
AXI5 to AHB5 bridge and AHB5 to AXI5 bridge

Copyright © 2018, 2019 Arm Limited or its affiliates. All rights reserved.

Release Information

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date</th>
<th>Confidentiality</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000-00</td>
<td>30 November 2018</td>
<td>Confidential</td>
<td>First release for r0p0 BET.</td>
</tr>
<tr>
<td>0000-01</td>
<td>29 March 2019</td>
<td>Non-Confidential</td>
<td>First early access release for r0p0.</td>
</tr>
<tr>
<td>0000-02</td>
<td>15 November 2019</td>
<td>Non-Confidential</td>
<td>First release for r0p0.</td>
</tr>
</tbody>
</table>

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

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

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at [http://www.arm.com/company/policies/trademarks](http://www.arm.com/company/policies/trademarks).

Copyright © 2018, 2019 Arm Limited or its affiliates. All rights reserved.


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 Final, that is for a developed product.

Web Address
www.arm.com
Contents

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

Preface

About this book .................................................................................................................. 7
Feedback ............................................................................................................................ 10

Chapter 1 Introduction

1.1 About the XHB-500 bridges ......................................................................................... 1-12
1.2 Compliance .................................................................................................................. 1-14
1.3 Product documentation ............................................................................................... 1-15
1.4 Product revisions ........................................................................................................ 1-16

Chapter 2 Functional description, AXI5 to AHB5 bridge

2.1 AMBA bus properties ................................................................................................. 2-18
2.2 Burst conversions ...................................................................................................... 2-19
2.3 1KB boundary crossing ............................................................................................. 2-21
2.4 Protection control translation ...................................................................................... 2-22
2.5 Exclusive and locked accesses ................................................................................... 2-23
2.6 Sparse writes ............................................................................................................. 2-25
2.7 Address alignment ...................................................................................................... 2-26
2.8 User signals ............................................................................................................... 2-27
2.9 Q-Channels ................................................................................................................. 2-28
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 on page 7.*
- *Feedback on page 10.*
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 \textit{rm}pn identifier indicates the revision status of the product described in this book, for example, r1p2, where:

\begin{itemize}
  \item \textit{rm} Identifies the major revision of the product, for example, r1.
  \item \textit{pn} Identifies the minor revision or modification status of the product, for example, p2.
\end{itemize}

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:

\textbf{Chapter 1 Introduction}
This chapter introduces the XHB-500 bridges.

\textbf{Chapter 2 Functional description, AXI5 to AHB5 bridge}
This chapter describes the functionality of the AXI5 to AHB5 bridge.

\textbf{Chapter 3 Functional description, AHB5 to AXI5 bridge}
This chapter describes the functionality of the AHB5 to AXI5 bridge.

\textbf{Chapter 4 Programmers model}
This chapter describes the programmers model.

\textbf{Appendix A Signal descriptions}
This appendix describes the interface signals of each XHB-500 bridge.

\textbf{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 \textit{Arm® Glossary} for more information.

Typographic conventions

\begin{itemize}
  \item \textit{italic}
    Introduces special terminology, denotes cross-references, and citations.
  \item \textbf{bold}
    Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
  \item \texttt{monospace}
    Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
  \item \texttt{monospace}
    Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
\end{itemize}
**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.

![Timing Diagram](image)

**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, n 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® Low Power Interface Specification, ARM® Q-Channel and P-Channel Interfaces* (IHI 0068).

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_02_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 on page 1-12.
• 1.2 Compliance on page 1-14.
• 1.3 Product documentation on page 1-15.
• 1.4 Product revisions on page 1-16.
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 bridge is a low-latency bridge that performs no transaction buffering.

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

![AXI5 to AHB5 bridge interfaces](image)

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.
  - 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 bridge is a low-latency bridge.

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

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.
- Arm® 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 on page 2-18.
• 2.2 Burst conversions on page 2-19.
• 2.3 1KB boundary crossing on page 2-21.
• 2.4 Protection control translation on page 2-22.
• 2.5 Exclusive and locked accesses on page 2-23.
• 2.6 Sparse writes on page 2-25.
• 2.7 Address alignment on page 2-26.
• 2.8 User signals on page 2-27.
• 2.9 Q-Channels on page 2-28.
• 2.10 Register slices on page 2-29.
• 2.11 Read and write transaction scheduling on page 2-32.
• 2.12 Response scheduling, response FIFO on page 2-34.
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>
<thead>
<tr>
<th>AXI5 property</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordered_Write_Observation</td>
<td>TRUE</td>
<td>Improved support for the Producer/Consumer ordering model.</td>
</tr>
<tr>
<td>Multi_Copy_Atomicity</td>
<td>TRUE</td>
<td>Support for multi-copy atomicity.</td>
</tr>
<tr>
<td>Atomic_Transactions</td>
<td>FALSE</td>
<td>-</td>
</tr>
<tr>
<td>Check_Type</td>
<td>FALSE</td>
<td>-</td>
</tr>
<tr>
<td>Poison</td>
<td>FALSE</td>
<td>-</td>
</tr>
<tr>
<td>QoS_Accept</td>
<td>FALSE</td>
<td>AXI4 QoS is supported, but QoS_Accept signaling is not supported.</td>
</tr>
<tr>
<td>Trace_Signals</td>
<td>FALSE</td>
<td>-</td>
</tr>
<tr>
<td>Loopback_Signals</td>
<td>TRUE</td>
<td>-</td>
</tr>
<tr>
<td>Wakeup_Signals</td>
<td>TRUE</td>
<td>Q-Channel activity is generated from the \texttt{awakeup} input signal.</td>
</tr>
<tr>
<td>Untranslated_Transactions</td>
<td>FALSE</td>
<td>-</td>
</tr>
<tr>
<td>NSAccess_Identifiers</td>
<td>TRUE</td>
<td>Supported by using the \texttt{hnsaid[3:0]} sideband signals on the AHB5 master interface.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>AHB5 property</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended_Memory_Types</td>
<td>True</td>
<td>\texttt{hprot} value is generated from \texttt{axcache, axdomain,} and \texttt{axprot}.</td>
</tr>
<tr>
<td>Secure_Transfers</td>
<td>True</td>
<td>\texttt{axprot[1]}, the Secure bit, is converted to \texttt{hnonsec}.</td>
</tr>
<tr>
<td>Endian</td>
<td>False</td>
<td>AXI uses a byte-invariant bus, both of its types are directly convertible to AHB. AHB word-invariant endianness is not supported.</td>
</tr>
<tr>
<td>Stable_Between_Clock</td>
<td>False</td>
<td>Not supported.</td>
</tr>
<tr>
<td>Exclusive_Transfers</td>
<td>True</td>
<td>All AHB Exclusive Transfers are supported, but AXI Exclusive bursts are not supported.</td>
</tr>
<tr>
<td>Multi_Copy_Atomicity</td>
<td>True</td>
<td>Pass-through, no buffering.</td>
</tr>
</tbody>
</table>
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 \textit{hburst}, \textit{hsize}, and \textit{htrans} signals.

\textbf{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 bridge 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>
<thead>
<tr>
<th>AXI burst type &amp; length</th>
<th>hburst[2:0]</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIXED, any length</td>
<td>SINGLE</td>
<td>Never crosses the 1KB border, but can be sparse.</td>
</tr>
<tr>
<td>INCR, length 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INCR, length 4</td>
<td>INCR4</td>
<td>The mapping is possible only when all the following conditions apply:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The burst does not cross 1KB border.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For writes, \textit{awsparse} is LOW or \textit{hwstrb} is in use.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A read is not unaligned Non-modifiable.</td>
</tr>
<tr>
<td>WRAP, length 4</td>
<td>WRAP4</td>
<td></td>
</tr>
<tr>
<td>WRAP, length 8</td>
<td>WRAP8</td>
<td></td>
</tr>
<tr>
<td>WRAP, length 16</td>
<td>WRAP16</td>
<td></td>
</tr>
<tr>
<td>Any</td>
<td>INCR</td>
<td>This mapping occurs when any of the following conditions occur:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The incoming AXI burst is of a type and length combination that is not previously listed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The burst crosses a 1KB border.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For writes, \textit{awsparse} is HIGH.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• An AXI WRAP burst that has a length of 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• AXI transaction size is 1024 and AXI burst type is WRAP with length 16.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The read is unaligned Non-modifiable.</td>
</tr>
</tbody>
</table>

If the bridge splits a burst, then it issues multiple INCR bursts.

\textbf{hsize[2:0] mapping}

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

\textbf{htrans[1:0] mapping}

The \textit{htrans[1:0]} signal indicates the transfer type and whether a beat continues or splits a burst. The following table lists \textit{htrans[1:0]} settings for various conditions.
<table>
<thead>
<tr>
<th>Conditions for htrans[1:0] mapping</th>
<th>htrans[1:0]</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>IDLE</td>
<td>-</td>
</tr>
<tr>
<td>An AXI burst is in progress but the relevant AXI channel (W, R, or B) is not ready. No AHB burst is in progress.</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>All AXI beat strobes are LOW.</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>First beat of an AXI transaction.</td>
<td>NONSEQ</td>
<td>-</td>
</tr>
<tr>
<td><strong>hburst</strong> = SINGLE</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>1KB border crossing beat.</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Sparse AXI beat.</td>
<td></td>
<td>NONSEQ for all AHB transfers belonging to the AXI beats.</td>
</tr>
<tr>
<td>Previous AXI beat was sparse.</td>
<td></td>
<td>Wrap-back beat of an AHB-INCR-converted AXI WRAP transaction. See the <em>Any</em> on page 2-19 row in the <strong>hburst[2:0]</strong> mapping table.</td>
</tr>
<tr>
<td>Wrap-back beat.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AHB burst is in progress, but the relevant AXI channel (W, R, or B) is not ready.</td>
<td>BUSY</td>
<td><strong>hburst</strong> != SINGLE</td>
</tr>
<tr>
<td>Any other cases.</td>
<td>SEQ</td>
<td>-</td>
</tr>
</tbody>
</table>
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 \texttt{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 `hnonsec` AHB signals.

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

<table>
<thead>
<tr>
<th>AHB signals</th>
<th>AHB bit number, function</th>
<th>AXI signal mapping</th>
</tr>
</thead>
</table>
| hprot       | Bit[6], Shareable        | `hprot[6]` is derived from `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. |
| hprot       | Bit[5], Allocate         | `hprot[5]` is set to:  
  • `arcache[2]` for read transactions.  
  • `awcache[3]` for write transactions. |
The `hprot[4]` signal is set to:  
  • 0 if `axcache[3:2]` == `0b00`.  
  • 1 if `axcache[3:2]` != `0b00`. |
| hprot       | Bit[2], Bufferable       | `hprot[2]` is derived from `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`). |
| hprot       | Bit[1], Privileged       | `hprot[1]` is set to `axprot[0]`. |
| hprot       | Bit[0], Data/Opcode      | `hprot[0]` is set to `axprot[2]`. |
| hnonsec     | Bit[0], Secure           | `hnonsec` is set to `axprot[1]`. |
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>
<thead>
<tr>
<th>AXI Exclusive access type</th>
<th>AHB transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXI Exclusive read.</td>
<td>Exclusive AHB transfers, for a single AXI transaction.</td>
</tr>
<tr>
<td></td>
<td>Normal AHB transfers, for a burst AXI transaction.</td>
</tr>
<tr>
<td>AXI non-sparse Exclusive write.</td>
<td>Exclusive AHB transfers, for a single AXI transaction.</td>
</tr>
<tr>
<td></td>
<td>Normal AHB transfers, for a burst AXI transaction.</td>
</tr>
<tr>
<td>AXI sparse Exclusive write.</td>
<td>Normal (SLVERR) AHB transfer, for a single AXI transaction.</td>
</tr>
<tr>
<td></td>
<td>Normal AHB transfers, for a burst AXI transaction.</td>
</tr>
</tbody>
</table>

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 Exclusive transfers over AHB-Lite, 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 2-32.

--- Note ---

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

--- Caution ---

Inconsistent use of the awsparse signal might cause memory corruption. See 2.12 Response scheduling, response FIFO on page 2-34 for more information.

---

**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 \texttt{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 bridge 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 (\texttt{awaddr[31:0]}) and \texttt{wstrb} signals must be consistent. Since the first beat is always sparse, the AXI master must drive \texttt{awsparse} HIGH if the bridge is configured with \texttt{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 speculative read 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 data 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 all AXI data transfers to several aligned AHB beats.
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>
<thead>
<tr>
<th>AXI5 slave interface</th>
<th>AHB5 master interface</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal</td>
<td>Direction</td>
<td>Signal</td>
</tr>
<tr>
<td>aruser</td>
<td>Input</td>
<td>hauser</td>
</tr>
<tr>
<td>awuser</td>
<td></td>
<td>hwuser</td>
</tr>
<tr>
<td>wuser</td>
<td>Input</td>
<td>hwuser</td>
</tr>
<tr>
<td>ruser</td>
<td>Output</td>
<td>hruser</td>
</tr>
</tbody>
</table>

Note

The AXI5 to AHB5 bridge 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 *Arm*® *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 has configurable register slices on each slave and master interface, which gives flexibility in the timing of its interfaces.

**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**

When the bridge is idle, and hready and rready are HIGH:

- If no register slices are enabled, the minimum read latency is 1 clk cycle.
- If all registering is enabled, the minimum read latency is 4 clk cycles.

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

\[
\text{read\_latency\_no\_reg} = 1 + \text{REGISTER\_AXI\_AR} + \text{REGISTER\__AHB\_CNTRL} + \text{REGISTER\__AHB\_DATA} \\
= 1 + 0 + 0 + 0 \\
= 1 \text{ clk cycle of latency.}
\]

\[
\text{read\_latency\_all\_reg} = 1 + \text{REGISTER\_AXI\_AR} + \text{REGISTER\__AHB\_CNTRL} + \text{REGISTER\__AHB\_DATA} \\
= 1 + 1 + 1 + 1 \\
= 4 \text{ clk cycles of latency. This latency occurs for a fully registered bridge, that is, the AXI register slices set to FULL and the AHB read data and control register slices enabled.}
\]

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

See *Latency values and the configuration parameters* on page 2-31 for information about the latency value for a given setting of a configuration parameter.
Latency calculations for write transfers
When the bridge is idle, and \texttt{hready} and \texttt{bready} are HIGH:

- If no register slices are enabled, the minimum write latency is 1 \texttt{clk} cycle.
- If all registering is enabled, the minimum write latency is 4 \texttt{clk} cycles.

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

When there are no added wait states, the write latency is calculated as:

\[
\text{write\_latency}_{(\text{no reg})} = 1 + \text{max}(\text{REGISTER\_AXI\_AW, REGISTER\_AXI\_W}) + \text{REGISTER\__AHB\_CNTRL} + \text{REGISTER\__AHB\_DATA}
\]

\[
= 1 + 0 + 0 + 0
\]

\[
= 1 \text{ \texttt{clk} cycle of latency.}
\]

\[
\text{write\_latency}_{(\text{all reg})} = 1 + \text{max}(\text{REGISTER\_AXI\_AW, REGISTER\_AXI\_W}) + \text{REGISTER\__AHB\_CNTRL} + \text{REGISTER\__AHB\_DATA}
\]

\[
= 1 + 1 + 1 + 1
\]

\[
= 4 \text{ \texttt{clk} cycles of latency.}
\]

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.

Additional wait states occur when the master is not ready or the write data is delayed.

See \textit{Latency values and the configuration parameters} on page 2-31 for information about the latency value for a given setting of a configuration parameter.

**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 \texttt{HWSTRB\_ENABLE} parameter. If \texttt{HWSTRB\_ENABLE} is set to \texttt{OFF}, an incoming read can interrupt a write in progress, which then results in an extra transaction being in progress.

The following table lists the AXI acceptance capabilities.

<table>
<thead>
<tr>
<th>Capability</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read acceptance.</td>
<td>(1 + AR) acceptance</td>
<td>The value of \textit{AR acceptance} is:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0, when AR channel register slice is in BYPASS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1, when AR channel register slice is in FORWARD or REVERSE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2, when AR channel register slice is in FULL.</td>
</tr>
<tr>
<td>Write acceptance.</td>
<td>(1 + AW) acceptance</td>
<td>The value of \textit{AW acceptance} is:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0, when AW channel register slice is in BYPASS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1, when AW channel register slice is in FORWARD or REVERSE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2, when AW channel register slice is in FULL.</td>
</tr>
<tr>
<td>Combined acceptance.</td>
<td>(1 + AR) acceptance + (AW) acceptance</td>
<td>When \texttt{HWSTRB_ENABLE} == \texttt{ON}.</td>
</tr>
<tr>
<td></td>
<td>(2 + AR) acceptance + (AW) acceptance</td>
<td>When \texttt{HWSTRB_ENABLE} == \texttt{OFF}.</td>
</tr>
</tbody>
</table>
Latency values and the configuration parameters

The following table lists the AXI latencies for different settings of the REGISTER_AXI_* configuration parameters.

<table>
<thead>
<tr>
<th>AXI configuration parameter</th>
<th>Added minimum latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGISTER_AXI_* == BYPASS</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER_AXI_* == FORWARD</td>
<td>1</td>
</tr>
<tr>
<td>REGISTER_AXI_* == REVERSE</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER_AXI_* == FULL</td>
<td>1</td>
</tr>
</tbody>
</table>

The following table lists the AHB latencies for different settings of the REGISTER_AHB_* configuration parameters.

<table>
<thead>
<tr>
<th>Parameter combinations</th>
<th>Added minimum latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency from AXI transaction input to AHB output - REGISTER_AHB_CNTRL</td>
<td></td>
</tr>
<tr>
<td>REGISTER_AHB_CNTRL == OFF</td>
<td>0</td>
</tr>
<tr>
<td>REGISTER_AHB_CNTRL == ON</td>
<td>1</td>
</tr>
<tr>
<td>Latency from AHB response input to AXI response channel outputs - REGISTER_AHB_DATA</td>
<td></td>
</tr>
<tr>
<td>REGISTER_AHB_RDATA == ON or REGISTER_AXI_R/B == FORWARD or REGISTER_AXI_ R/B == FULL</td>
<td>1</td>
</tr>
<tr>
<td>Any other combination</td>
<td>0</td>
</tr>
</tbody>
</table>
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).

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.

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.

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](image-url)
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 (bready 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.

    ——— Caution ———
    Inconsistent use of the awsparse signal might cause memory corruption.
    ————

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.

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 on page 3-36.
• 3.2 Burst translation on page 3-37.
• 3.3 Protection control translation on page 3-38.
• 3.4 Response generation on page 3-39.
• 3.5 Exclusive and locked accesses on page 3-40.
• 3.6 QoS, region, and NSAID signaling on page 3-41.
• 3.7 User signals on page 3-42.
• 3.8 Q-Channels on page 3-43.
• 3.9 Early write response and RAW hazard on page 3-44.
• 3.10 Register slices on page 3-45.
### 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>
<thead>
<tr>
<th>AHB5 property</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended_Memory_Types</td>
<td>True</td>
<td>Some extra hprot bits that can be translated, are used to generate AXI signals. See 3.3 Protection control translation on page 3-38.</td>
</tr>
<tr>
<td>Secure_Transfers</td>
<td>True</td>
<td>hnonsec is copied to axprot[1].</td>
</tr>
<tr>
<td>Endian</td>
<td>False</td>
<td>Pass-through. No built-in endian adaptation, it functions as a simple bridge.</td>
</tr>
<tr>
<td>Stable_Between_Clock</td>
<td>False</td>
<td>Not supported.</td>
</tr>
<tr>
<td>Exclusive_Transfers</td>
<td>True</td>
<td>Exclusive Transfers are translated.</td>
</tr>
<tr>
<td>Multi_Copy_Atomicity</td>
<td>True</td>
<td>Pass-through, no buffering. Early write response does not affect this capability.</td>
</tr>
</tbody>
</table>

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

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

With the exception of undefined length bursts, the AHB5 to AXI5 bridge can perform a simple translation to create the AXI burst.

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.

Non-modifiable transactions

To avoid speculative transfers, the bridge translates Non-modifiable transactions to single transfers. Arm recommends that you map Non-modifiable transactions to single transfers, especially reads. This is because AHB transactions can be terminated early, but device memory should not be read speculatively by the bridge.

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 will also terminate the AXI burst if the AHB burst ends shorter than speculated due to issuing AHB INCR bursts as AXI INCR4 bursts.
### 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>
<thead>
<tr>
<th>AXI signals</th>
<th>AXI bit number, function</th>
<th>AHB signal mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>arprot[2:0]</code></td>
<td>Bit[2], Opcode/Data</td>
<td><code>axprot[2]</code> is set to !<code>hprot[0]</code></td>
</tr>
<tr>
<td>and <code>awprot[2:0]</code></td>
<td>Bit[1], Secure</td>
<td><code>axprot[1]</code> is set to <code>hnonsec</code></td>
</tr>
<tr>
<td></td>
<td>Bit[0], Privileged</td>
<td><code>axprot[0]</code> is set to <code>hprot[1]</code></td>
</tr>
</tbody>
</table>
| `arcache[3:0]`    | Bit[3], Other Allocate   | `arcache[3]` is set to:
|                   |                          | • LOW for Exclusive accesses.                                                     |
|                   | Bit[2], Allocate        | `arcache[2]` is set to:
|                   |                          | • LOW for Exclusive accesses.                                                    |
|                   | Bit[1], Modifiable      | `arcache[1]` is set to `hprot[3]`                                                |
|                   | Bit[0], Bufferable      | `arcache[0]` is set to `hprot[2]`                                                |
| `awcache[3:0]`    | Bit[3], Allocate        | `awcache[3]` is set to:
|                   |                          | • LOW for Exclusive accesses.                                                     |
|                   | Bit[2], Other Allocate   | `awcache[2]` is set to:
|                   |                          | • LOW for Exclusive accesses.                                                    |
|                   | Bit[1], Modifiable      | `awcache[1]` is set to `hprot[3]`                                                |
|                   | Bit[0], Bufferable      | `awcache[0]` is set to `hprot[2]`                                                |
| `ardomain[1:0]`   | Bits[1:0], shareability domain | `axdomain` derived from `hprot[6]`                              |
| and `awdomain[1:0]`|                          | `axdomain` is set to:
|                   |                          | • 0b11 when `hprot[6]` is LOW.                                                  |
|                   |                          | • 0b02 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 \texttt{hreadyout} and sets \texttt{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 \texttt{hreadyout} when the AXI read data is available.
- For writes, the bridge asserts \texttt{hreadyout} when the AXI write data is accepted.

\textbf{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.

\textbf{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 sets write data to zero with write strobes LOW, and it discards any AXI error responses during the transaction.

If the bridge receives a write error response, then it pulses the \texttt{buf_write_error_irq} interrupt signal for 1 \texttt{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.

\textbf{Write buffering}

The bridge contains a write buffer that allows it to accept write data without having to wait for the AXI W channel. This improves performance, especially when the AHB response is registered. The depth of the buffer is 1-3 beats depending on the configuration.
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 was set 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>
<thead>
<tr>
<th>AHB5 slave interface</th>
<th>AXI5 master interface</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal</td>
<td>Direction</td>
<td>Phase</td>
</tr>
<tr>
<td>hauser</td>
<td>Input</td>
<td>Address</td>
</tr>
<tr>
<td>hwuser</td>
<td>Input</td>
<td>Data</td>
</tr>
<tr>
<td>hruser</td>
<td>Output</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AXI has a single address phase for each burst. This means that the hauser signal is only used when an AXI burst is started. For fixed length bursts, the hauser signal is not transferred to AXI for beats after the first one. For undefined length AHB INCR bursts, the first and every fourth consequent beat’s hauser is transferred if the bursts are converted to AXI transfers with a length of 4. If you want to transfer each beat’s hauser in a burst, set hprot[3] to LOW. This marks the transfer as unmodifiable, which causes the bridge to transfer it as a series of single transfers.

**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 *Arm® 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. The bridge buffers the address phase internally and stalls the response until the bridge wakes up and the AXI transfer starts.

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 two-stage synchronizer is always present on the `pwr_qreqn` input of the power Q-Channel. The presence of a two-stage 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 fifth write until one of the previous 4 writes completes.
3.10 Register slices

The bridge has configurable register slices on each slave and master interface, which gives flexibility in the timing of its interfaces.

**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 read data path and the shared read/write response signals (hreadyout, hresp, hexokay).
- **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 between 4 and 6 clk cycles. It is 4 cycles if the connected slave has arready and rready HIGH. It is 5 cycles if one of these signals is LOW and 6 cycles if both are LOW.

Excluding any wait states added by the slave, the read latency is:

\[
\text{read latency} = \text{REGISTER_AHB_CNTRL} + \text{lat(AXI_AR)} + \text{lat(AXI_R)} + \text{REGISTER_AHB_RDATA}
\]

The REGISTER_AHB_CNTRL and REGISTER_AHB_RDATA are configuration parameters that are set to OFF or ON. When a parameter is set to OFF the latency value is zero, and the latency value is one when set to ON.

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

**Latency calculations for write transfers**

If no register slices are enabled, then the bridge has a minimum write latency of zero cycles when it provides an early write response, otherwise there is 1 clk cycle of latency. If all registering is enabled on AHB and AXI, the write latency is 6-8 clk cycles.

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:

\[ \text{write\_latency\_ewr} = \text{REGISTER\_AHB\_CNTRL} + \text{REGISTER\_AHB\_WDATA} + \text{lat(AXI\_W)} + \text{REGISTER\_AHB\_RDATA} \]

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

\[ \text{write\_latency\_noewr\_last} = \text{REGISTER\_AHB\_CNTRL} + \text{REGISTER\_AHB\_WDATA} + \text{lat(AXI\_W)} + 1 + \text{lat(AXI\_B)} + \text{REGISTER\_AHB\_RDATA} \]

The REGISTER\_AHB\_CNTRL, REGISTER\_AHB\_WDATA, and REGISTER\_AHB\_RDATA are configuration parameters that are set to OFF or ON. When a parameter is set to OFF the latency value is zero, and the latency value is one when set to ON.

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 on page 4-48.
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 on page Appx-A-50.
• A.2 AHB5 to AXI5 bridge signals on page Appx-A-54.
### 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>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>Input</td>
<td>Clock</td>
</tr>
<tr>
<td>resetn</td>
<td>Input</td>
<td>Active-LOW reset. Reset can go LOW asynchronously but must go HIGH synchronously.</td>
</tr>
</tbody>
</table>

The following table lists the AXI5 slave interface signals.

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

AR channel signals:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arvalid</td>
<td>Input</td>
<td>Read address valid signal.</td>
</tr>
<tr>
<td>araddr[31:0]</td>
<td>Input</td>
<td>Read address signal.</td>
</tr>
<tr>
<td>ardomain[1:0]</td>
<td>Input</td>
<td>Used for creating an AHB shareable protection control bit for read transactions.</td>
</tr>
<tr>
<td>arbust[1:0]</td>
<td>Input</td>
<td>Read burst type signal.</td>
</tr>
<tr>
<td>arid[ID_WIDTH–1:0]</td>
<td>Input</td>
<td>Read request ID signal.</td>
</tr>
<tr>
<td>Signal</td>
<td>Direction</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>arlen[7:0]</td>
<td>Input</td>
<td>Read address burst length signal.</td>
</tr>
<tr>
<td>arsize[2:0]</td>
<td>Input</td>
<td>Read burst size signal.</td>
</tr>
<tr>
<td>arlock</td>
<td>Input</td>
<td>Read lock type signal.</td>
</tr>
<tr>
<td>arprot[2:0]</td>
<td>Input</td>
<td>Read protection type signal.</td>
</tr>
<tr>
<td>arready</td>
<td>Output</td>
<td>Read address ready signal.</td>
</tr>
<tr>
<td>arcache[3:0]</td>
<td>Input</td>
<td>Indicates how transactions are required to progress through a system.</td>
</tr>
<tr>
<td>arregion[3:0]</td>
<td>Input</td>
<td>Permits a single physical interface on a slave to be used for multiple logical interfaces.</td>
</tr>
<tr>
<td>arnsaid[3:0]</td>
<td>Input</td>
<td>Provides extra access controls for reads to Non-secure memory locations.</td>
</tr>
<tr>
<td>arloop[LB_WIDTH−1:0]</td>
<td>Input</td>
<td>Value to return on the R channel loopback signal, rloop.</td>
</tr>
<tr>
<td>arqos[3:0]</td>
<td>Input</td>
<td>QoS identifier.</td>
</tr>
<tr>
<td>aruser[USER_AX_WIDTH−1:0]</td>
<td>Input</td>
<td>User-defined signal.</td>
</tr>
<tr>
<td>wvalid</td>
<td>Input</td>
<td>Write data valid signal.</td>
</tr>
<tr>
<td>wlast</td>
<td>Input</td>
<td>Indicates last transfer in a write burst.</td>
</tr>
<tr>
<td>wstrb[(DATA_WIDTH/8)−1:0]</td>
<td>Input</td>
<td>Write byte lane strobes.</td>
</tr>
<tr>
<td>wdata[DATA_WIDTH−1:0]</td>
<td>Input</td>
<td>Write data signal.</td>
</tr>
<tr>
<td>wuser[USER_W_WIDTH−1:0]</td>
<td>Input</td>
<td>User-defined signal.</td>
</tr>
<tr>
<td>wready</td>
<td>Output</td>
<td>Write data ready signal.</td>
</tr>
<tr>
<td>rvalid</td>
<td>Output</td>
<td>Read data valid signal.</td>
</tr>
<tr>
<td>rid[ID_WIDTH−1:0]</td>
<td>Output</td>
<td>Read data ID.</td>
</tr>
<tr>
<td>rlast</td>
<td>Output</td>
<td>Indicates last transfer in read data.</td>
</tr>
<tr>
<td>rdata[DATA_WIDTH−1:0]</td>
<td>Output</td>
<td>Read data.</td>
</tr>
<tr>
<td>ruser[USER_R_WIDTH−1:0]</td>
<td>Output</td>
<td>User-defined signal.</td>
</tr>
<tr>
<td>rresp[1:0]</td>
<td>Output</td>
<td>Read data response.</td>
</tr>
<tr>
<td>rready</td>
<td>Input</td>
<td>Read data ready signal.</td>
</tr>
<tr>
<td>rloop[LB_WIDTH−1:0]</td>
<td>Output</td>
<td>Return path for the AR channel loopback signal, arloop.</td>
</tr>
<tr>
<td>bvalid</td>
<td>Output</td>
<td>Write response valid signal.</td>
</tr>
<tr>
<td>bid[ID_WIDTH−1:0]</td>
<td>Output</td>
<td>Write response ID signal.</td>
</tr>
<tr>
<td>bresp[1:0]</td>
<td>Output</td>
<td>Write response signal.</td>
</tr>
<tr>
<td>bready</td>
<td>Input</td>
<td>Write response ready signal.</td>
</tr>
<tr>
<td>bloop[LB_WIDTH−1:0]</td>
<td>Output</td>
<td>Return path for the AW channel loopback signal, awloop.</td>
</tr>
</tbody>
</table>

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

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

The following table lists the AHB5 master interface signals.

Table A-4  AHB5 master interface signals

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

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

Table A-5  AHB5 master interface sideband signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hqos[3:0]</td>
<td>Output</td>
<td>QoS signal. This signal has the same timing requirements as AHB address phase signals, and is constant throughout a burst.</td>
</tr>
<tr>
<td>hregion[3:0]</td>
<td>Output</td>
<td>Region identifier signal. This signal has the same timing requirements as AHB address phase signals, and is constant throughout a burst.</td>
</tr>
</tbody>
</table>
### Table A-5  AHB5 master interface sideband signals (continued)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hnsaid[3:0]</td>
<td>Output</td>
<td>Non-secure Access Identifier (NSAID) signal. This signal has the same timing requirements as AHB address phase signals, and is constant throughout a burst.</td>
</tr>
<tr>
<td>hwstrb[(DATA_WIDTH/8)−1:0]</td>
<td>Output</td>
<td>Replicates the wstrb content, when the HWSTRB_ENABLE parameter is set to ON.</td>
</tr>
</tbody>
</table>

The following table lists the Q-Channel signals.

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

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clock control Q-Channel signals:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clk_qreqn</td>
<td>Input</td>
<td>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.</td>
</tr>
<tr>
<td>clk_qacceptn</td>
<td>Output</td>
<td>This signal indicates when the bridge accepts the quiescence request.</td>
</tr>
<tr>
<td>clk_qdeny</td>
<td>Output</td>
<td>This signal indicates when the bridge denies the quiescence request.</td>
</tr>
<tr>
<td>clk_qactive</td>
<td>Output</td>
<td>This signal indicates when the bridge is active or it is requesting to exit from quiescence.</td>
</tr>
<tr>
<td><strong>Power control Q-Channel signals:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pwr_qreqn</td>
<td>Input</td>
<td>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.</td>
</tr>
<tr>
<td>pwr_qacceptn</td>
<td>Output</td>
<td>This signal indicates when the bridge accepts the quiescence request.</td>
</tr>
<tr>
<td>pwr_qdeny</td>
<td>Output</td>
<td>This signal indicates when the bridge denies the quiescence request.</td>
</tr>
<tr>
<td>pwr_qactive</td>
<td>Output</td>
<td>This signal indicates when the bridge is active or it is requesting to exit from quiescence.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>Input</td>
<td>Clock</td>
</tr>
<tr>
<td>resetn</td>
<td>Input</td>
<td>Active-LOW reset. Reset can go LOW asynchronously but must go HIGH synchronously.</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsel</td>
<td>Input</td>
<td>This signal selects the AHB5 slave interface.</td>
</tr>
<tr>
<td>hnonsec</td>
<td>Input</td>
<td>Security level, asserted to indicate a Non-secure transfer.</td>
</tr>
<tr>
<td>haddr[ADDR_WIDTH−1:0]</td>
<td>Input</td>
<td>Transfer address.</td>
</tr>
<tr>
<td>htrans[1:0]</td>
<td>Input</td>
<td>Transfer type.</td>
</tr>
<tr>
<td>hsize[2:0]</td>
<td>Input</td>
<td>Transfer size.</td>
</tr>
<tr>
<td>hwdata[DATA_WIDTH−1:0]</td>
<td>Input</td>
<td>Write data.</td>
</tr>
<tr>
<td>hexcl</td>
<td>Input</td>
<td>Exclusive Transfer indicator.</td>
</tr>
<tr>
<td>hmaster[ID_WIDTH−1:0]</td>
<td>Input</td>
<td>Master identifier.</td>
</tr>
<tr>
<td>hrdata[DATA_WIDTH−1:0]</td>
<td>Output</td>
<td>Read data.</td>
</tr>
<tr>
<td>hreadyout</td>
<td>Output</td>
<td>Slave ready.</td>
</tr>
<tr>
<td>hresp</td>
<td>Output</td>
<td>Slave response.</td>
</tr>
<tr>
<td>hexokay</td>
<td>Output</td>
<td>Exclusive okay.</td>
</tr>
<tr>
<td>hauser[USER_AX_WIDTH−1:0]</td>
<td>Input</td>
<td>Address channel User signals.</td>
</tr>
<tr>
<td>hruser[USER_R_WIDTH−1:0]</td>
<td>Output</td>
<td>Read data channel User signals.</td>
</tr>
<tr>
<td>hwuser[USER_W_WIDTH−1:0]</td>
<td>Input</td>
<td>Write data channel User signals.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hqos[3:0]</td>
<td>Input</td>
<td>QoS signal. This signal has the same timing requirements as AHB address phase signals, and must be constant throughout a burst.</td>
</tr>
<tr>
<td>hregion[3:0]</td>
<td>Input</td>
<td>Region identifier signal. This signal has the same timing requirements as AHB address phase signals, and must be constant throughout a burst.</td>
</tr>
<tr>
<td>hnsaid[3:0]</td>
<td>Input</td>
<td>Non-secure Access Identifier (NSAID) signal. This signal has the same timing requirements as AHB address phase signals, and must be constant throughout a burst.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buf_write_error_irq</td>
<td>Output</td>
<td>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 3-44. 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 3-37.</td>
</tr>
</tbody>
</table>
| 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

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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. |
### Table A-11 AXI5 master interface signals (continued)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>awregion[3:0]</td>
<td>Output</td>
<td>Permits a single physical interface on a slave to be used for multiple logical interfaces.</td>
</tr>
<tr>
<td>awnsaid[3:0]</td>
<td>Output</td>
<td>Provides extra access controls for writes to Non-secure memory locations.</td>
</tr>
<tr>
<td>awuser[USER_AX_WIDTH−1:0]</td>
<td>Output</td>
<td>Write address channel User signals.</td>
</tr>
</tbody>
</table>

**AR channel signals:**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arvalid</td>
<td>Output</td>
<td>Read address valid signal.</td>
</tr>
<tr>
<td>araddr[31:0]</td>
<td>Output</td>
<td>Read address signal.</td>
</tr>
<tr>
<td>ardomain[1:0]</td>
<td>Output</td>
<td>Indicates the shareability domain of a read transaction.</td>
</tr>
<tr>
<td>arburst[1:0]</td>
<td>Output</td>
<td>Read burst type signal.</td>
</tr>
<tr>
<td>arid[ID_WIDTH−1:0]</td>
<td>Output</td>
<td>Read request ID signal.</td>
</tr>
<tr>
<td>arlen[7:0]</td>
<td>Output</td>
<td>Read address burst length signal.</td>
</tr>
<tr>
<td>arsize[2:0]</td>
<td>Output</td>
<td>Read burst size signal.</td>
</tr>
<tr>
<td>arlock</td>
<td>Output</td>
<td>Read lock type signal.</td>
</tr>
<tr>
<td>arprot[2:0]</td>
<td>Output</td>
<td>Read protection type signal.</td>
</tr>
<tr>
<td>arready</td>
<td>Input</td>
<td>Read address ready signal</td>
</tr>
<tr>
<td>arcache[3:0]</td>
<td>Output</td>
<td>Indicates how transactions are required to progress through a system.</td>
</tr>
<tr>
<td>arregion[3:0]</td>
<td>Output</td>
<td>Permits a single physical interface on a slave to be used for multiple logical interfaces.</td>
</tr>
<tr>
<td>arnsaid[3:0]</td>
<td>Output</td>
<td>Provides extra access controls for reads from Non-secure memory locations.</td>
</tr>
<tr>
<td>aruser[USER_AX_WIDTH−1:0]</td>
<td>Output</td>
<td>Read address channel User signals.</td>
</tr>
</tbody>
</table>

**W channel signals:**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wvalid</td>
<td>Output</td>
<td>Write data valid signal.</td>
</tr>
<tr>
<td>wlast</td>
<td>Output</td>
<td>Indicates last transfer in a write burst.</td>
</tr>
<tr>
<td>wstrb[(DATA_WIDTH/8)−1:0]</td>
<td>Output</td>
<td>Write byte lane strobes.</td>
</tr>
<tr>
<td>wdata[DATA_WIDTH−1:0]</td>
<td>Output</td>
<td>Write data signal.</td>
</tr>
<tr>
<td>wuser[USER_W_WIDTH−1:0]</td>
<td>Output</td>
<td>Write channel User signals.</td>
</tr>
<tr>
<td>wready</td>
<td>Input</td>
<td>Write data ready signal.</td>
</tr>
</tbody>
</table>

**R channel signals:**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rvalid</td>
<td>Input</td>
<td>Read data valid signal.</td>
</tr>
<tr>
<td>rid[ID_WIDTH−1:0]</td>
<td>Input</td>
<td>Read data ID.</td>
</tr>
<tr>
<td>rlast</td>
<td>Input</td>
<td>Indicates last transfer in read data.</td>
</tr>
<tr>
<td>rdata[DATA_WIDTH−1:0]</td>
<td>Input</td>
<td>Read data.</td>
</tr>
<tr>
<td>ruser[USER_R_WIDTH−1:0]</td>
<td>Input</td>
<td>Read channel User signals.</td>
</tr>
</tbody>
</table>
Table A-11  AXI5 master interface signals (continued)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rresp[1:0]</td>
<td>Input</td>
<td>Read data response.</td>
</tr>
<tr>
<td>rready</td>
<td>Output</td>
<td>Read data ready signal.</td>
</tr>
</tbody>
</table>

B channel signals:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bvalid</td>
<td>Input</td>
<td>Write response valid signal.</td>
</tr>
<tr>
<td>bid[ID_WIDTH−1:0]</td>
<td>Input</td>
<td>Write response ID signal.</td>
</tr>
<tr>
<td>bresp[1:0]</td>
<td>Input</td>
<td>Write response signal.</td>
</tr>
<tr>
<td>bready</td>
<td>Output</td>
<td>Write response ready signal.</td>
</tr>
</tbody>
</table>

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

Table A-12  AXI5 master interface low-power signal

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
</table>
| awake     | Output    | Indicates that the bridge is processing an AXI transaction. \textit{awake} is HIGH when any of the following occur:  
• \textit{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

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock control Q-Channel signals:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clk_qreqn</td>
<td>Input</td>
<td>This signal indicates when the controller issues a quiescence entry or exit request to the bridge.</td>
</tr>
<tr>
<td>clk_qacceptn</td>
<td>Output</td>
<td>This signal indicates when the bridge accepts the quiescence request.</td>
</tr>
<tr>
<td>clk_qdeny</td>
<td>Output</td>
<td>This signal indicates when the bridge denies the quiescence request.</td>
</tr>
<tr>
<td>clk_qactive</td>
<td>Output</td>
<td>This signal indicates when the bridge is active or it is requesting to exit from quiescence.</td>
</tr>
</tbody>
</table>

Power control Q-Channel signals:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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-59.
B.1 Revisions

This appendix describes changes between released issues of this book.

Table B-1 Issue 0000-00

<table>
<thead>
<tr>
<th>Change</th>
<th>Location</th>
<th>Affects</th>
</tr>
</thead>
<tbody>
<tr>
<td>First release</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table B-2 Differences between issue 0000-00 and issue 0000-01

<table>
<thead>
<tr>
<th>Change</th>
<th>Location</th>
<th>Affects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added caution about possible memory corruption.</td>
<td>2.6 Sparse writes on page 2-25</td>
<td>All revisions</td>
</tr>
<tr>
<td>Updated the latency calculations.</td>
<td>Latency calculations for read transfers on page 2-29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Latency calculations for write transfers on page 2-30</td>
<td></td>
</tr>
<tr>
<td>Added content.</td>
<td>Non-modifiable transactions on page 3-37</td>
<td></td>
</tr>
<tr>
<td>Added content.</td>
<td>Write buffering on page 3-39</td>
<td></td>
</tr>
<tr>
<td>Added content about the behavior during quiescence.</td>
<td>3.8 Q-Channels on page 3-43</td>
<td></td>
</tr>
<tr>
<td>Updated the latency calculations.</td>
<td>Latency calculations for read transfers on page 3-45</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Latency calculations for write transfers on page 3-45</td>
<td></td>
</tr>
<tr>
<td>Corrected the description for \texttt{bvalid} and \texttt{bid}.</td>
<td>A.1 AXI5 to AHB5 bridge signals on page Appx-A-50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A.2 AHB5 to AXI5 bridge signals on page Appx-A-54</td>
<td></td>
</tr>
</tbody>
</table>

Table B-3 Differences between issue 0000-01 and 0000-02

<table>
<thead>
<tr>
<th>Change</th>
<th>Location</th>
<th>Affects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editorial changes in various sections. No actual content difference.</td>
<td>-</td>
<td>All revisions</td>
</tr>
</tbody>
</table>