then this signal goes HIGH for 1 **clk** cycle. An early terminated write can also occur for a Modifiable undefined length burst that the bridge transfers as one or more 4-beat bursts. See *Undefined length bursts* on page 3-36.
irq_en	Input	Interrupt enable: • HIGH = Enables the **buf_write_error_irq** interrupt. • LOW = Disables the **buf_write_error_irq** interrupt.

The following table shows the AXI5 master interface signals. For more information about the AMBA AXI5 signals, see the *Arm® AMBA® AXI and ACE Protocol Specification*.

Table A-11 AXI5 master interface signals

Signal	Direction	Description
AW channel signals:		
awvalid	Output	Write address valid signal.
awaddr[31:0]	Output	Write address signal.
awdomain[1:0]	Output	Indicates the shareability domain of a write transaction.
awburst[1:0]	Output	Write burst type signal.
awid[ID_WIDTH−1:0]	Output	Write request ID signal.
awlen[7:0]	Output	Write burst length signal.
awsize[2:0]	Output	Write burst size signal.
awlock	Output	Write lock type signal.
awprot[2:0]	Output	Write protection type signal.
awready	Input	Write address ready signal.
awcache[3:0]	Output	Indicates how transactions are required to progress through a system.
awregion[3:0]	Output	Permits a single physical interface on a slave to be used for multiple logical interfaces.
awnsaid[3:0]	Output	Provides extra access controls for writes to Non-secure memory locations.

Table A-11 AXI5 master interface signals (continued)

Signal	Direction	Description
awqos[3:0]	Output	QoS identifier.
awuser[USER_AX_WIDTH−1:0]	Output	Write address channel User signals.
AR channel signals:		
arvalid	Output	Read address valid signal.
araddr[31:0]	Output	Read address signal.
ardomain[1:0]	Output	Indicates the shareability domain of a read transaction.
arburst[1:0]	Output	Read burst type signal.
arid[ID_WIDTH−1:0]	Output	Read request ID signal.
arlen[7:0]	Output	Read address burst length signal.
arsize[2:0]	Output	Read burst size signal.
arlock	Output	Read lock type signal.
arprot[2:0]	Output	Read protection type signal.
arready	Input	Read address ready signal.
arcache[3:0]	Output	Indicates how transactions are required to progress through a system.
arregion[3:0]	Output	Permits a single physical interface on a slave to be used for multiple logical interfaces.
arnsaid[3:0]	Output	Provides extra access controls for reads from Non-secure memory locations.
arqos[3:0]	Output	QoS identifier.
aruser[USER_AX_WIDTH−1:0]	Output	Read address channel User signals.
W channel signals:		
wvalid	Output	Write data valid signal.
wlast	Output	Indicates last transfer in a write burst.
wstrb[(DATA_WIDTH/8)−1:0]	Output	Write byte lane strobes.
wdata[DATA_WIDTH−1:0]	Output	Write data signal.
wuser[USER_W_WIDTH−1:0]	Output	Write channel User signals.
wready	Input	Write data ready signal.
R channel signals:		
rvalid	Input	Read data valid signal.
rid[ID_WIDTH−1:0]	Input	Read data ID.
rlast	Input	Indicates last transfer in read data.
rdata[DATA_WIDTH−1:0]	Input	Read data.
ruser[USER_R_WIDTH−1:0]	Input	Read channel User signals.
rresp[1:0]	Input	Read data response.
rready	Output	Read data ready signal.
B channel signals:		

Table A-11 AXI5 master interface signals (continued)

Signal	Direction	Description
bvalid	Input	Read data valid signal.
bid[ID_WIDTH−1:0]	Input	Read data ID signal.
bresp[1:0]	Input	Write response signal.
bready	Output	Write response ready signal.

The following table lists a low-power signal for the AXI5 master interface.

Table A-12 AXI5 master interface low-power signal

Signal	Direction	Description
awakeup	Output	Indicates that the bridge is processing an AXI transaction. **awakeup** is HIGH when any of the following occur: • **htrans** is not in the IDLE state. • The write buffer is not empty. • The internal *Finite State Machines* (FSMs) are not idle.

The following table lists the Q-Channel signals.

Table A-13 Q-Channel signals for the AHB5 to AXI5 bridge

Signal	Direction	Description
Clock control Q-Channel signals:		
clk_qreqn	Input	This signal indicates when the controller issues a quiescence entry or exit request to the bridge.
clk_qacceptn	Output	This signal indicates when the bridge accepts the quiescence request.
clk_qdeny	Output	This signal indicates when the bridge denies the quiescence request.
clk_qactive	Output	This signal indicates when the bridge is active or it is requesting to exit from quiescence.
Power control Q-Channel signals:		
pwr_qreqn	Input	This signal indicates when the controller issues a quiescence entry or exit request to the bridge. The input contains a 2-stage synchronizer, so the signal can transition asynchronously.
pwr_qacceptn	Output	This signal indicates when the bridge accepts the quiescence request.
pwr_qdeny	Output	This signal indicates when the bridge denies the quiescence request.
pwr_qactive	Output	This signal indicates when the bridge is active or it is requesting to exit from quiescence.

Appendix B
Revisions

This appendix describes the technical changes between released issues of this book.

It contains the following section:

- *B.1 Revisions* on page Appx-B-57.

B.1 Revisions

This appendix describes changes between released issues of this book.

Table B-1 Issue 0000-00

Change	Location	Affects
First release	-	-