DesignWare® DW_axi_a2x

Databook

DW_axi_a2x – ***Product Code***

2.04a
March 2020

Contents

Revision History

This section tracks the significant documentation changes that occur from release-to-release and during a release.

Version	Date	Description
2.04a	March 2020	■ Revision version change for 2020.03a release ■ Updated synthesis results in "Performance" on page 153 ■ Updated "Clocks and Resets" on page 71 ■ Updated "Synchronizers used in DW_axi_a2x" on page 162 ■ Renamed Clocking section to "Clocks and Resets" on page 71 ■ Renamed Synchronizers chapter to Appendix A, "Basic Core Module (BCM) Library" ■ "Signal Descriptions" on page 101, "Parameter Descriptions" on page 79, "Internal Parameter Descriptions" on page 149 auto-extracted from the RTL
2.03a	February 2018	■ Revision version change for 2018.02a release ■ Updated synthesis results in "Performance" on page 153 ■ Removed Chapter 2 Building and Verifying a Component or Subsystem from the databook and added the contents in the newly created user guide ■ "Signal Descriptions" on page 101, "Parameter Descriptions" on page 79, "Internal Parameter Descriptions" on page 149 auto-extracted from the RTL with change bars
2.02a	March 2016	■ Revision version change for 2016.03a release ■ Added "Running SpyGlass® Lint and SpyGlass® CDC" section ■ Added "Running Spyglass on Generated Code with coreAssembler" section ■ "Signal Descriptions" on page 101 and "Parameter Descriptions" on page 79 auto-extracted from the RTL ■ Added "Internal Parameter Descriptions" on page 149 ■ Added 1024 bits as a configurable data bus width in "General DW_axi_a2x Features" on page 15 ■ Added Appendix A, "Basic Core Module (BCM) Library" ■ Updated area and power numbers in "Performance" on page 153
1.01a	May 2013	Version change for 2013.05a release. Updated the template.
1.00a	Jan 2012	Initial version

Preface

This databook provides information that you need to interface the DW_axi_a2x component to the Synopsys DesignWare Synthesizable Components for AMBA environment. This component conforms to the AMBA 3 AXI and AMBA 4 AXI specifications defined in the *AMBA AXI and ACE Protocol Specification* from ARM.

The information in this databook includes a functional description, and signal and parameter descriptions, as well as information on how you can configure, create RTL for, simulate, and synthesize the component using coreConsultant. Also provided are an overview of the component testbench, a description of the tests that are run to verify the coreKit, and synthesis information for the coreKit.

Organization

The chapters of this databook are organized as follows:

- Chapter 1, "Product Overview" provides a system overview, a component block diagram, basic features, and an overview of the verification environment.
- Chapter 2, "Functional Description" describes the functional operation of the DW_axi_a2x.
- Chapter 3, "Parameter Descriptions" identifies the configurable parameters supported by the DW_axi_a2x.
- Chapter 4, "Signal Descriptions" provides a list and description of the DW_axi_a2x signals.
- Chapter 5, "Internal Parameter Descriptions" provides a list of internal parameter descriptions that might be indirectly referenced in expressions in the Signals and Parameters chapters.
- Chapter 6, "Integration Considerations" includes information you need to integrate the configured DW_axi_a2x into your design.
- Chapter 7, "Verification" provides information on verifying the configured DW_axi_a2x.
- Appendix A, "Basic Core Module (BCM) Library" documents the synchronizer methods (blocks of synchronizer functionality) used in DW_axi_a2x to cross clock boundaries.
- Appendix B, "Glossary" provides a glossary of general terms.

Related Documentation

- *Using DesignWare Library IP in coreAssembler* – Contains information on getting started with using DesignWare SIP components for AMBA 2 and AMBA 3 AXI/AMBA 4 AXI components within coreTools
- *coreAssembler User Guide* – Contains information on using coreAssembler

- *coreConsultant User Guide* – Contains information on using coreConsultant

To see a complete listing of documentation within the DesignWare Synthesizable Components for AMBA, refer to the *Guide to Documentation for DesignWare Synthesizable Components for AMBA 2, AMBA 3 AXI, and AMBA 4 AXI.*

Web Resources

- DesignWare IP product information: https://www.synopsys.com/designware-ip.html
- Your custom DesignWare IP page: https://www.synopsys.com/dw/mydesignware.php
- Documentation through SolvNetPlus: https://solvnetplus.synopsys.com (Synopsys password required)
- Synopsys Common Licensing (SCL): https://www.synopsys.com/keys

Customer Support

Synopsys provides the following various methods for contacting Customer Support:

- Prepare the following debug information, if applicable:
  - For environment set-up problems or failures with configuration, simulation, or synthesis that occur within coreConsultant or coreAssembler, select the following menu:

    File > Build Debug Tar-file

    Check all the boxes in the dialog box that apply to your issue. This option gathers all the Synopsys product data needed to begin debugging an issue and writes it to the <*core tool startup directory*>/debug.tar.gz file.
  - For simulation issues outside of coreConsultant or coreAssembler:
    - Create a waveforms file (such as VPD or VCD).
    - Identify the hierarchy path to the DesignWare instance.
    - Identify the timestamp of any signals or locations in the waveforms that are not understood.
- *For the fastest response*, enter a case through SolvNetPlus:

a. https://solvnetplus.synopsys.com

Note SolvNetPlus does not support Internet Explorer. Use a supported browser such as Microsoft Edge, Google Chrome, Mozilla Firefox, or Apple Safari.

b. Click the **Cases** menu and then click **Create a New Case** (below the list of cases).

c. Complete the mandatory fields that are marked with an asterisk and click **Save**.

Ensure to include the following:

- **Product L1:** DesignWare Library IP
- **Product L2:** <name of L2>

d. After creating the case, attach any debug files you created.

For more information about general usage information, refer to the following article in SolvNetPlus:

https://solvnetplus.synopsys.com/s/article/SolvNetPlus-Usage-Help-Resources

- Or, send an e-mail message to support_center@synopsys.com (your email will be queued and then, on a first-come, first-served basis, manually routed to the correct support engineer):
  - Include the Product L1 and Product L2 names, and Version number in your e-mail so it can be routed correctly.
  - For simulation issues, include the timestamp of any signals or locations in waveforms that are not understood
  - Attach any debug files you created.
- Or, telephone your local support center:
  - North America:
    Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
  - All other countries:
    https://www.synopsys.com/support/global-support-centers.html

Product Code

Table 1-1 lists all the components associated with the product code for DesignWare AMBA Fabric.

Table 1-1 DesignWare AMBA Fabric – Product Code: 3768-0

Component Name	Description
DW_ahb	High performance, low latency interconnect fabric for AMBA 2 AHB
DW_ahb_eh2h	High performance, high bandwidth AMBA 2 AHB to AHB bridge
DW_ahb_h2h	Area efficient, low bandwidth AMBA 2 AHB to AHB Bridge
DW_ahb_icm	Configurable multi-layer interconnection matrix
DW_ahb_ictl	Configurable vectored interrupt controllers for AHB bus systems
DW_apb	High performance, low latency interconnect fabric & bridge for AMBA 2 APB for direct connect to AMBA 2 AHB fabric
DW_apb_ictl	Configurable vectored interrupt controllers for APB bus systems
DW_axi	High performance, low latency interconnect fabric for AMBA 3 AXI and AMBA 4 AXI
DW_axi_a2x	Configurable bridge between AMBA 3 AXI/AMBA 4 AXI components and AHB components or between AMBA 3 AXI/AMBA 4 AXI components and AMBA 3 AXI/AMBA 4 AXI components.
DW_axi_gm	Simplify the connection of third party/custom master controllers to any AMBA 3 AXI or AMBA 4 AXI fabric
DW_axi_gs	Simplify the connection of third party/custom slave controllers to any AMBA 3 AXI or AMBA 4 AXI fabric
DW_axi_hmx	Configurable high performance interface from an AHB master to an AMBA 3 AXI and AMBA 4 AXI slave

Table 1-1 DesignWare AMBA Fabric – Product Code: 3768-0 (Continued)

Component Name	Description
DW_axi_rs	Configurable standalone pipelining stage for AMBA 3 AXI or AMBA 4 AXI subsystems
DW_axi_x2h	Bridge from AMBA 3 AXI or AMBA 4 AXI to AMBA 2.0 AHB, enabling easy integration of legacy AHB designs with newer AXI systems
DW_axi_x2p	High performance, low latency interconnect fabric and bridge for AMBA 2 & 3 APB for direct connect to AMBA 3 AXI or AMBA 4 AXI fabric
DW_axi_x2x	Flexible bridge between multiple AMBA 3 AXI components or busses

1 Product Overview

The DW_axi_a2x is a configurable bridge between an AHB (Advanced High-performance Bus) or AXI (Advanced eXtensible Interface) bus protocol and an AXI bus protocol. The DW_axi_a2x component complies with AMBA AXI and AHB specifications and is part of the family of DesignWare Synthesizable Components for AMBA.

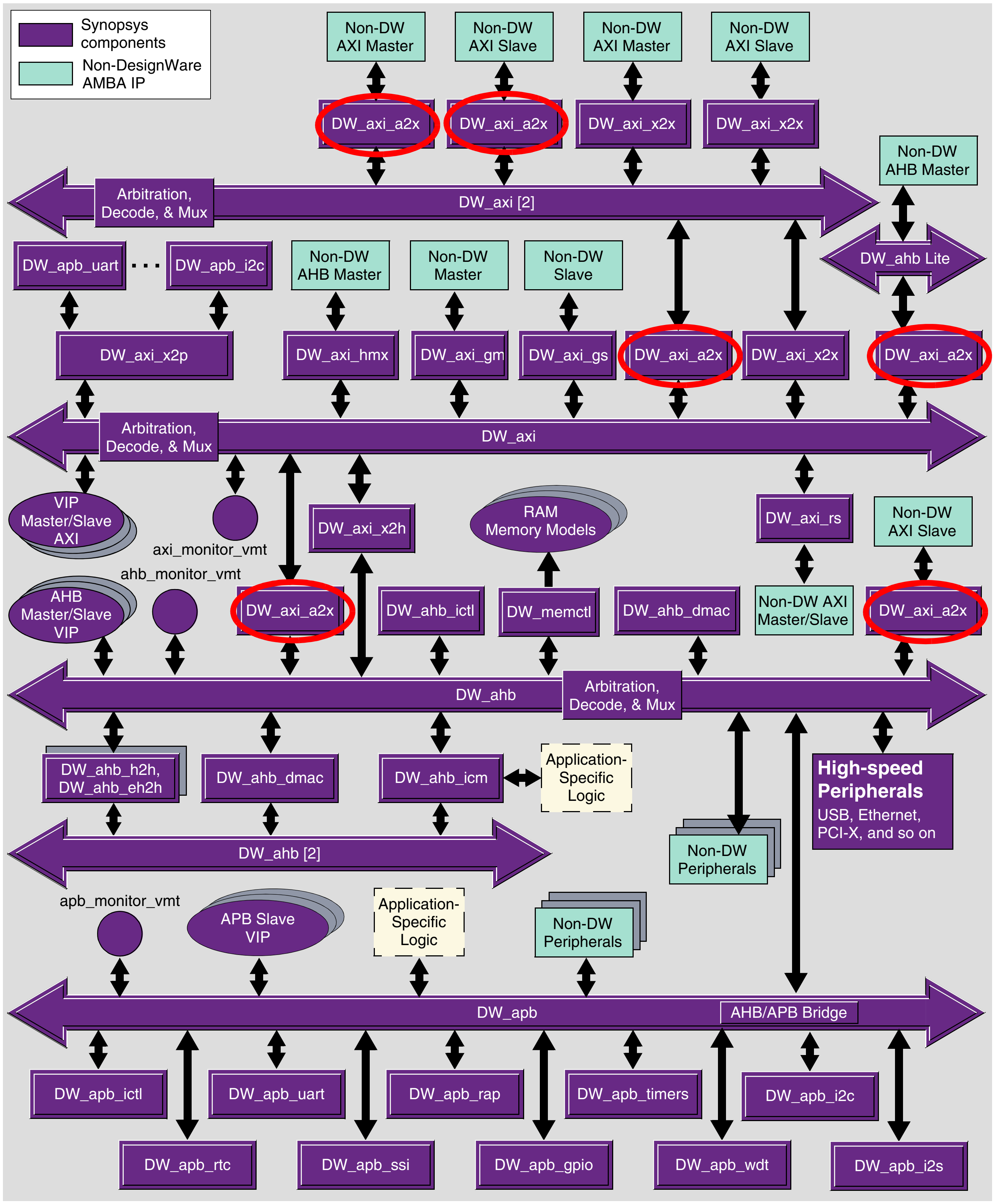
1.1 DesignWare Synthesizable Components for AMBA System Overview

The Synopsys DesignWare Synthesizable Components environment is a parameterizable bus system containing AMBA 2.0-compliant AHB and APB components, and AMBA 3 AXI/AMBA 4 AXI and ACE-Lite compliant components.

Figure 1-1 illustrates one example of this environment, including the AXI bus, the AHB bus, and the APB bus. Included in this subsystem are synthesizable IP for AXI/AHB/APB peripherals, bus bridges, and an AXI bus protocol and AHB bus fabric. Also included are verification IP for AXI/AHB/APB master/slave models and bus monitors. In order to display the databook for a DW_* component, click on the corresponding component object in the illustration.

Attention	Links resolve only if you are viewing this databook from your $DESIGNWARE_HOME tree, and to only those components that are installed in the tree.

Figure 1-1 Example of DW_axi_a2x in a Complex System



Usage scenarios can include the following connections:

- AHB bus fabric to AXI slave
- AHB bus fabric to AXI bus fabric
- AXI/ACE-Lite master to AXI/ACE-Lite slave
- AXI/ACE-Lite master to AXI/ACE-Lite bus fabric
- AXI/ACE-Lite bus fabric to AXI/ACE-Lite slave
- AXI/ACE-Lite bus fabric to AXI/ACE-Lite bus fabric

1.2 General Product Description

The DesignWare DW_axi_a2x provides a method to transfer transactions between an AHB or AXI bus fabric to an AXI bus fabric. It supports AMBA 3 AXI, AMBA 4 AXI, and ACE-Lite on the AXI interface.

1.3 DW_axi_a2x Features

The following subsections list the features of the DW_axi_a2x.

1.3.1 General DW_axi_a2x Features

The general features of the DW_axi_a2x are:

- Complies with the following specifications:
  - AMBA 3 AXI
  - AMBA 4 AXI
  - ACE-Lite
- Complies with the AMBA AHB protocol specification
- Connects AHB bus fabric, or AXI bus fabric to AXI bus fabric
- Supports asynchronous, synchronous, and quasi-synchronous clocking
- Provides asynchronous timing mode options to ease clock domain crossing
- Configurable buffer depth for address and data channels
- Configurable Store and Forward or Cut-Through mode
- Configurable address port widths of any range from 32 to 64
- Configurable data bus widths of 8, 16, 32 64, 128, 256, 512, and 1024 bits
- Support for transaction down-sizing (larger AHB/AXI bus to smaller AXI bus)
- Dynamic controllable support for transaction up-sizing (smaller AHB/AXI bus to larger AXI bus)
- Provides support for read data interleaving
- Support for sideband/user signals

1.3.2 AHB Features

The AHB features of the DW_axi_a2x are:

- AHB SPLIT response used to control AHB bus for efficiency and coherency with bus arbiter
- AHB master ID to AXI ID translation
- Support for all AHB burst types
  - Conversion of undefined length AHB read INCR transactions to defined-length AXI burst
  - Conversion of undefined length AHB write INCR transactions to defined-length AXI burst
- Supports AHB locked to AXI locked translation (when the AMBA 3 AXI mode is selected)
- Ability to control transaction ordering from a single AHB master in non-bufferable or dynamic configurations
- Provides AHB-Lite mode
  - SPLIT response not supported
  - hready driven low

  For more information on AHB Lite, refer to "Functional Description" in the *DesignWare DW_ahb Databook*.

1.3.3 DW_axi_a2x Transaction Features

The transaction features of the DW_axi_a2x are:

- Independent write/read control and data buffers
- Write transactions issued on secondary port in the same order received from the primary port
- Read transactions issued on secondary port in the same order received from the primary port
- Reads can pass-out internally buffered writes
- Writes can pass-out internally buffered reads
- Read data returned to the primary port in the same order received from the secondary port
- AXI secondary port outstanding read and write transactions support:
  - AXI configurations:
    - up to 16 unique read transaction IDs
    - up to 64 read transactions per unique ID
  - AHB configurations:
    - up to number of AHB masters (*A2X_NUM_AHBM*) unique read transaction ID
    - up to 16 read transactions per AHB master
- AXI secondary port outstanding write transactions support:
  - AHB/AXI configurations in bufferable mode:
    - No restriction placed on secondary port

- AXI configurations in non-bufferable response mode:
  - up to 32 unique write transaction IDs
  - up to 16 write translations per unique ID
- AHB configurations in non-bufferable or dynamic response mode:
  - up to number of AHB Masters (*A2X_NUM_AHBM*) unique write transaction ID
  - up to 16 write transactions per AHB master

- Provides bufferable or non-bufferable response mode
- Provides AMBA 4 AXI signaling
  - Quality of service (QoS) signals
  - Multi-region signals
- Provides ACE-Lite signaling
  - Domain, Snoop, and Barrier signals
- Provides Sideband/User signaling
  - Sideband signals when the AMBA 3 AXI mode is selected
  - User signals when either the AMBA 4 AXI or the ACE-Lite mode is selected
- Support for AXI/AHB locked transactions (when the AMBA 3 AXI mode is selected)
- Provides endian support:
  - Word-invariant endianness (BE32) between AHB and AXI
  - Address-invariant endianness (BEA) between AHB and AXI
  - Byte-invariant endianness (BE8) between AXI and AXI

1.3.4 AHB Transaction Features

The AHB transaction features of the DW_axi_a2x are:

- Dynamic or static control for write/read ordering from same AHB master
- AHB Write Transactions
  - DW_axi_a2x de-asserts hready if address or write data buffer full
  - In split mode:
    - i. DW_axi_a2x write transaction split if hready remains low for defined period
    - ii. When AHB Master is split, DW_axi_a2x strobes remaining data beats of write transaction
    - iii. AHB master is recalled; DW_axi_a2x can accept more write data
    - iv. When master returns, AHB transaction is treated as new AXI transaction.; however, length is adjusted to number of remaining data beats from original AHB transaction
  - All buffered AHB transactions receive OK response
  - All bufferable AHB INCR transactions divided into multiple AXI transfers with length of each transfer defined by hincr_wbcnt_m* port or A2X_HINCR_WBCNT_MAX parameter

- In split mode, non-Bufferable transactions split on last beat and recalled when response returned to DW_axi_a2x
- In non-split, hready is driven low in response to non-bufferable transactions and only asserted when response is returned to DW_axi_a2x
- Non-bufferable AHB write INCR transaction is treated as single INCR transactions with secondary port transaction length (awlen_sp) of 0

- AHB Read Transactions
  - In split mode:
    i. All accepted AHB reads receive split response from DW_axi_a2x
    ii. AHB master is then recalled from split when data returned on AXI
  - In non-split mode, hready is driven low in response to AHB read request and remains low until data is returned to DW_axi_a2x
  - AHB INCR transactions converted to defined-length AXI transaction with prefetch length defined by hincr_rbcnt_m* port or A2X_HINCR_RBCNT_MAX parameter
  - All remaining data beats flushed from read data buffer if AHB read INCR transaction are less than read prefetch length
  - AHB Master with read ERROR response has remaining data beats flushed from read data buffer if htrans driven to IDLE or NSEQ
- EBT (Early Burst Terminated) Write Conditions
  - Any defined-length write transaction early-burst terminated by another transaction has remaining data beats strobed on secondary port
  - When early-burst terminated master returns, AHB transaction is treated as new AXI transaction; length is adjusted to number of remaining data beats from original AHB transaction
  - If non-bufferable write transaction is early-burst terminated by another transaction, DW_axi_a2x may send multiple non-bufferable AXI write transactions on secondary port; responses are combined into one AHB response before recalling AHB master from split
- EBT (Early Burst Terminated) Read Conditions
  - Any defined-length read transaction must return to DW_axi_a2x to retrieve all data; no action taken if defined-length read transaction is early-burst terminated

1.3.5 DW_axi_a2x Locked Transactions

Locked transaction features of the DW_axi_a2x are:

- Supported with non-bufferable response mode
- Last transaction of a downsized unlock sent as unlocking transaction

---

Note Locked transactions are supported only when the AMBA 3 AXI mode is selected.

1.3.6 AHB Locked Transactions

Locked AHB transaction features of the DW_axi_a2x are:

- DW_axi_a2x de-asserts hready on last data beats for AHB locked write or on first read transaction for AHB locked read
- Separate read data buffer for locked read data
- Read data buffer depth selected such that all outstanding AXI read data transactions can be internally buffered before sending AXI locked transaction

Note Locked transactions are supported only when the AMBA 3 AXI mode is selected.

1.3.7 Latency Features

Latency features of the DW_axi_a2x are:

- AXI latency
  - Primary port write address to secondary port address — minimum one clock cycle
  - Primary port read address to secondary port address — minimum one clock cycle
  - Primary port write data to secondary port — minimum one clock cycle
  - Secondary port read data to primary port — minimum one clock cycle
- AHB latency
  - AHB write address to secondary port address — minimum one clock cycle
  - AHB read address to secondary port address — minimum one clock cycle
  - AHB write data to secondary port — minimum one clock cycle
    - AHB read address to AHB read data - minimum four clock cycles in AHB non-split mode
    - One clock cycle for AHB read address to AXI secondary port
    - One clock cycle for AHB read data to appear on secondary port interface.
    - One clock cycle for secondary port read data to primary port
    - One clock cycle to return the first read data beat
  - AHB read address to AHB read data — minimum seven clock cycles in AHB split mode
    - One clock cycle for AHB read address to AXI secondary port
    - One clock cycle for AHB read data to appear on secondary port interface.
    - One clock cycle for secondary port read data to primary port
    - One clock cycle to recall the AHB master from split
    - Two clock cycle for AHB Master to return from split.
    - One clock cycle to return the first read data beat

1.4 Standards Compliance

The DW_axi_a2x conforms to the *AMBA AXI and ACE Protocol Specification* from ARM. Readers are assumed to be familiar with these specifications.

1.5 Verification Environment Overview

The DW_axi_a2x includes an extensive verification environment, which sets up and invokes your selected simulation tool to execute tests that verify the functionality of the configured component. You can then analyze the results of the simulation through the simulation waveforms to see how the DW_axi_a2x performs under the provided test conditions.

The DW_axi_a2x testbench environment is described in "Verification" on page 157.

1.6 Where To Go From Here

At this point, you may want to get started working with the DW_axi_a2x component within a subsystem or by itself. Synopsys provides several tools within its coreTools suite of products for the purposes of configuration, synthesis, and verification of single or multiple synthesizable IP components—coreConsultant and coreAssembler. For information on the different coreTools, refer to *Guide to coreTools Documentation*.

For more information about configuring, synthesizing, and verifying just your DW_axi_a2x component, see the *"Overview of the coreConsultant Configuration and Integration Process"* section in the *DesignWare Synthesizable Components for AMBA 2 User Guide*.

For more information about implementing your DW_axi_a2x component within a DesignWare subsystem using coreAssembler, see the "Overview of the coreAssembler Configuration and Integration Process" section in the *DesignWare Synthesizable Components for AMBA 2 User Guide*.

2 Functional Description

The DW_axi_a2x connects any of the following fabrics:

- AHB to AMBA 3 AXI
- AHB to AMBA 4 AXI
- AMBA 3 AXI to AMBA 3 AXI
- AMBA 4 AXI to AMBA 4 AXI
- ACE-Lite to ACE-Lite

The DW_axi_a2x supports:

- Bridging from primary port fabric (AHB or AXI) to secondary port fabric
- Supports asynchronous primary and secondary port clocks
- Supports transaction downsizing (primary port data bus wider than secondary port data bus)
- Supports transaction upsizing (primary port data bus narrower than secondary port data bus)

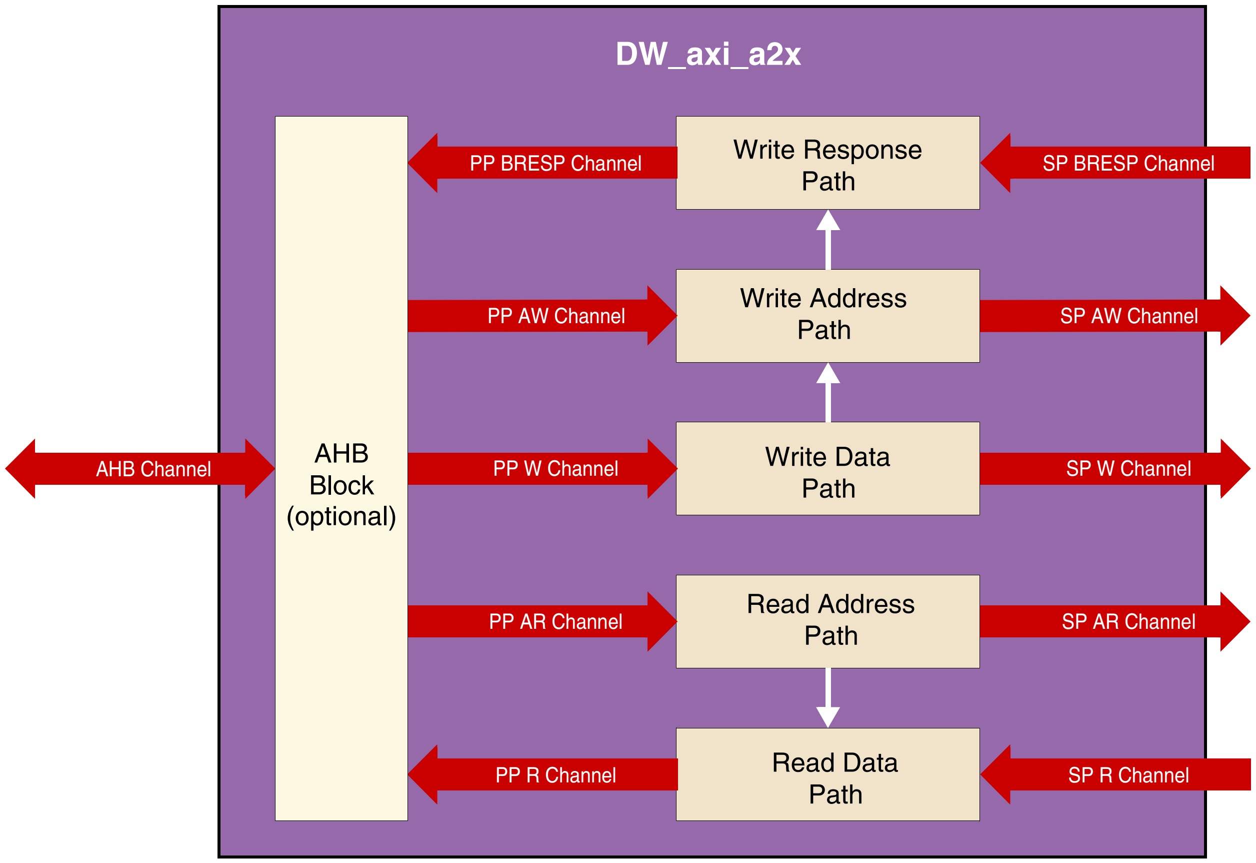
The DW_axi_a2x has a store and forward feature that does the following:

- Ensures all write data for transaction is present in write data buffer before issuing write address onto secondary port
- Ensures available space in read data buffer for all beats of transaction before issuing read address onto secondary port

2.1 DW_axi_a2x Block Diagram

Figure 2-1 illustrates the block diagram of the DW_axi_a2x.

Figure 2-1 DW_axi_a2x Block Diagram



2.2 AHB Primary Port Configuration

The AHB block shown in Figure 2-1 is included in the RTL when the A2X_PP_MODE configuration parameter is set to 0.

2.3 DW_axi_a2x Channel Transactions

Transactions on every channel are buffered from channel source to channel sink. The following relate to DW_axi_a2x channel transactions:

- Transfers are accepted from source as soon as there is space in channel buffer.
- When operating in cut-through mode, transfers are issued to channel sink as soon as channel buffer is populated.
- Stalling conditions, such as store/forward and resizing, can delay the transfer to the channel sink;.
- For synchronous configurations, minimum latency from channel source to channel sink is one clock cycle.
- For asynchronous configurations, minimum latency from channel source to channel sink is one primary clock cycle and two secondary clock cycles.

2.4 Transaction Resizing

Transaction resizing requires the DW_axi_a2x to change the size (and possibly the length) of a transaction. This resizing occurs when the DW_axi_a2x transfers data from a smaller primary port to a larger secondary port—upsizing—or when the DW_axi_a2x transfers data from a larger primary port to a smaller secondary port—downsizing.

Transaction resizing can be performed separately for read and write transactions, depending upon the following configurations:

- When DW_axi_a2x is configured for AMBA 3 AXI mode:
  - DW_axi_a2x performs write transaction upsizing only when awresize_pp is set for write transactions.
  - DW_axi_a2x performs read transaction upsizing only when arresize_pp is set for read transactions.

  DW_axi_a2x does not consider awcache[1]/arcache[1] signal for upsizing of read or write transactions.
- When DW_axi_a2x is configured for AMBA 4 AXI/ACE-Lite mode:
  - DW_axi_a2x performs write transaction upsizing only when both awresize_pp and awcache[1] are set for write transactions.
  - DW_axi_a2x performs read transaction upsizing only when both arresize_pp and arcache[1] are set for read transactions.

For downsizing read or write transactions, DW_axi_a2x always resizes the transactions to multiple smaller transactions when the secondary port data width is less than the primary port data width. This is done even when the awcache[1] or arcache[1] is set. Therefore, non-modifiable transactions are supported only when the secondary port data width is greater than the primary port data width.

The following sections on transaction upsizing and downsizing describe the transformations for AXI-to-AXI transactions. To upsize or downsize an AHB transaction, the DW_axi_a2x does the following:

1. Transforms the AHB transaction into an AXI transaction; refer to "AMBA 3 AXI Non-Protocol Sideband Signals" on page 48
2. Applies resizing rules

Thus for an AHB configuration with a primary port data width of 32 and a secondary port data width of 64, the DW_axi_a2x:

1. Converts the AHB transaction into a AXI transaction based on the primary port data width
2. Resizes to the secondary port data width

2.4.1 Transaction Upsizing

Transaction upsizing takes transactions from a smaller primary port data bus and alters the burst size to the maximum allowable size on a larger secondary port. The burst length of the transaction becomes smaller as a result of this operation.

Transaction upsizing can be dynamically disabled for both read and write transactions by de-asserting the arresize_pp and awresize_pp input ports during the address phase of the transaction. If this bit is de-asserted, then the transaction is passed unaltered from the channel source to the channel sink.

The upsized transaction issued on the secondary port is calculated as follows:

- *upsize_ratio*:

  $$\frac{secondary\_port\_data\_width}{primary\_port\_data\_width}$$

- *size*:

  If the input primary port transaction size is equal to the maximum size allowed for the primary port, then:

  secondary_port_size = *max_secondary_port_size*

  Otherwise (input transaction is sub-sized on the primary port):

  secondary_port_size = *primary_port_size*

- *burst_length*:

  The burst length calculation depends on aligned or aligned address.

  - For aligned address:

    Secondary port burst length (awlen_sp/arlen_sp) = *Primary Port Burst length (awlen_pp or arlen_sp)/upsize_ratio*

  - For unaligned address:

    Secondary port burst length (awlen_sp/arlen_sp) = *(Primary Port Burst length (awlen_pp/arlen_sp)* + *Unaligned Beats)/upsize_ratio*

When upsizing, the following may change on the DW_axi_a2x secondary port:

- Transfer length
- Burst type
- Size
- Address

Since the transaction length is always a divided value of the upsizing ratio, only one transaction is required on the secondary port to complete the transfer. The exceptions to this are WRAP transactions because wraps are divided into INCR transactions; thus, when upsizing a wrap transaction, the DW_axi_a2x generates two addresses on the secondary port.

Figure 2-2 shows the upsizing of a DW_axi_a2x write transaction with a 16-bit primary port to a 32-bit secondary port when the primary port is configured for AMBA 3 AXI mode. Figure 2-3 shows a similar transaction in the AMBA 4 AXI mode. In this example, upsizing takes place only when both the modifiable bit (awcache[1]) and awresize_pp are driven to 1.

Figure 2-2 Write Transaction Upsizing in AMBA 3 AXI Mode

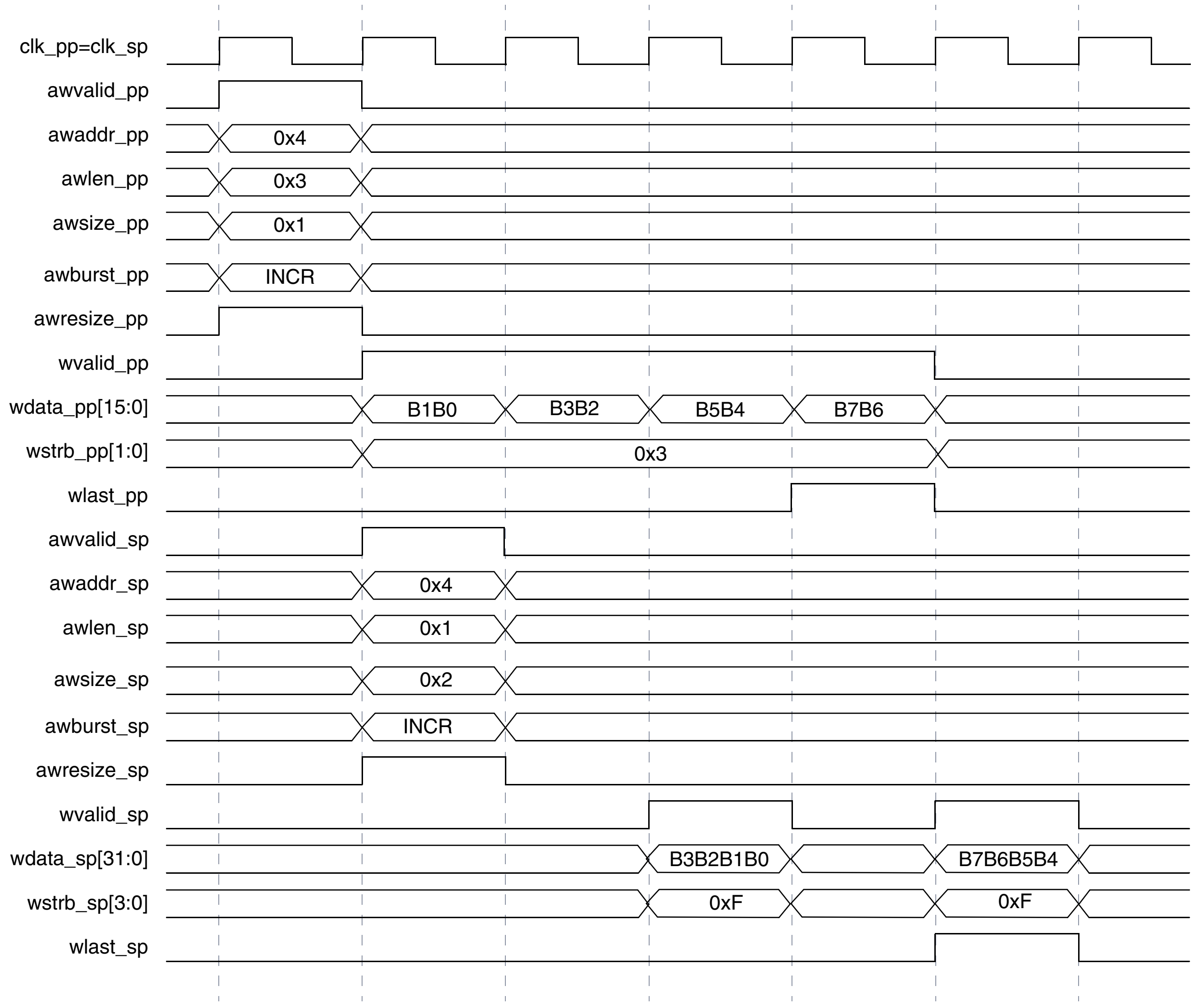
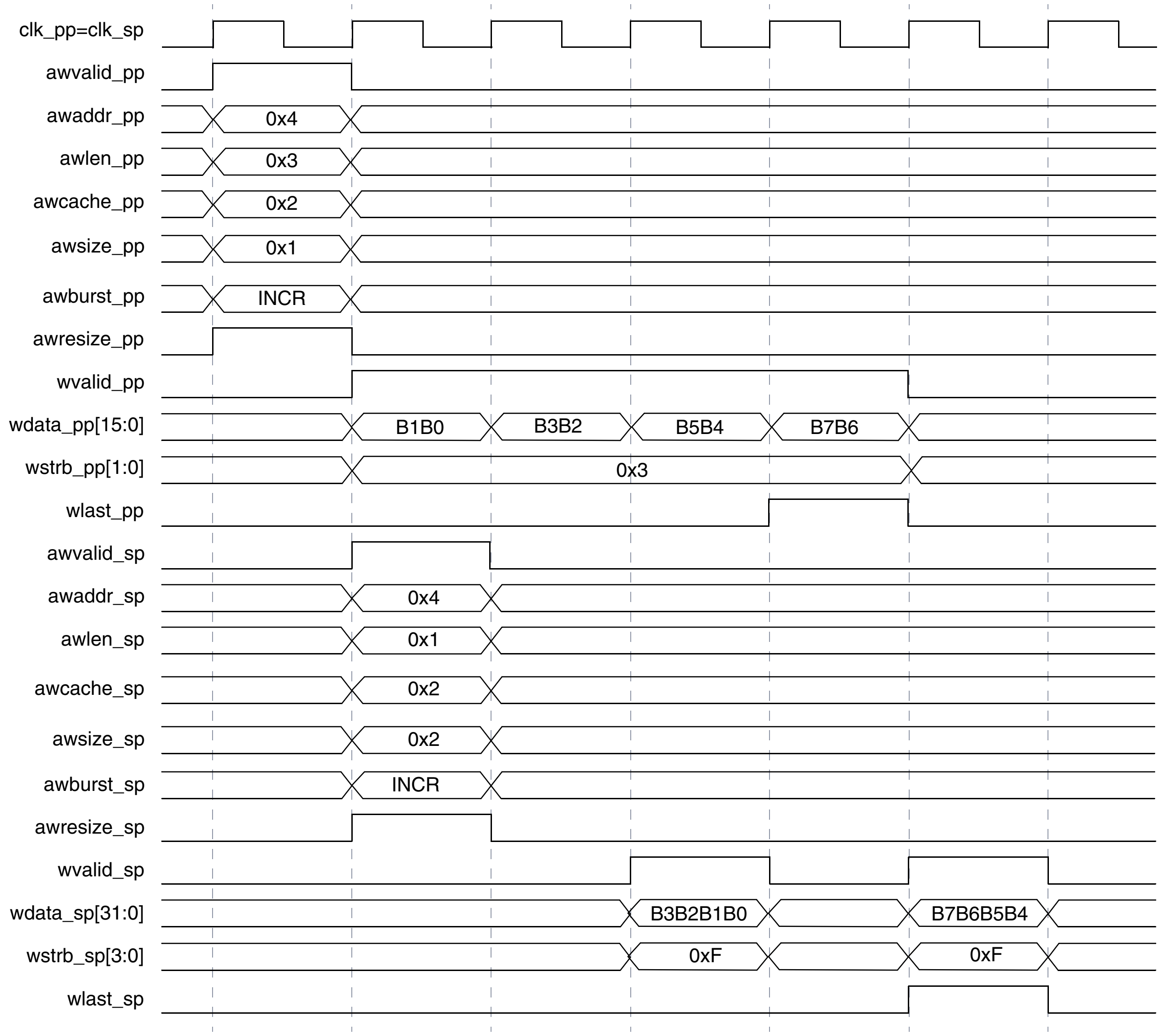


Figure 2-3 Write Transaction Upsizing in AMBA 4 AXI Mode



The examples shown in Figure 2-2 and Figure 2-3 assume that the secondary port is draining at the same clock rate as the primary port, thus requiring two clock cycles to buffer each secondary port data transfer.

If the primary port burst length was an odd length—awlen_pp was 0x2—the transaction on the secondary port would be awlen_sp of 0x1 with the last beats strobed; that is, awlen_sp would be 0x1 and the second data beat would be 0000B5B4 with a strobe of 0x3.

Figure 2-4 shows the upsizing of a DW_axi_a2x read transaction with an 16-bit primary port to a 32-bit secondary port when the primary port is configured for AMBA 3 AXI mode. Figure 2-5 shows a similar transaction for the AMBA 4 AXI mode.

Figure 2-4 Read Transaction Upsizing in AMBA 3 AXI Mode

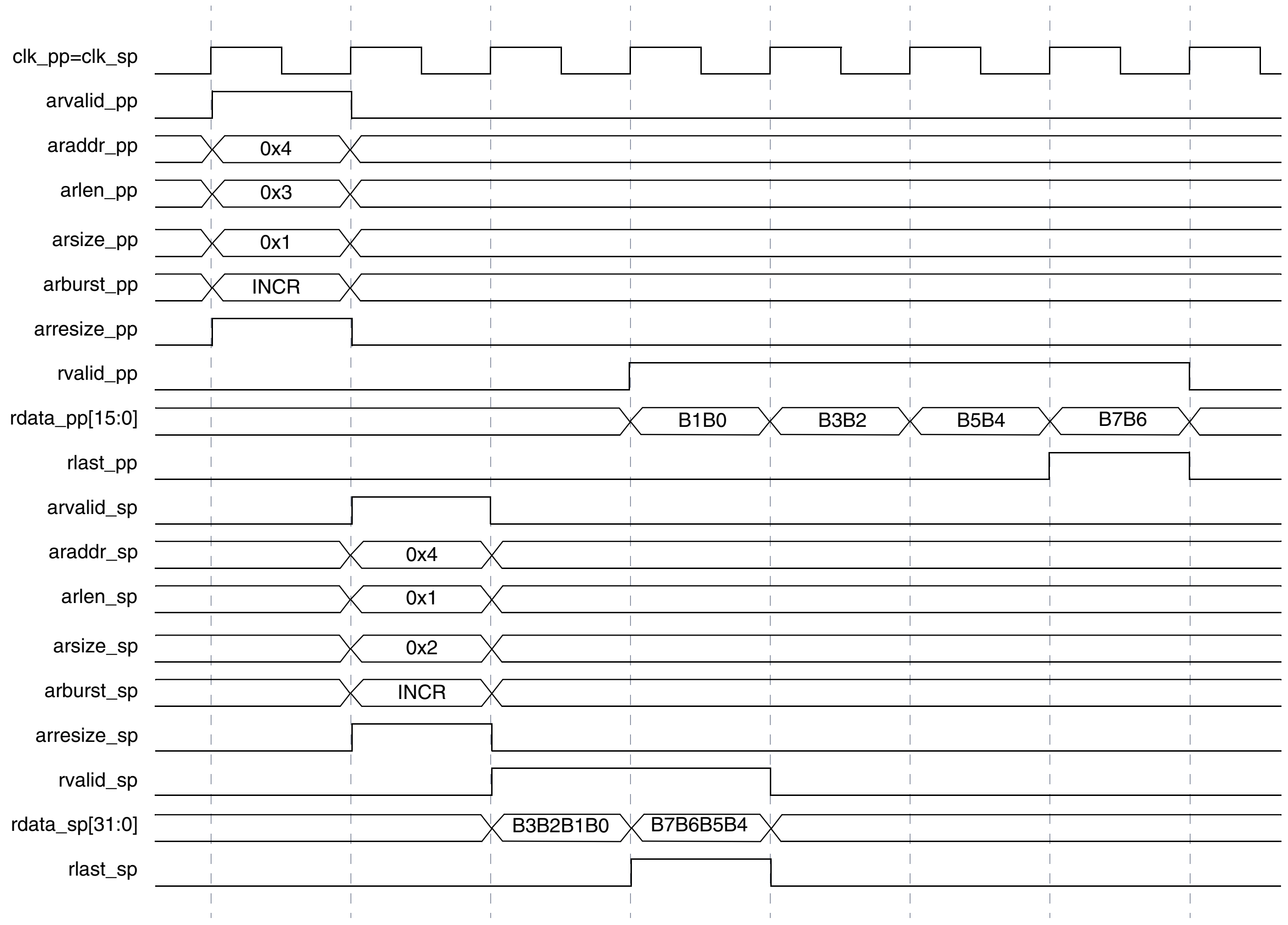
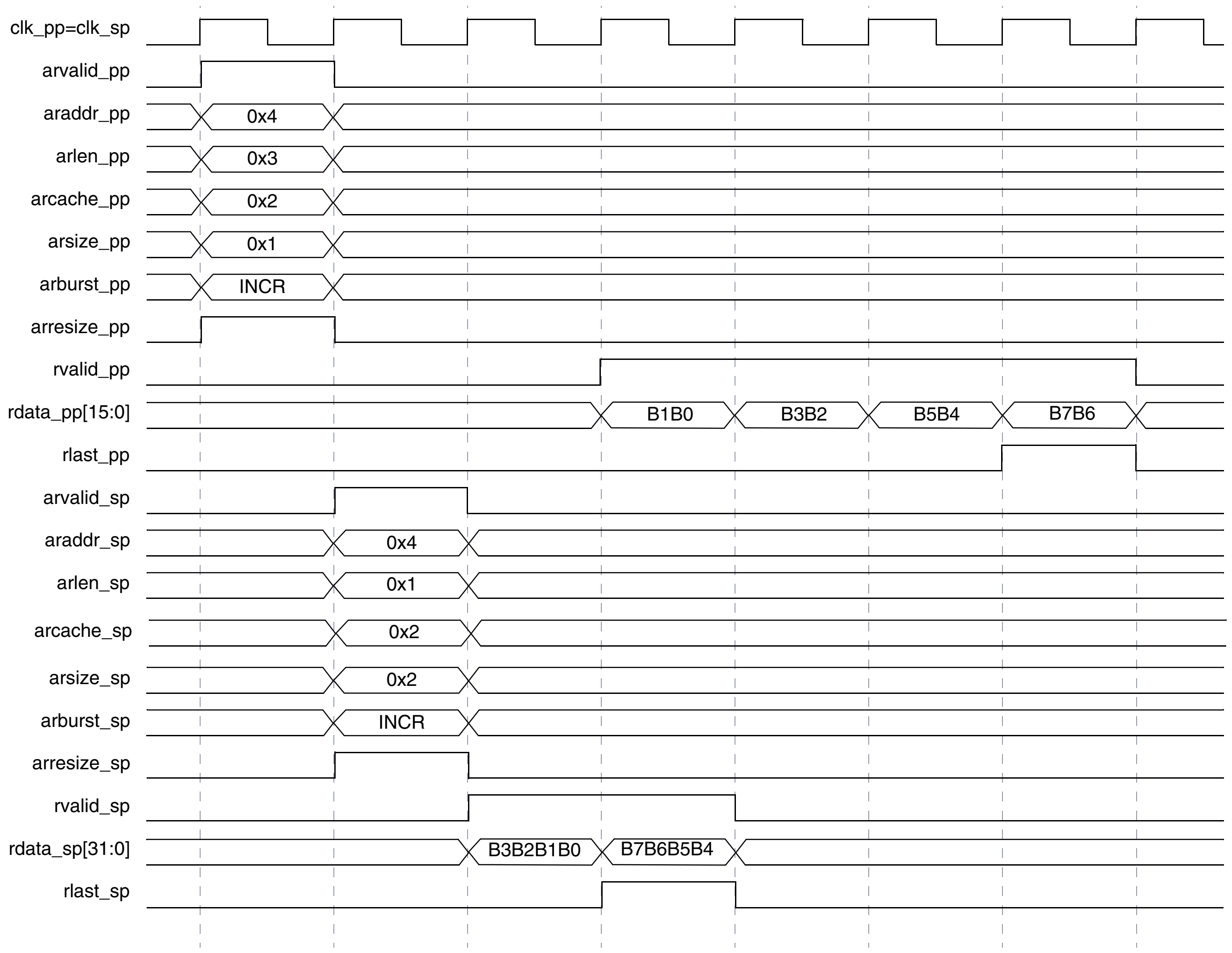


Figure 2-5 Read Transaction Upsizing in AMBA 4 AXI Mode



2.4.1.1 Sub-Sized Transactions

If the input primary port transaction size is not equal to the maximum size allowed for the primary port, the transaction is sub-sized. The transaction length and size remains unaltered; only the data beats are re-mapped to the correct beat location on the secondary port.

Figure 2-6 shows the upsizing of a sub-sized transaction on a 16-bit primary port to a 32-bit secondary port when the primary port is configured for AMBA 3 AXI mode. Figure 2-5 shows a similar transaction for the AMBA 4 AXI mode.

Figure 2-6 Write Sub-Sized Transaction in AMBA 3 AXI Mode

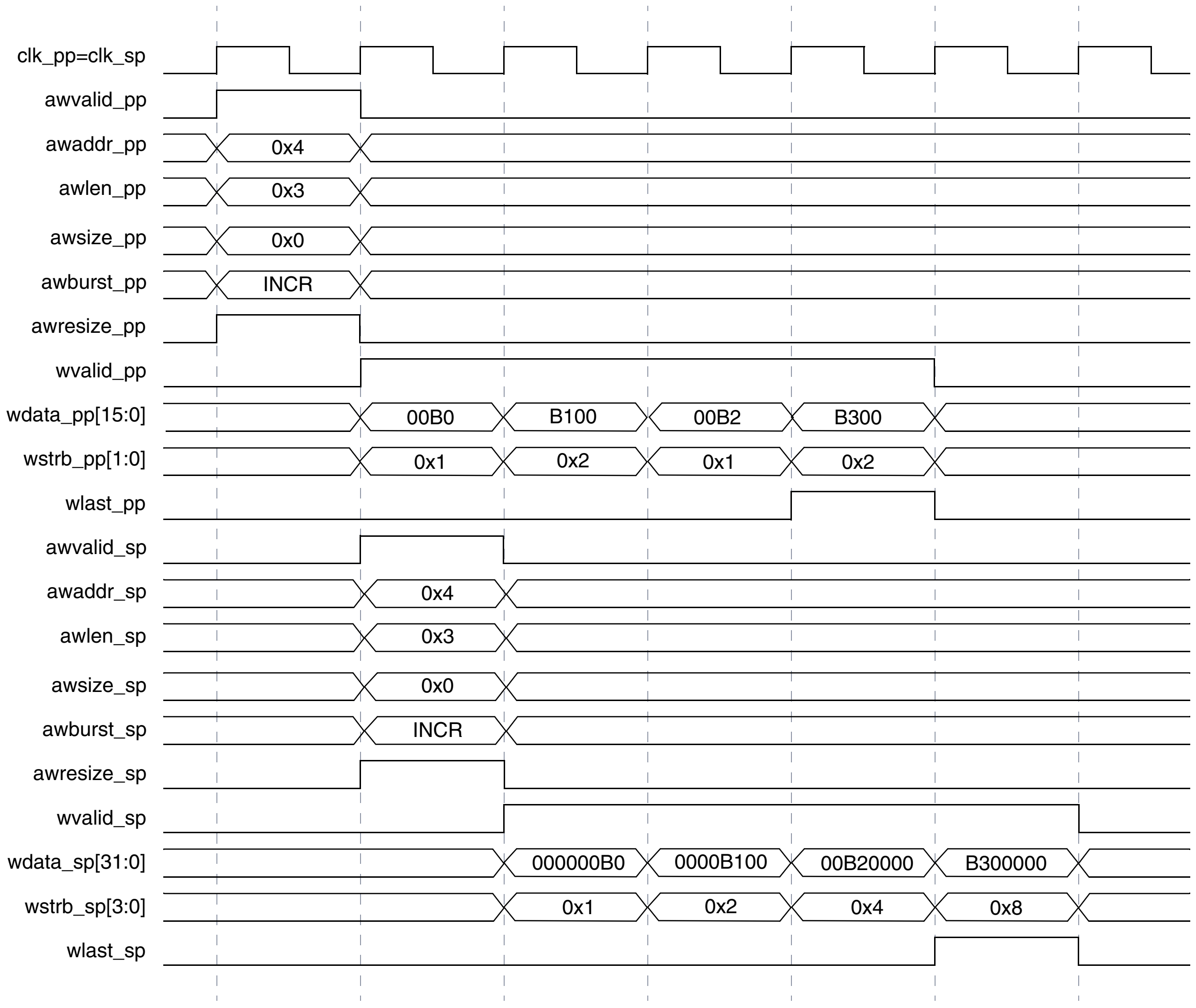
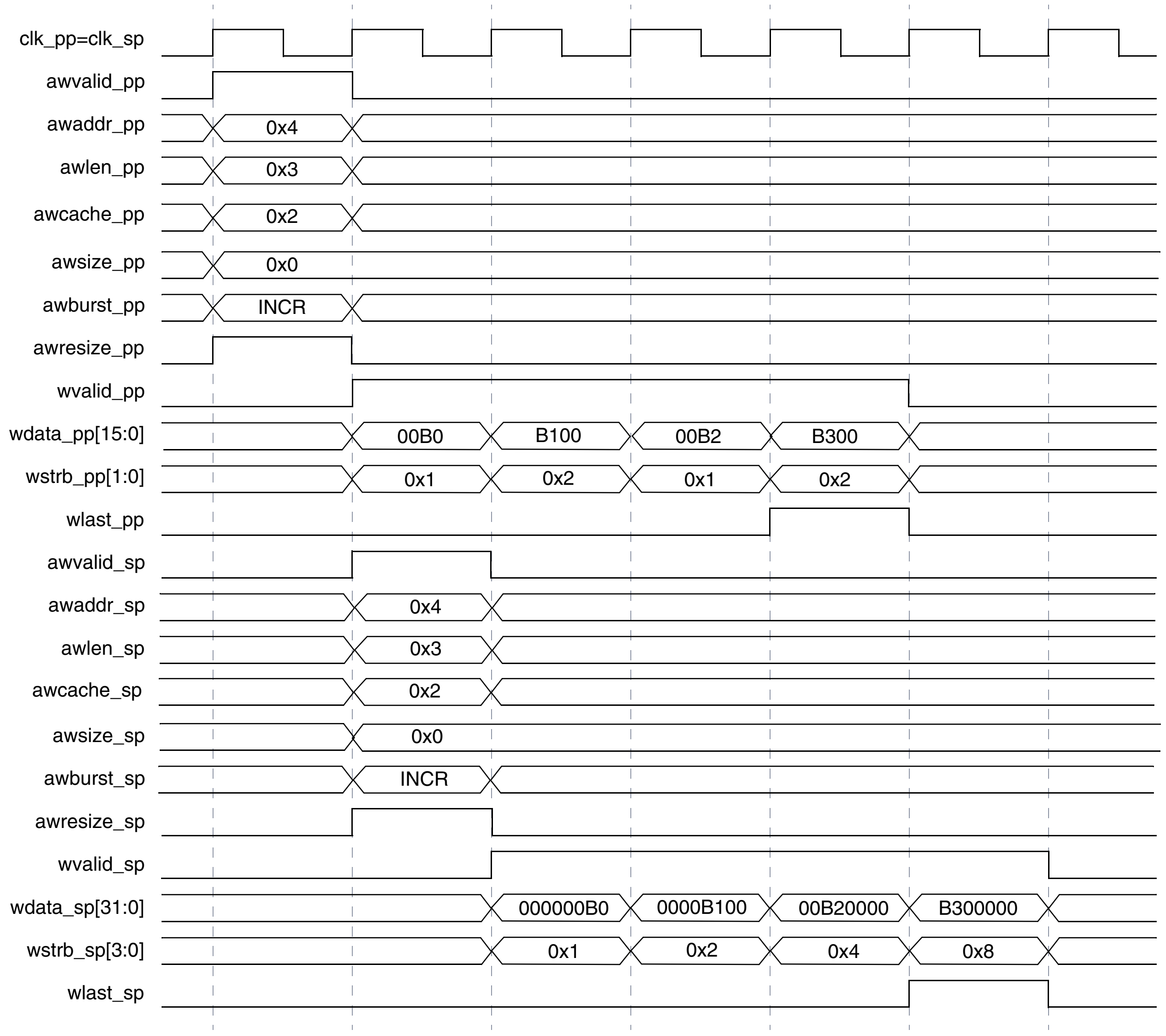


Figure 2-7 Write Sub-Sized Transaction in AMBA 4 AXI Mode



2.4.1.2 Unaligned Transactions

If the input primary port transaction address is unaligned, the secondary port transaction address remains unchanged; only the transaction length and size can change.

Figure 2-8 shows an example of an unaligned primary port transaction in AMBA 3 AXI mode and Figure 2-9 shows a similar transaction in AMBA 4 AXI mode.

Figure 2-8 Write Unaligned Upsized Transaction in AMBA 3 AXI Mode

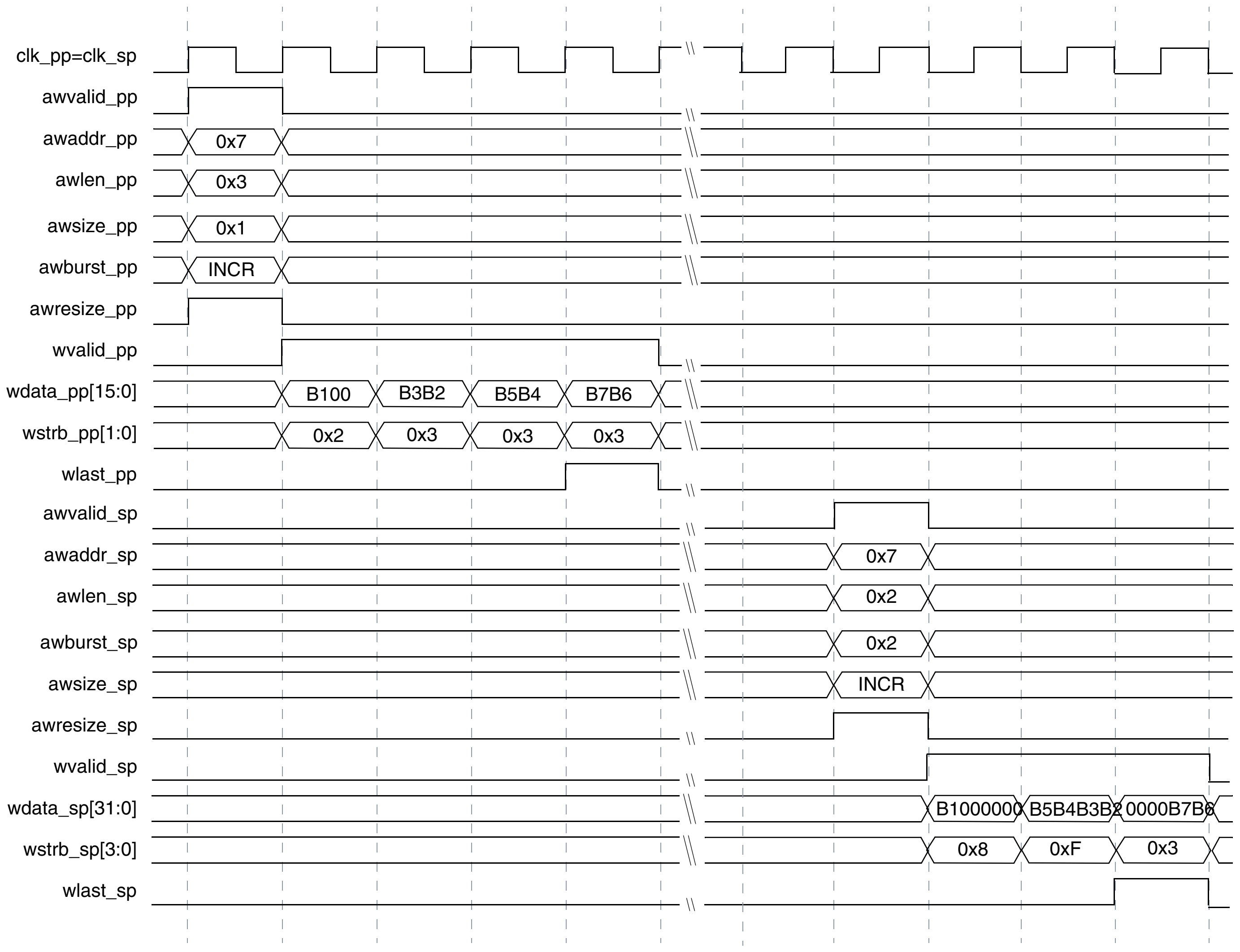
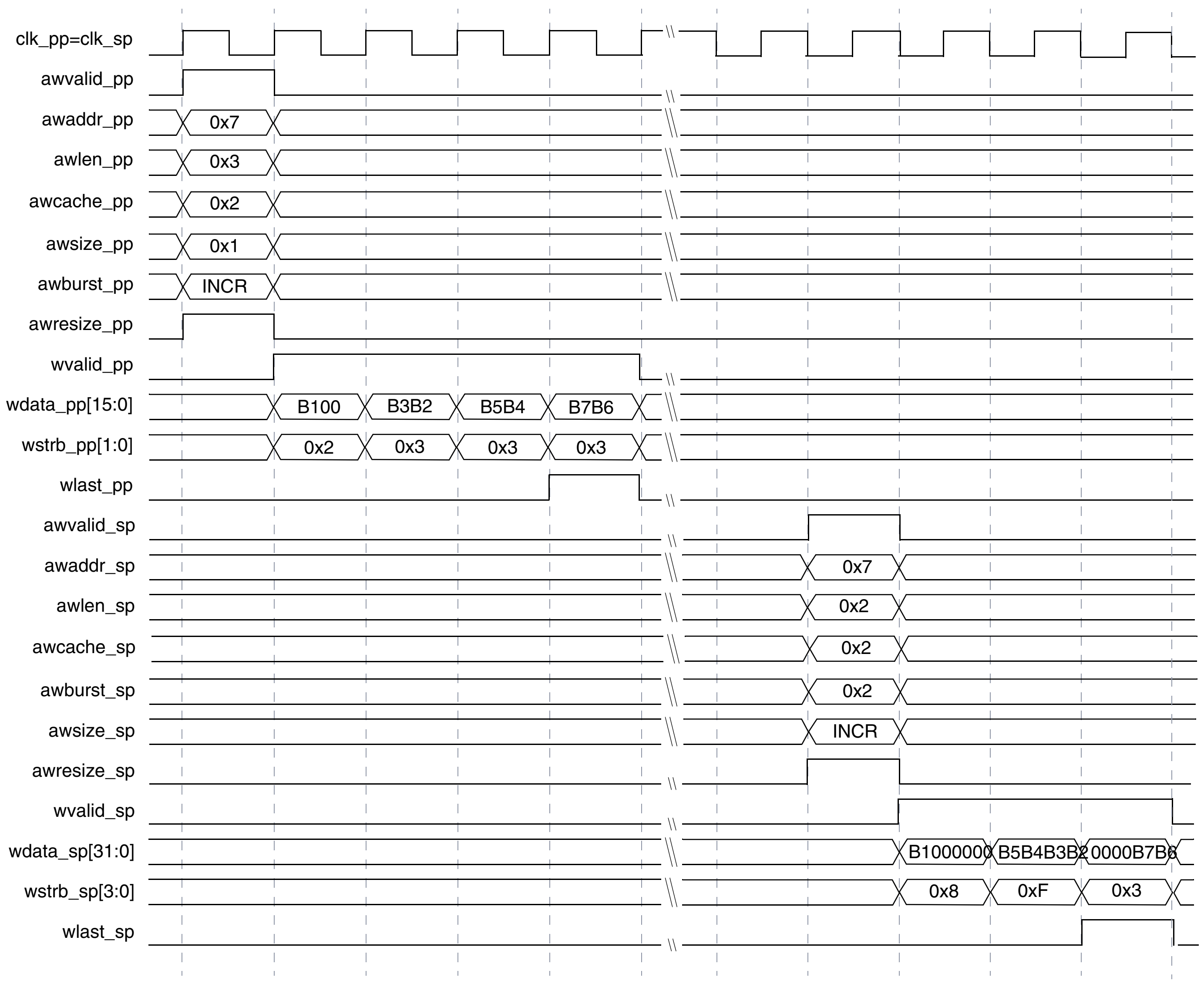


Figure 2-9 Write Unaligned Upsized Transaction in AMBA 4 AXI Mode



2.4.1.3 Burst Types

When upsizing a transaction, the length, burst type, and size may change. The new upsized length results in a divided value of the primary port burst length by the upsize ratio. The transactions burst types are as follows:

- AXI INCR transactions remain as INCR transaction with new upsized length
- AXI FIXED transactions remain as FIXED transaction with new upsized length
- AXI WRAP transactions are broken into series of INCR transactions controlled to observe address sequence dictated by wrap

Figure 2-10 shows an example of a write upsizing wrap transaction in AMBA 3 AXI mode and Figure 2-11 shows a similar transaction in AMBA 4 AXI mode.

Figure 2-10 Write Upsizing Wrap Transaction in AMBA 3 AXI Mode

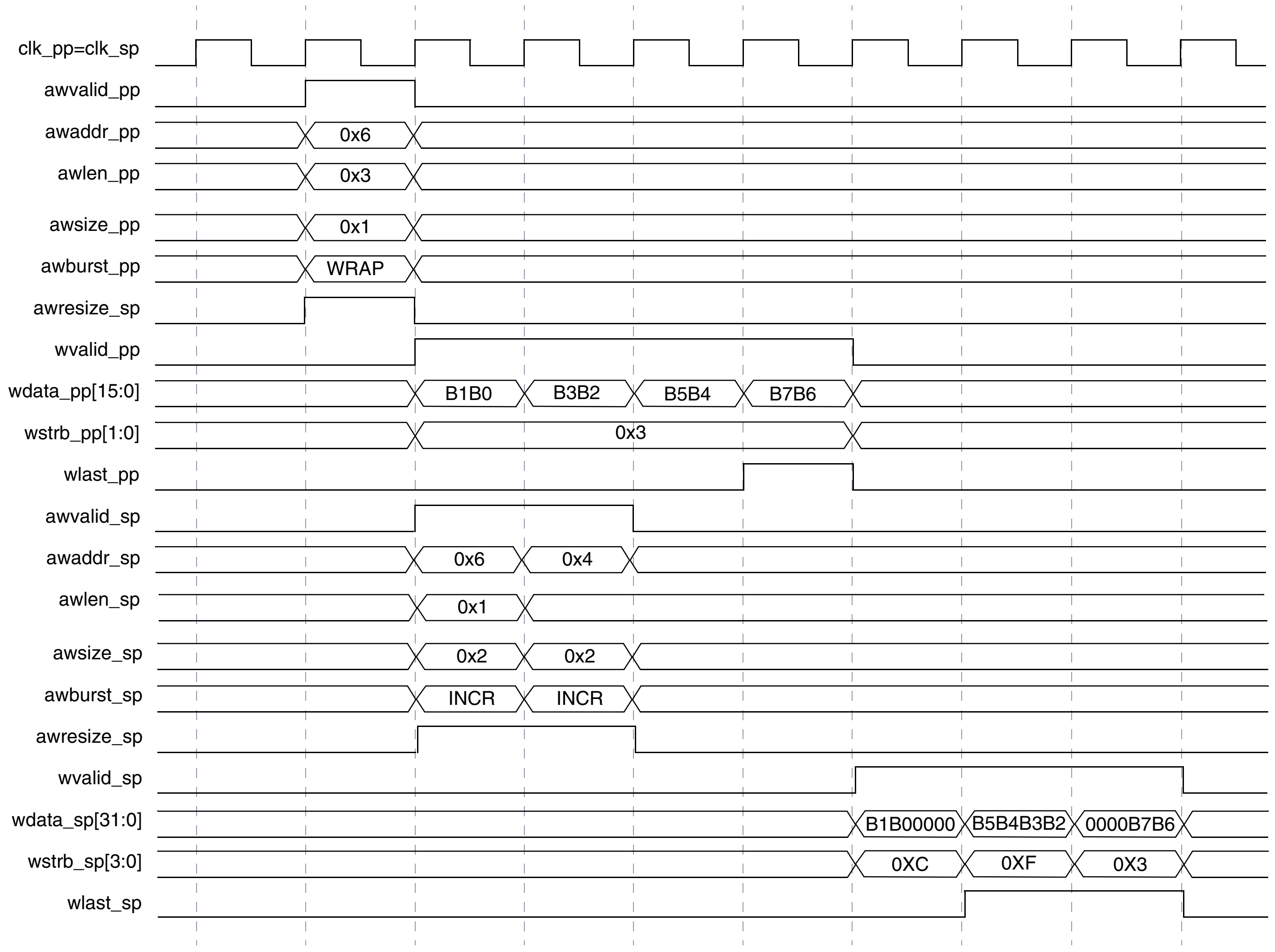
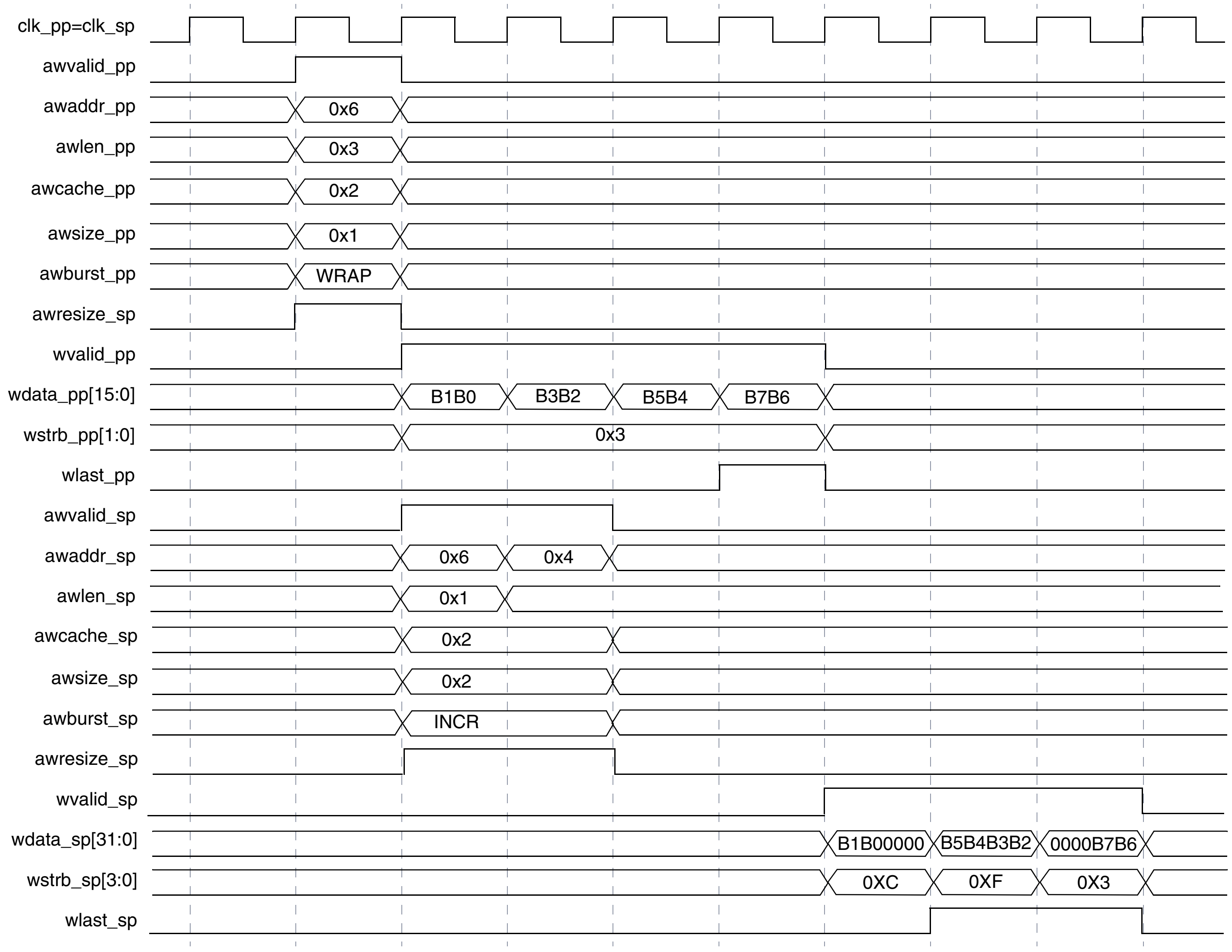


Figure 2-11 Write Upsizing Wrap Transaction in AMBA 4 AXI Mode

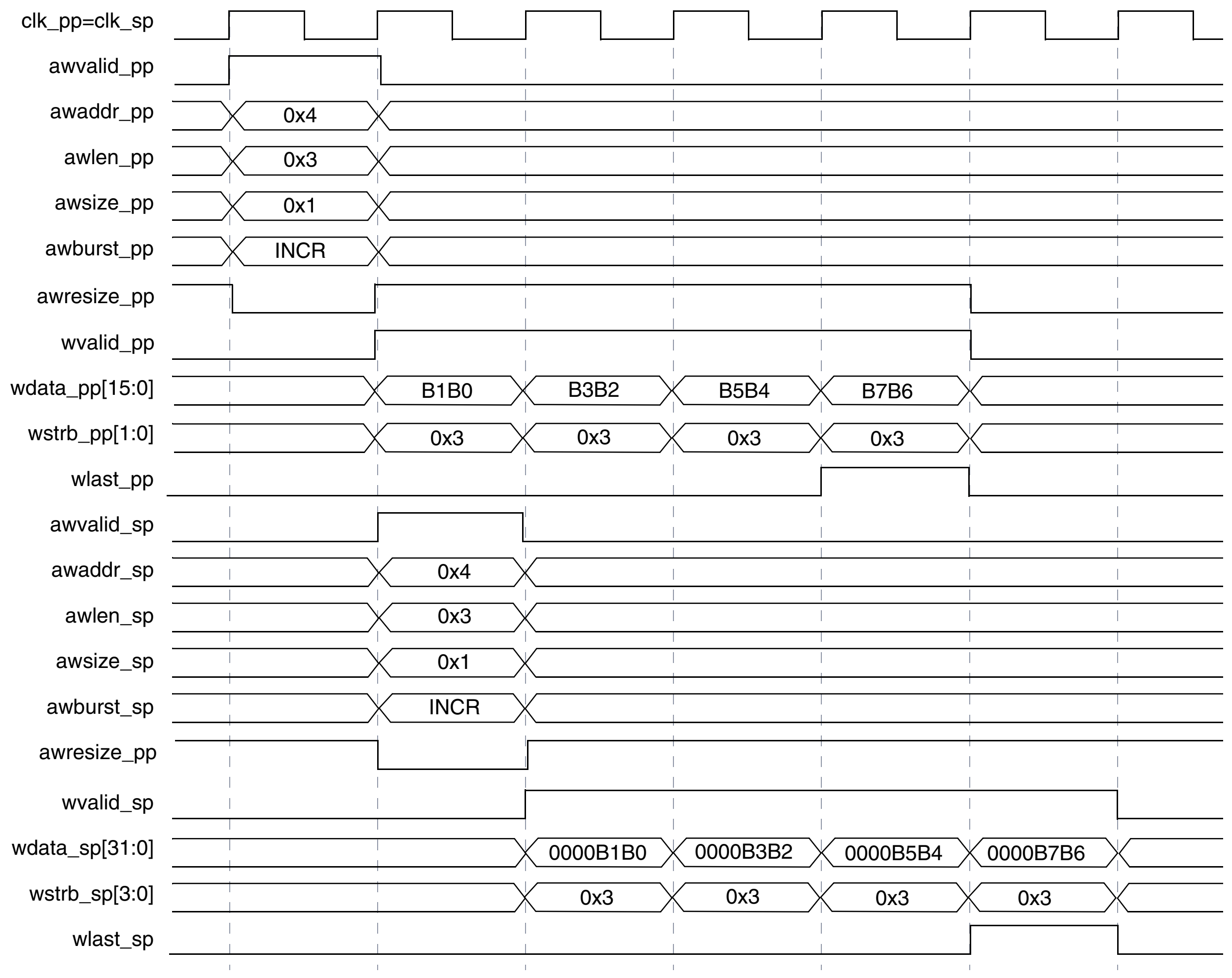


2.4.1.4 Disable Resize (awresize_pp/arresize_pp)

The DW_axi_a2x provides a signal on the AR and AW Channel that disables transaction upsizing in the DW_axi_a2x. These signals are offered only when the DW_axi_a2x is configured for upsizing.

Figure 2-12 shows an DW_axi_a2x transaction for an upsized configuration with awresize_pp set to 0.

Figure 2-12 A2X Disable Resize

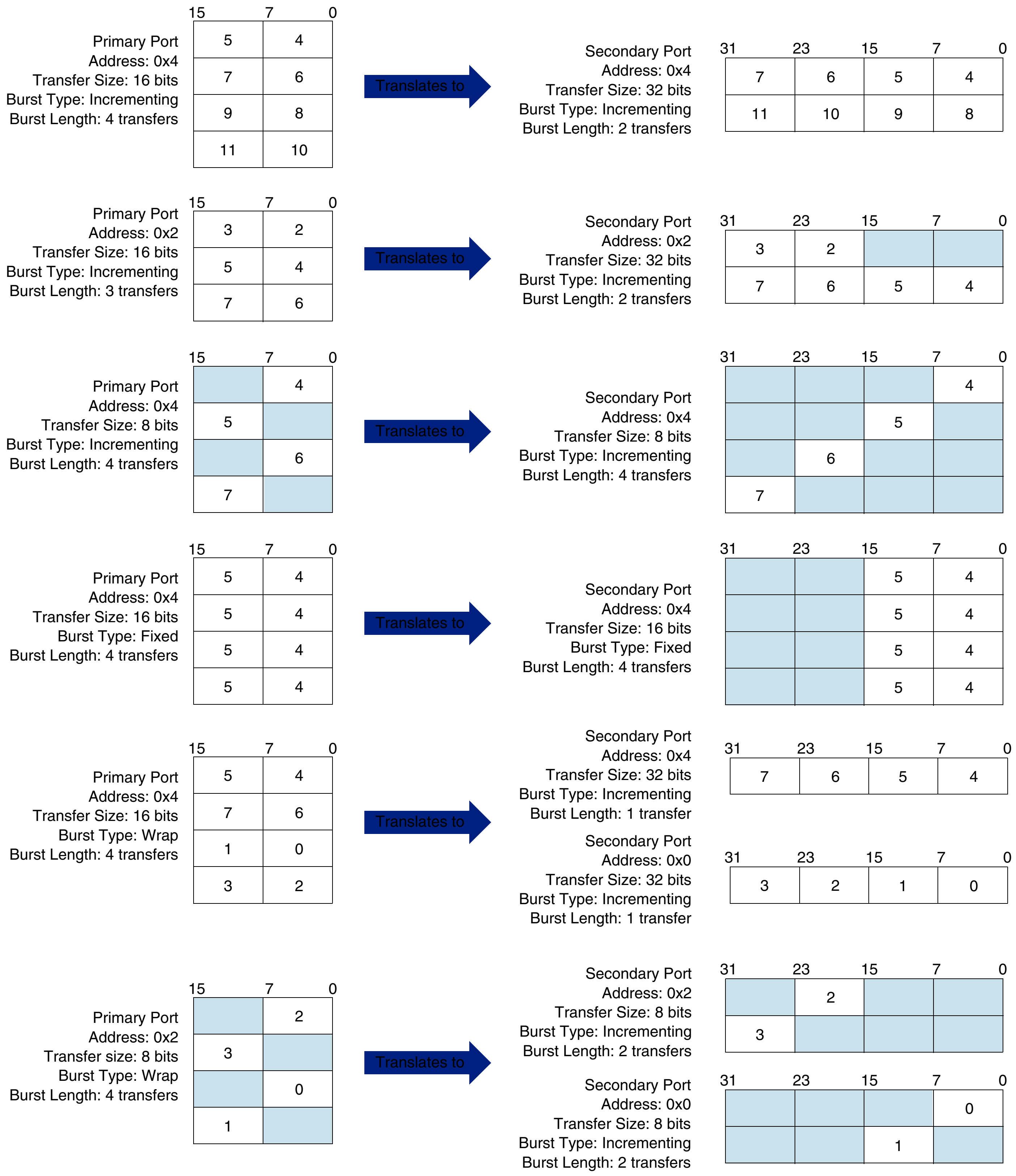


2.4.1.5 Upsized Transaction Examples

The diagrams in Figure 2-13 show examples of aligned and unaligned transfers on buses with different widths.

Each row in the diagrams represents a transfer. The shaded cells indicate bytes that are not transferred based on address and control information. The left-hand side shows how the transfer appears on the Primary Port, and the right-hand side shows how the transfers appear on the Secondary Port.

Figure 2-13 Upsized Transaction Examples



2.4.2 Transaction Downsizing

Transaction downsizing alters the burst size of transactions from a larger primary port data bus to the maximum allowable size on the smaller secondary port. The burst length of the transaction becomes larger as a result of this operation. When the downsized burst length is greater than the maximum secondary port length—that is, 2^{A2X_BLW}—the transaction is broken into multiple secondary port transactions.

The downsized transaction issued on the secondary port is calculated as follows:

- *downsize_ratio*:

  $$\frac{primary\_port\_bytes\_per\_beat}{max\_secondary\_port\_bytes\_per\_beat}$$

  Where:

  - *primary_port_bytes_per_beat* refers to the number of bytes in one primary port transaction. Thus, a write transaction with an awsize_pp of 0x1 implies a primary_port_bytes_per_beat of 2.
  - *max_secondary_port_bytes_per_beat* refers to the maximum number of secondary port bytes in one secondary port transaction; that is, secondary port data width/8.

- *size*:

  If the input primary port transaction size is less than the maximum size allowed for the secondary port, then:

  secondary_port_size = *primary_port_size*

  Otherwise:

  secondary_port_size = *max_secondary_port_size*

- *burst_length*:

  $$primary\_port\_burst\_length \times downsized\_ratio$$

Figure 2-14 shows transaction downsizing for a 32-bit primary port to a 16-bit secondary port. It may be necessary to break up the primary port transaction into multiple transactions on the secondary port if the resized length of the transaction is greater than the maximum secondary port transaction length; that is, 2^{A2X_BLW}.

Figure 2-14 Write Downsized Transaction

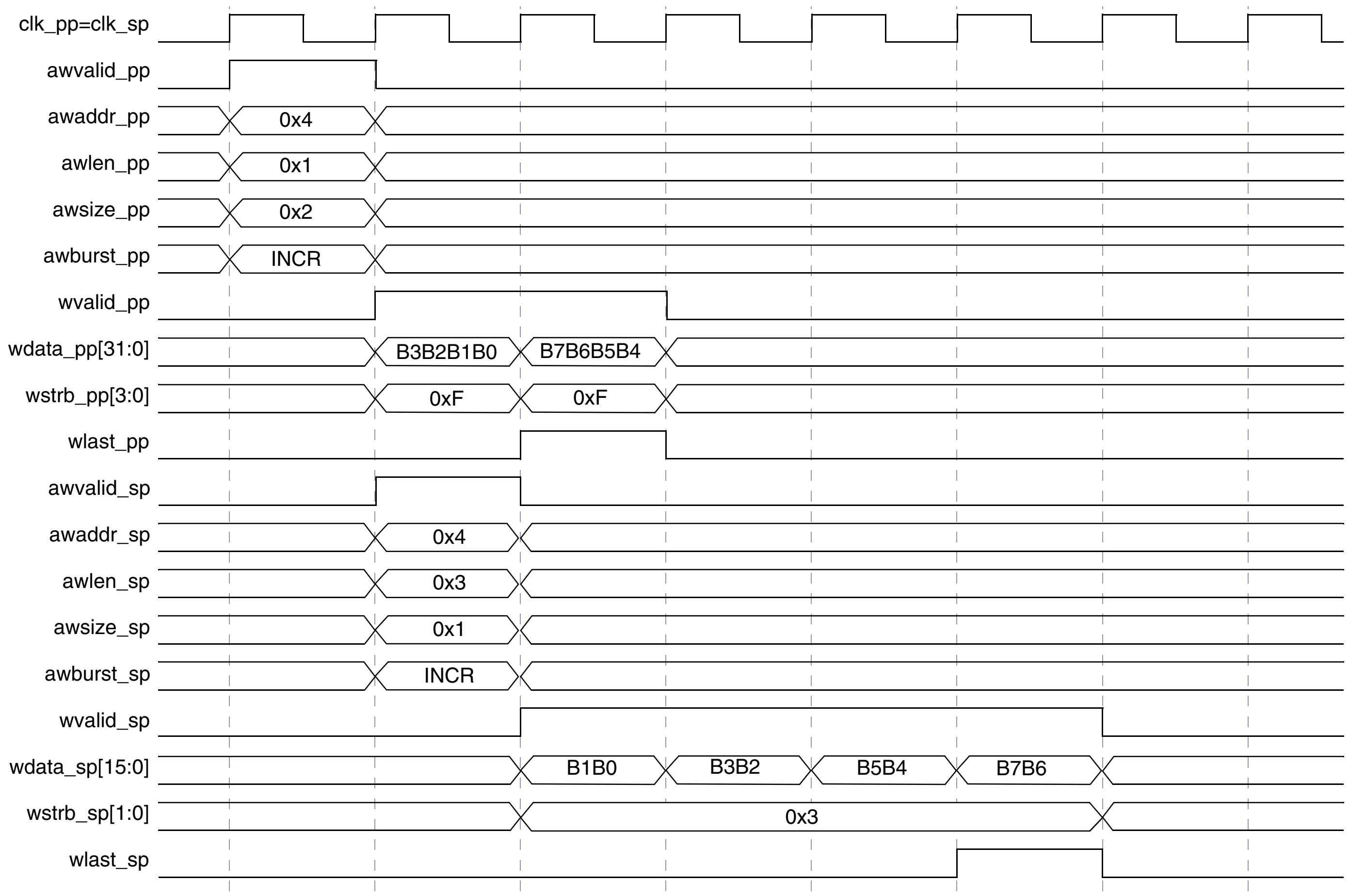


Figure 2-15 shows a downsized transaction where the primary port transaction is broken into two secondary port transactions. In this example, the maximum transaction length is 4; that is, A2X_PP_BLW is assumed to be 2.

Figure 2-15 Write Downsizing Transaction (A2X_PP_BLW=2)

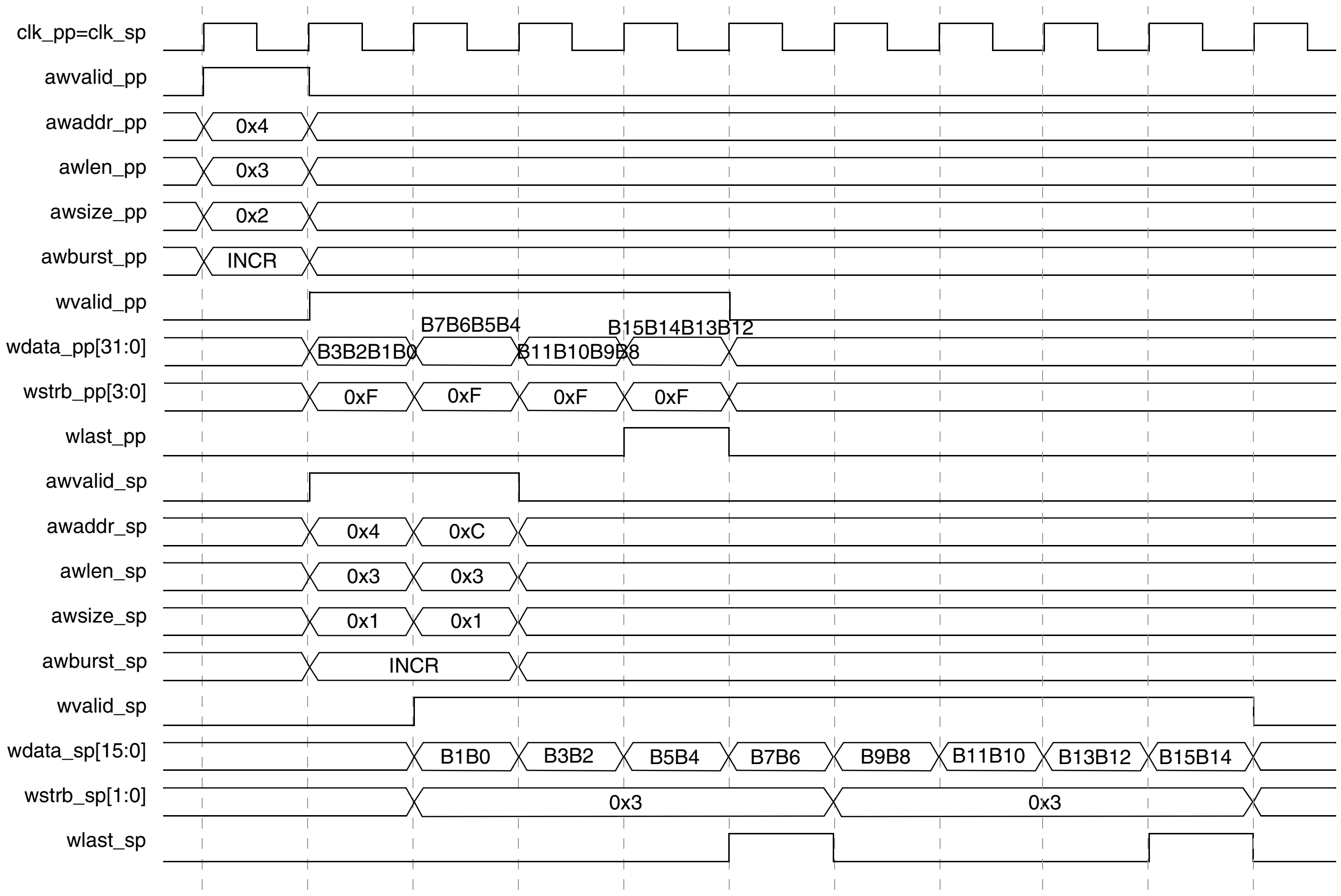
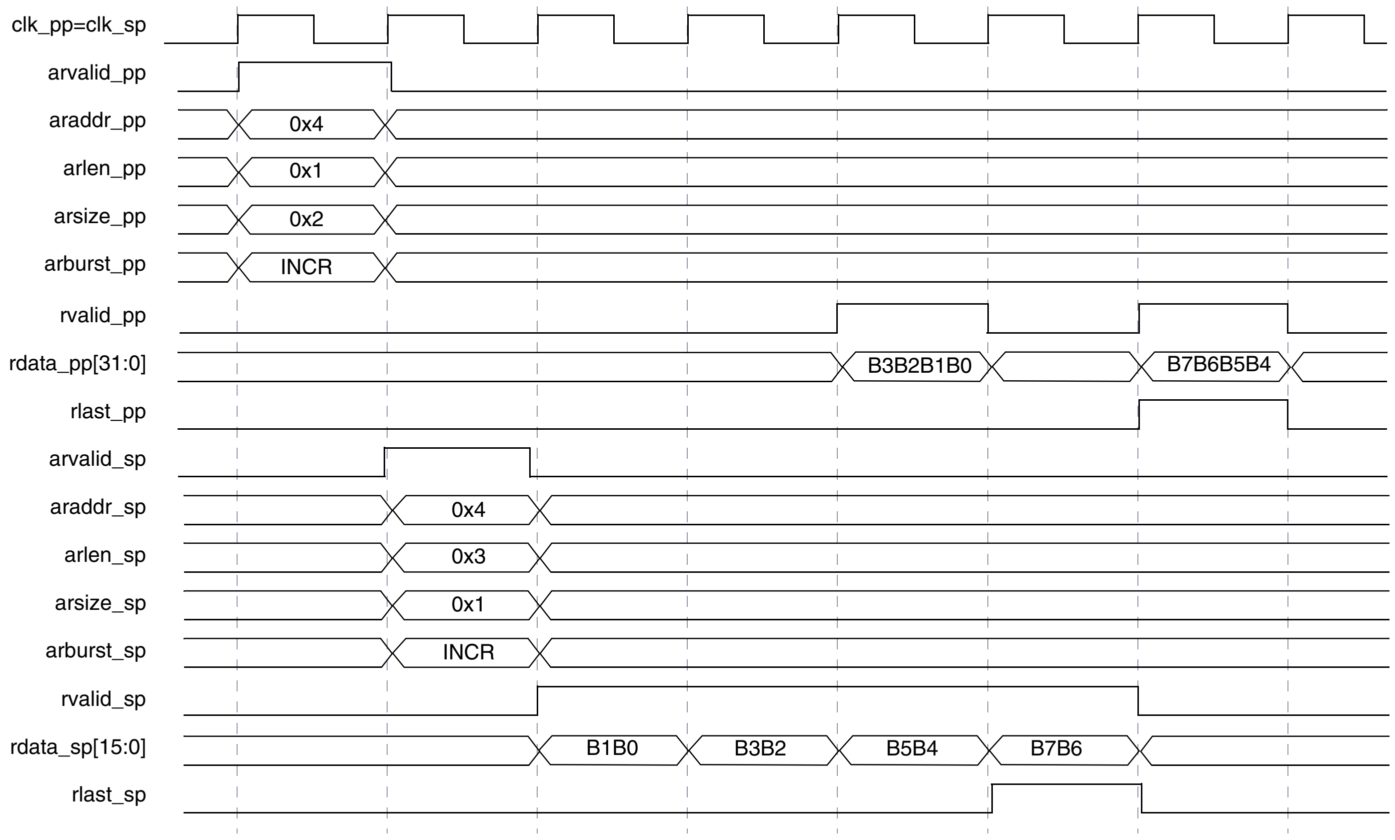


Figure 2-16 shows a read transaction downsized from a primary port data width of 32 bits to a secondary port data width of 16 bits. In this example, the primary port drains at the same clock rate as the secondary port, which requires two clock cycles to buffer the primary port data.

Figure 2-16 Read Downsized Transaction

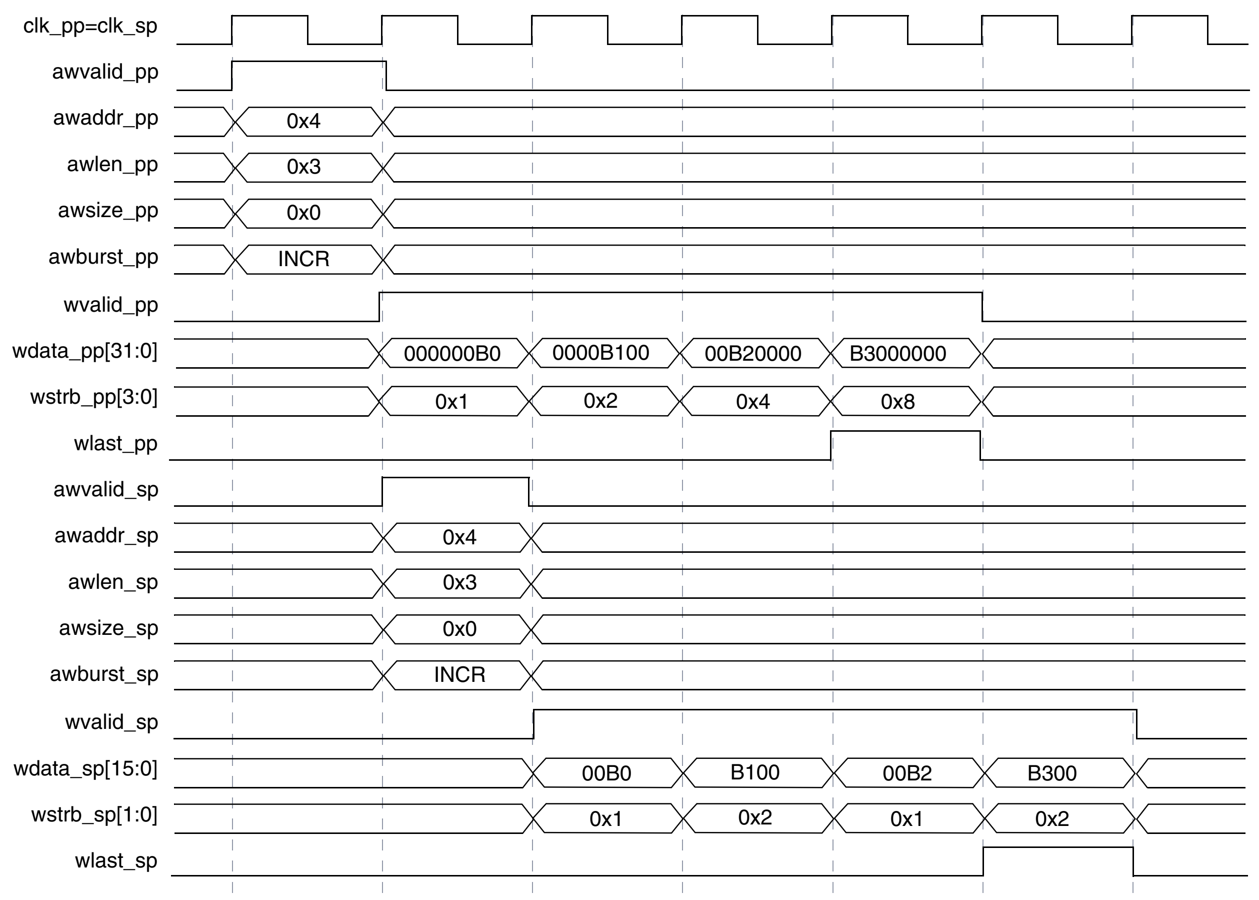


2.4.2.1 Sub-sized Transactions

If the primary port input transaction size is less than the maximum secondary port transaction size—that is, $\text{Log}_2(\text{SP\_DW}/8)$—the transaction is treated as sub-sized. The primary port address appears on the secondary port unaltered; only the date beats are mapped to the corresponding byte location.

Figure 2-17 shows the downsizing of a sub-sized transaction on a 32-bit primary port to a 16-bit secondary port.

Figure 2-17 Sub-Sized Transaction

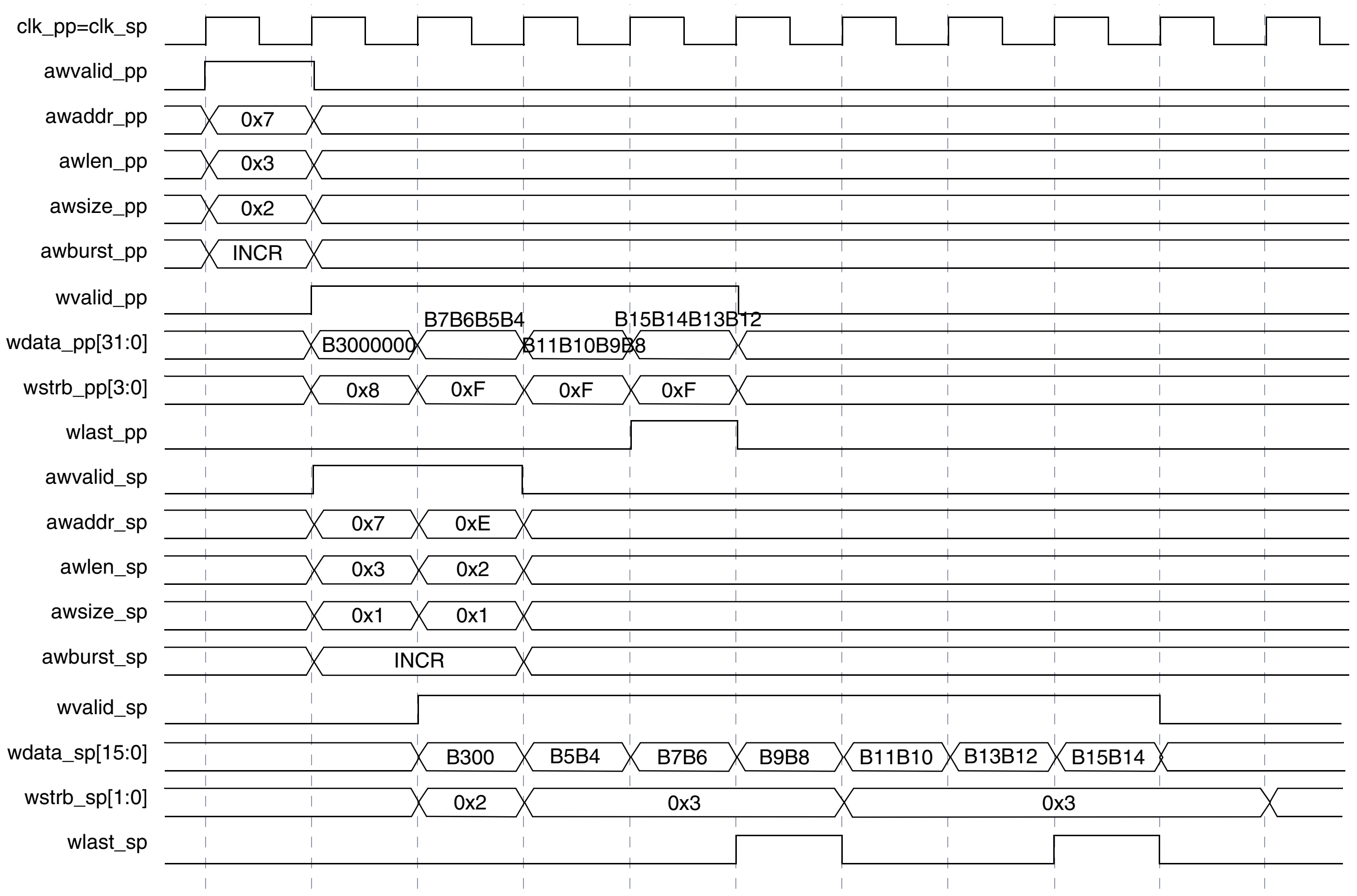


2.4.2.2 Unaligned Transactions

For downsized transactions broken into multiple secondary port transactions, the initial address issued on the secondary port is always the address received on the primary port. Any further addresses that are generated are aligned to the secondary port transaction size.

Figure 2-18 shows a downsized unaligned transaction with a maximum burst length of 4; that is, A2X_PP_BLW = 2.

Figure 2-18 Downsized and Unaligned Transaction



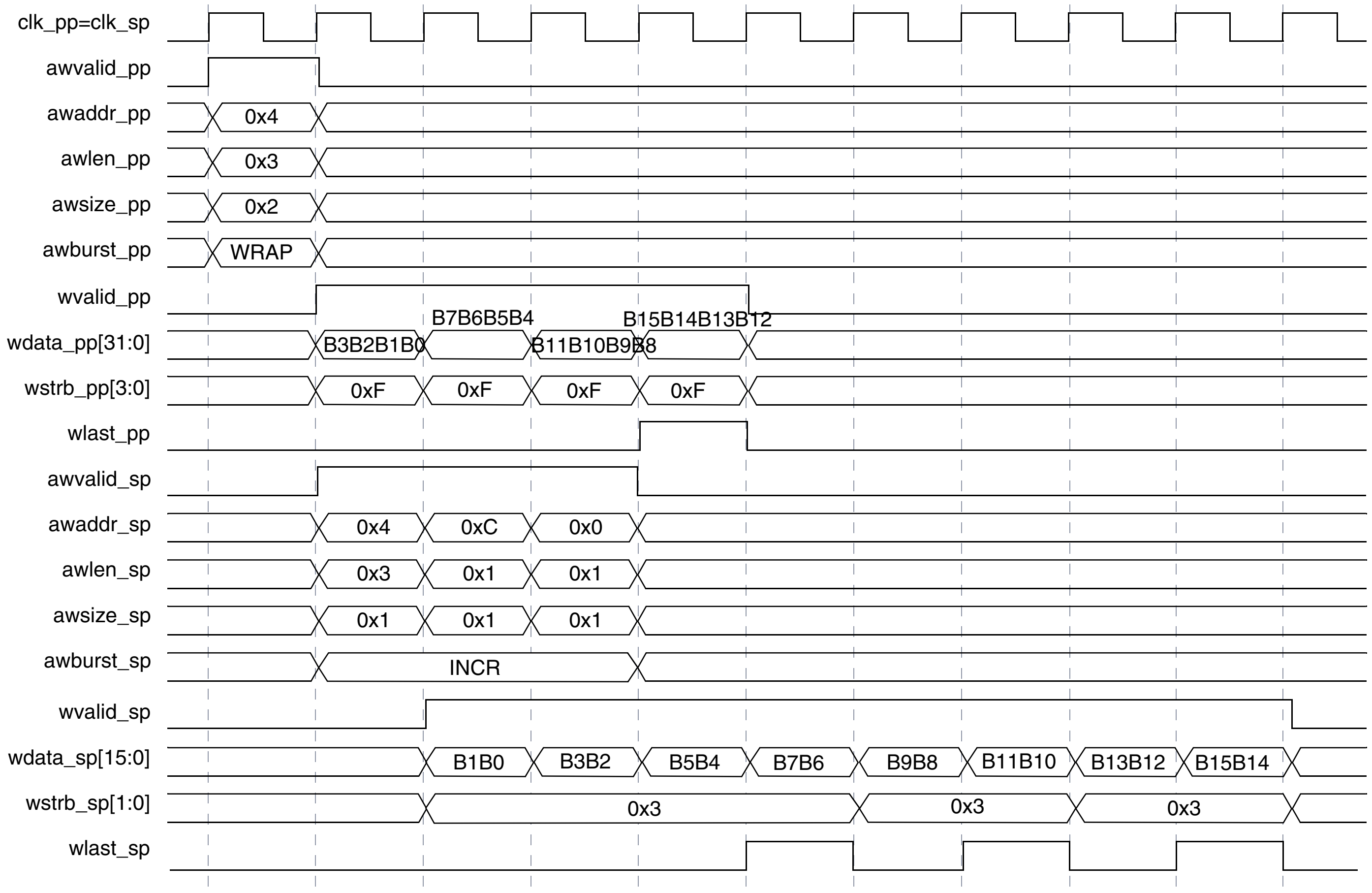
2.4.2.3 Burst Types

The primary port transaction is broken up based on the burst type and the maximum secondary port transaction length (2^{A2X_BLW}). Primary port transactions are broken up as follows:

- AXI INCR transactions with resized length greater than maximum length – broken into series of INCR transactions on secondary port
- AXI FIXED transactions with size greater than maximum secondary port size – broken into series of INCR transactions on secondary port
- AXI WRAP transactions – broken into series of INCR transactions controlled to observe address sequence dictated by wrap

Figure 2-19 shows an example of a downsized wrap transaction, which needs to be broken down into multiple INCR transactions. In this example, the maximum secondary port burst length is 4; that is, $2^{A2X\_BLW}(2)$.

Figure 2-19 Downsized WRAP Transaction

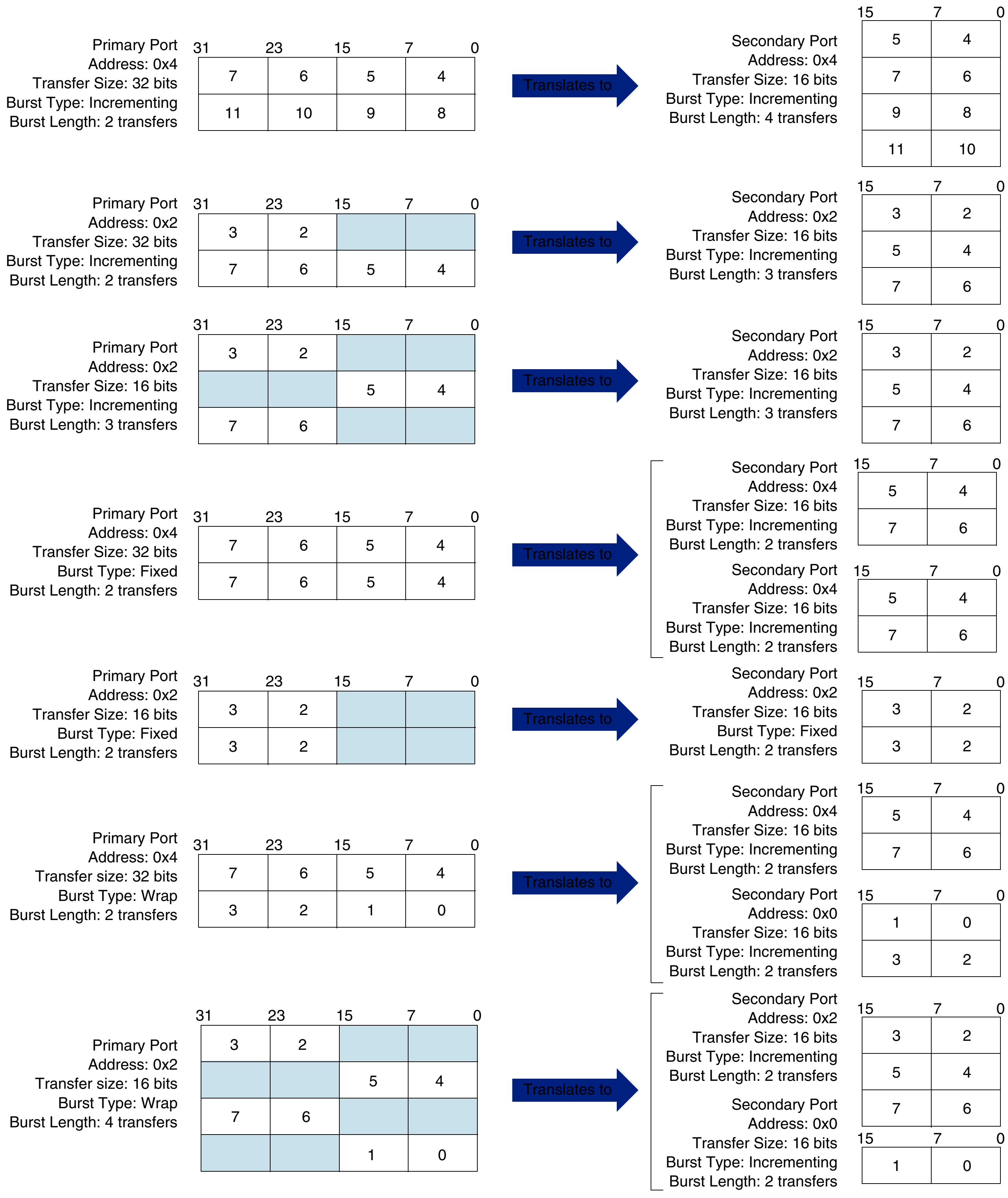


2.4.2.4 Downsized Transaction Examples

The diagrams in Figure 2-20 show examples of aligned and unaligned transfers on buses with different widths.

Each row in the figures represents a transfer. The shaded cells indicate bytes that are not transferred based on address and control information. The left hand size shows how the transfer appears on the Primary Port side and the right hand side shows how the transfers appear on the Secondary Port size.

Figure 2-20 Downsized Transaction Examples



2.5 Store-Forward Transactions

The DW_axi_a2x has a store and forward option to improve efficiency on the secondary bus and to provide a solution for applications that do not provide any data flow control.

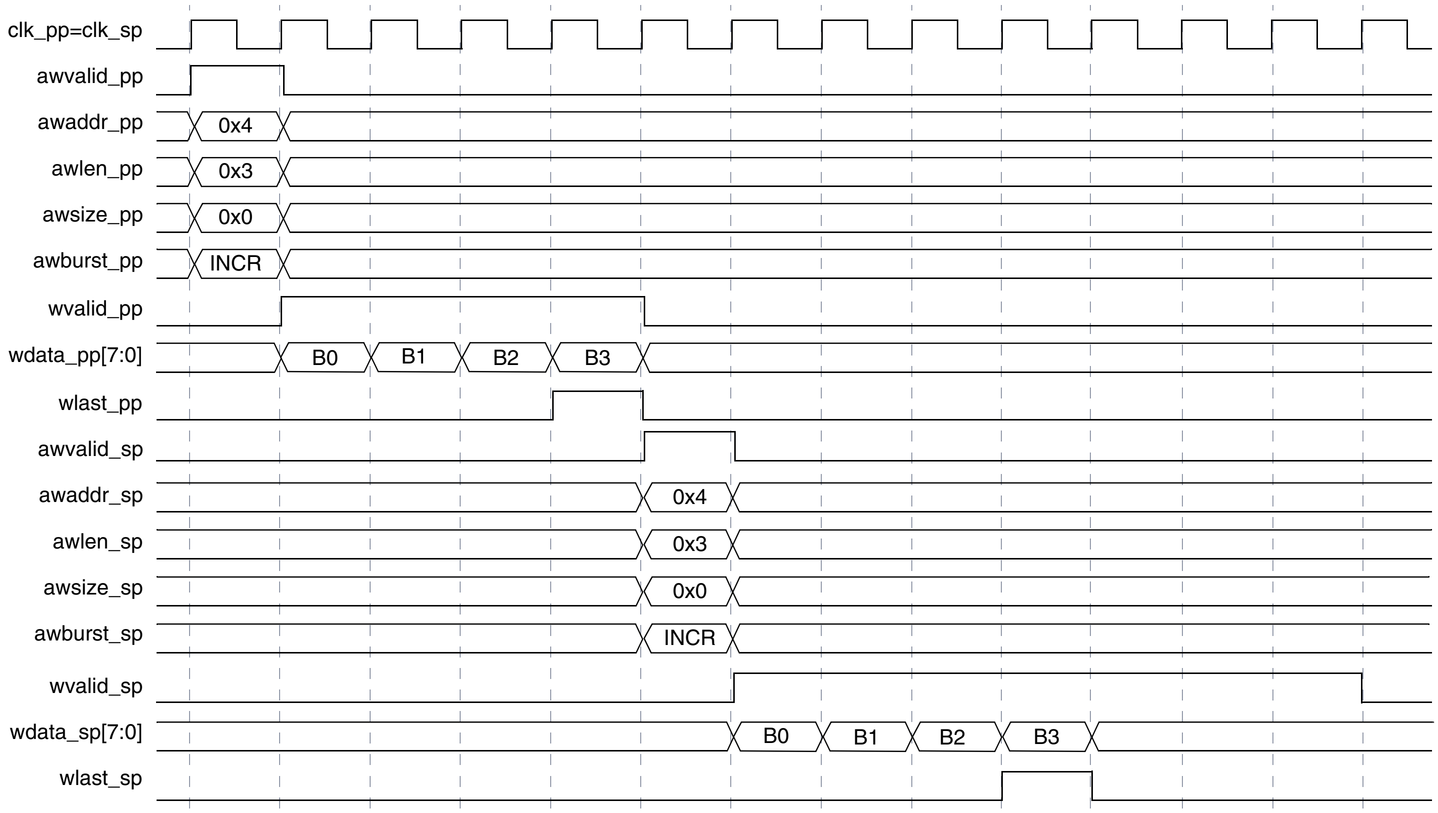
The store and forward feature ensures that data is always available in the write data buffer before issuing write address on the secondary AXI channel. This feature also ensures that there is enough free space in the read data buffer to accept the read data from the secondary AXI channel before issuing the read address to the secondary AXI channel.

2.5.1 Store-Forward Write

The Write AXI Store-Forward feature ensures that all write data for one AXI secondary port transaction is buffered before issuing an AXI address on the secondary port.

Figure 2-21 shows an AXI INCR transaction with an awlen of 0x3. Before the address is issued on the secondary port, all primary port data transfers must be captured.

Figure 2-21 Store-Forward Write



For transactions that are resized, the store-forward requirements are unchanged; that is, the store-forward control defines the number of secondary port write data transfers that must be internally buffered before the secondary port issues an address.

2.5.2 Store-Forward Read

The Read Store-Forward feature ensures that the A2X's read data buffer can consume all data beats before sending the AXI transaction. Hence it ensures that enough transfer space is available in the read data buffer before issuing a secondary port read address.

The DW_axi_a2x monitors the number of free spaces available in the read data buffer and the number of outstanding read data transactions to determine the amount of free buffer space available. When the number of free spaces is greater than or equal to the AXI secondary port transaction length, a read transaction is issued on the secondary port.

2.6 AXI Write Response Channel

The DW_axi_a2x offers two types of write response modes, which are selected by the A2X_BRESP_MODE configuration parameter:

- Bufferable – A2X_BRESP_MODE = 0
- Non-bufferable – A2X_BRESP_MODE = 1

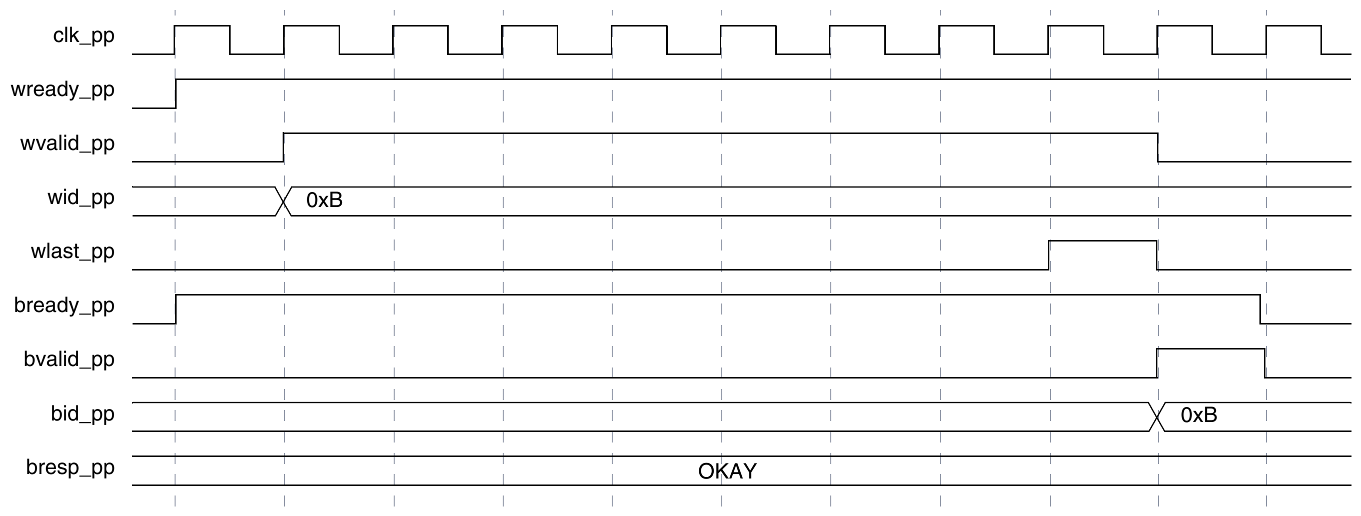
2.6.1 Bufferable Mode

In the bufferable mode of operation (A2X_BRESP_MODE = 0), the DW_axi_a2x always responds with an okay response after receiving the last write data beat for a given write transaction.

Note When DW_axi_a2x is configured in the AMBA 4 AXI or ACE-Lite mode (A2X_AXI_INTERFACE_TYPE = AXI4/ACELITE), bufferable responses are not supported.

Figure 2-22 shows the write response channel in bufferable mode.

Figure 2-22 Bufferable Response



2.6.2 Non-Bufferable Mode

In the non-bufferable mode of operation (A2X_BRESP_MODE = 1), the DW_axi_a2x always returns the response received from the secondary port.

In cases where the primary port is broken into multiple secondary port transactions, these responses returned on the secondary port are combined into one response on the primary port. If an error is detected on any of the secondary port responses, this error is returned to the primary port; otherwise the last secondary port response is returned.

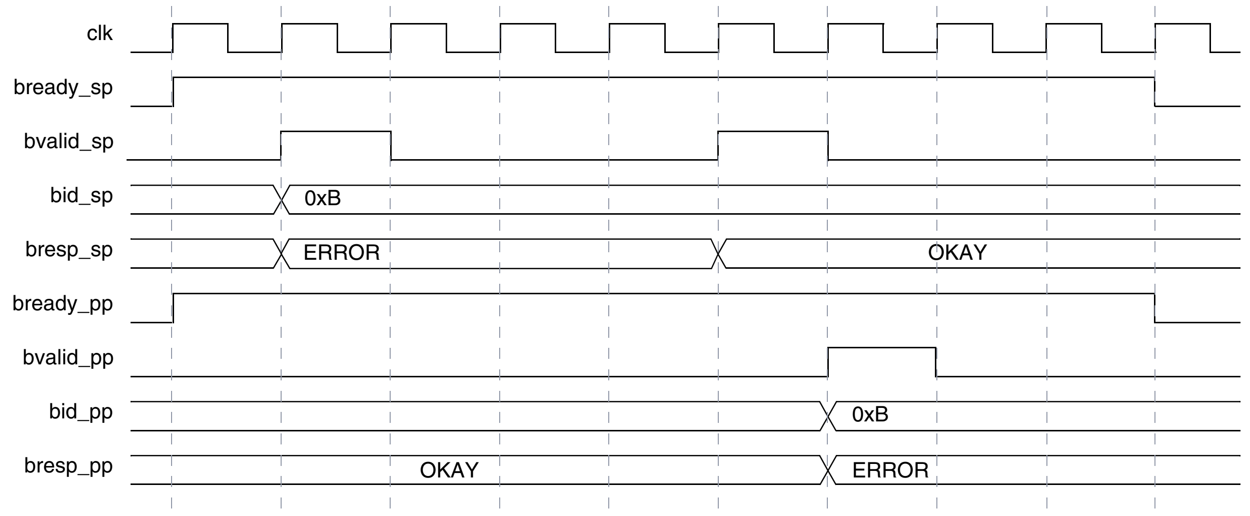
When combining the secondary port responses into primary port responses, the DW_axi_a2x gives priority to the responses type in the following order:

1. ERROR
2. DECERR
3. OKAY
4. EXOKAY

If any response is returned with ERROR, then the primary port response is of type ERROR.

Figure 2-23 shows the write response channel in non-bufferable mode. The example shows two secondary port transactions combined into one primary port response.

Figure 2-23 Non-Bufferable Response



2.7 Address Boundary Control

The *AMBA AXI Protocol Specification v1.0* states that bursts must not cross 4KB boundaries to prevent them from crossing between slaves and to limit the size of the address incrementer required within the slave.

If the DW_axi_a2x receives a transaction that does cross the 4K boundary and this transaction needs to be broken into multiple secondary port transactions, the DW_axi_a2x generates an incorrect address when the 4K boundary is reached.

2.8 Exclusive Accesses

An exclusive access that is resized to multiple smaller transactions cannot function as intended by the AXI protocol specification. Only exclusive accesses that do not need to be broken into multiple transactions on the DW_axi_a2x secondary port function as expected.

Therefore, for upsizing and downsizing configurations, the DW_axi_a2x cannot support:

- AXI transaction that has a primary port transaction size (awsize_pp/arsize_pp) larger than the maximum allowable transaction size (awsize_sp/arsize_sp) on the secondary port
- Wrap transaction of upsized or downsize configurations

Note The DW_axi_a2x must be configured for Non-Bufferable response mode (A2X_BRESP_MODE=1) to fully support AXI exclusive access. Configuring the DW_axi_a2x for Bufferable Mode (A2X_BRESP_MODE=0) returns OKAY responses after the last data beat is received on the primary port (wlast_pp).

2.9 Locked Transaction

When resizing a transaction, an unlocking command may be split into multiple transactions on the DW_axi_a2x. In this case, the DW_axi_a2x alters the LOCK bits of the commands such that only the last of the unlocking transactions indicates an unlocked transaction on the secondary port; all previous transactions indicate LOCKED. Otherwise, the slave can become unlocked before the original requested transaction has completed.

The DW_axi_a2x ensures that all outstanding locked transactions have completed before issuing the unlock command on the secondary port. It also ensures that the unlocking command has completed before issuing any further commands on the secondary port.

Selecting an unlocking transaction that does not require resizing improves the performance of the bridge, since the DW_axi_a2x does not have to break the transaction into multiple locked transactions followed by a unlocking transaction.

2.10 QoS, Region, and ACE-Lite Signals

The DW_axi_a2x provides QoS, region, and ACE-Lite signals for address channels. These signals enable you to transfer extra information when the AMBA 4 AXI or ACE-Lite interface is enabled.

For transactions downsized into multiple transactions, the downsized transfers have the same QoS, region, and ACE-Lite signals as the pre-downsized transfer.

2.11 AMBA 3 AXI Non-Protocol Sideband Signals

The DW_axi_a2x provides sideband signals for each channel that enables users to transfer extra information outside of the AXI protocol specification. The existence and widths of the sideband signals are individually configurable for each channel through the *SBW parameters.

For transactions downsized into multiple transactions, the downsized transfers have the same sideband signals content as the pre-downsized transfer. In other words, for two data beats on the primary port with sideband contents A and F, if each one is downsized to two beats, the result is that "A A F F" is issued on the secondary port.

For transactions that are upsized, the DW_axi_a2x forwards the sideband content of the last transfer used to create the upsized transaction.

No endian mapping is performed on the sideband signals as they pass through the DW_axi_a2x.

For transactions that require multiple secondary port write responses to generate one primary port response, the sideband data of the last secondary port response received is used to generate the primary port response.

2.12 AHB Transactions

The DW_axi_a2x provides an interface between an optional AHB block and the DW_axi_a2x secondary port.

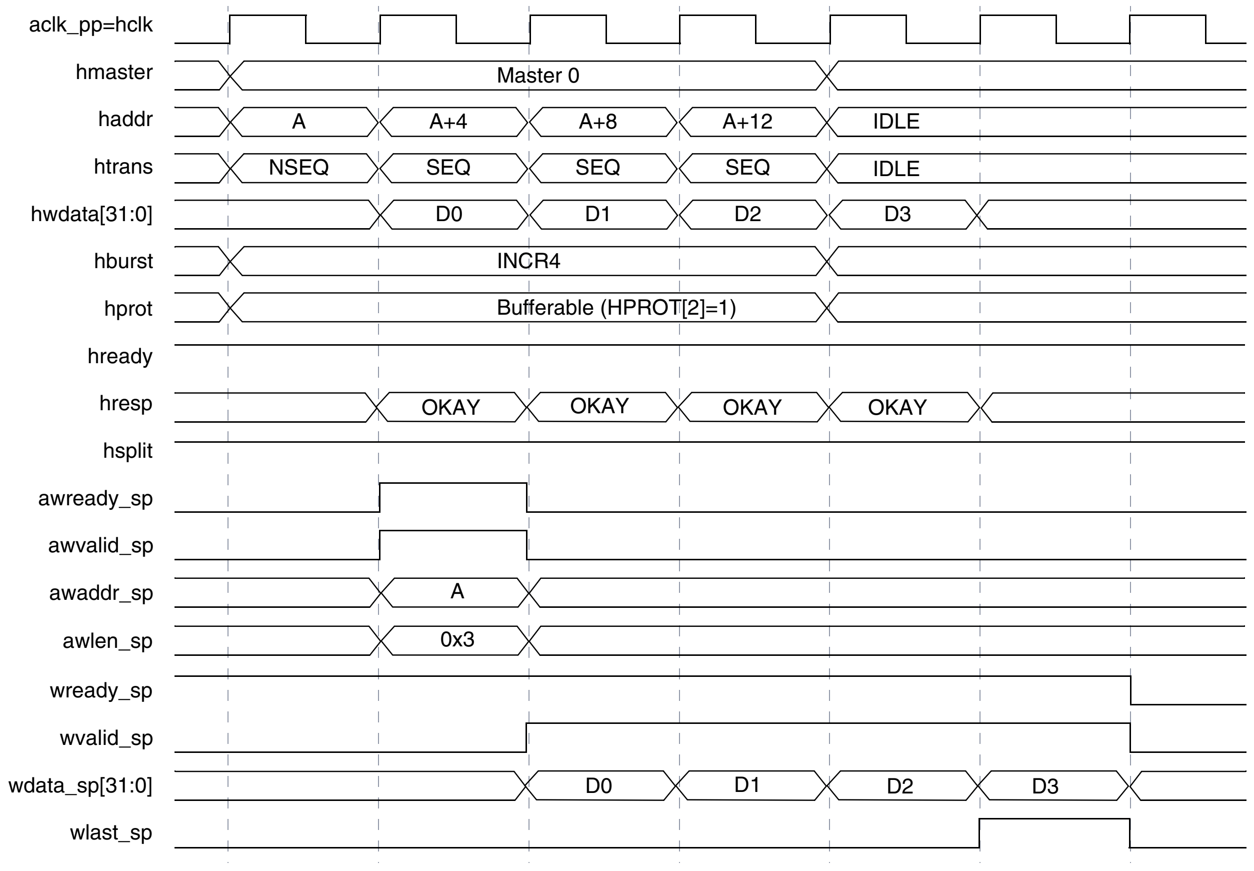
2.12.1 AHB Writes

The DW_axi_a2x is capable of translating AHB write transactions to AXI transactions on the DW_axi_a2x secondary port.

2.12.1.1 AHB Defined-Length Writes

Figure 2-24 shows an AHB write transaction translated into an AXI primary port write transaction.

Figure 2-24 AHB Write Transaction

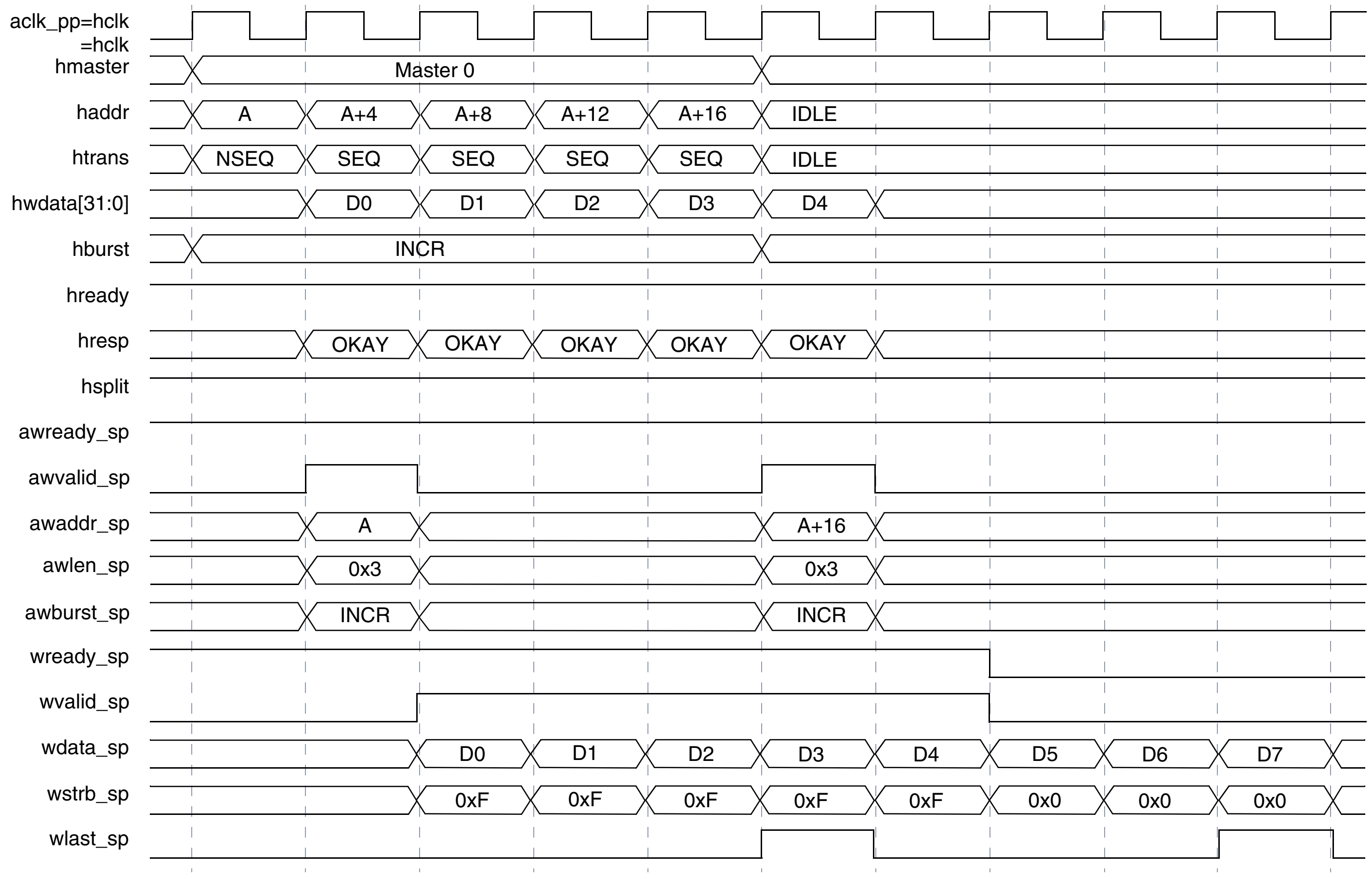


When a new transaction appears on the AHB bus, the DW_axi_a2x captures the address information of the first transaction—nseq—and issues this information on the AXI secondary port AW Channel. Each data beat for the INCR4 transaction is then captured and issued on the AXI secondary port W Channel.

2.12.1.2 AHB INCR Writes

Figure 2-25 shows an AHB INCR transaction with a write burst to AXI length conversion. During bufferable AHB write INCR transactions, the DW_axi_a2x uses the hincr_wbcnt_m port or the hardcoded A2X_HINCR_WBCNT_MAX parameter to generate the secondary port AXI write length (awlen). In this example, the A2X_HINCR_WBCNT_MAX has a value of (2^2)4. Once four write transactions have executed, the DW_axi_a2x generates another address with an awlen of 3.

Figure 2-25 AHB Write INCR Transaction



In Figure 2-25, the AHB INCR transaction results in five data transfers, which translates to two AW transactions on the secondary port—both with awlen of 3. Since only five data beats are sent by the AHB, the DW_axi_a2x sends the remaining data transfers for the second transaction on the secondary port with strobed data beats. If the INCR transaction completes before the first four beats are received, then the second address is not generated.

If the DW_axi_a2x uses the hincr_wbcnt_m* port, the DW_axi_a2x takes the value on the hincr_wbcnt_m* port when a new AHB INCR transaction is detected on the AHB bus. The example shows the initial transaction length of four beats; thus another transaction is generated after the first four data beats are captured. When capturing the address information for address A+16, the DW_axi_a2x samples the hincr_wbcnt again and uses that value for its AXI secondary port length—awlen_sp.

The A2X_HINCR_WBCNT parameter or the hincr_wbcnt_m* port defines the number of secondary port AXI transactions up to a maximum of 2^{A2X_BLW}. This limitation occurs because the DW_axi_a2x does not generate multiple AXI SP addresses for an AHB INCR write; instead, the DW_axi_a2x captures the address from the AHB bus. Additionally, the DW_axi_a2x cannot generate an AXI length greater that 1K bytes, as this is in violation of the AHB protocol.

2.12.1.3 AHB Write Buffer Full Response

If an AHB write transaction is issued when the write address or write data buffer is full, then the DW_axi_a2x responds by driving hready low in non-split mode. Once the buffer is non-full, the DW_axi_a2x asserts hready, which allows the write transaction to complete.

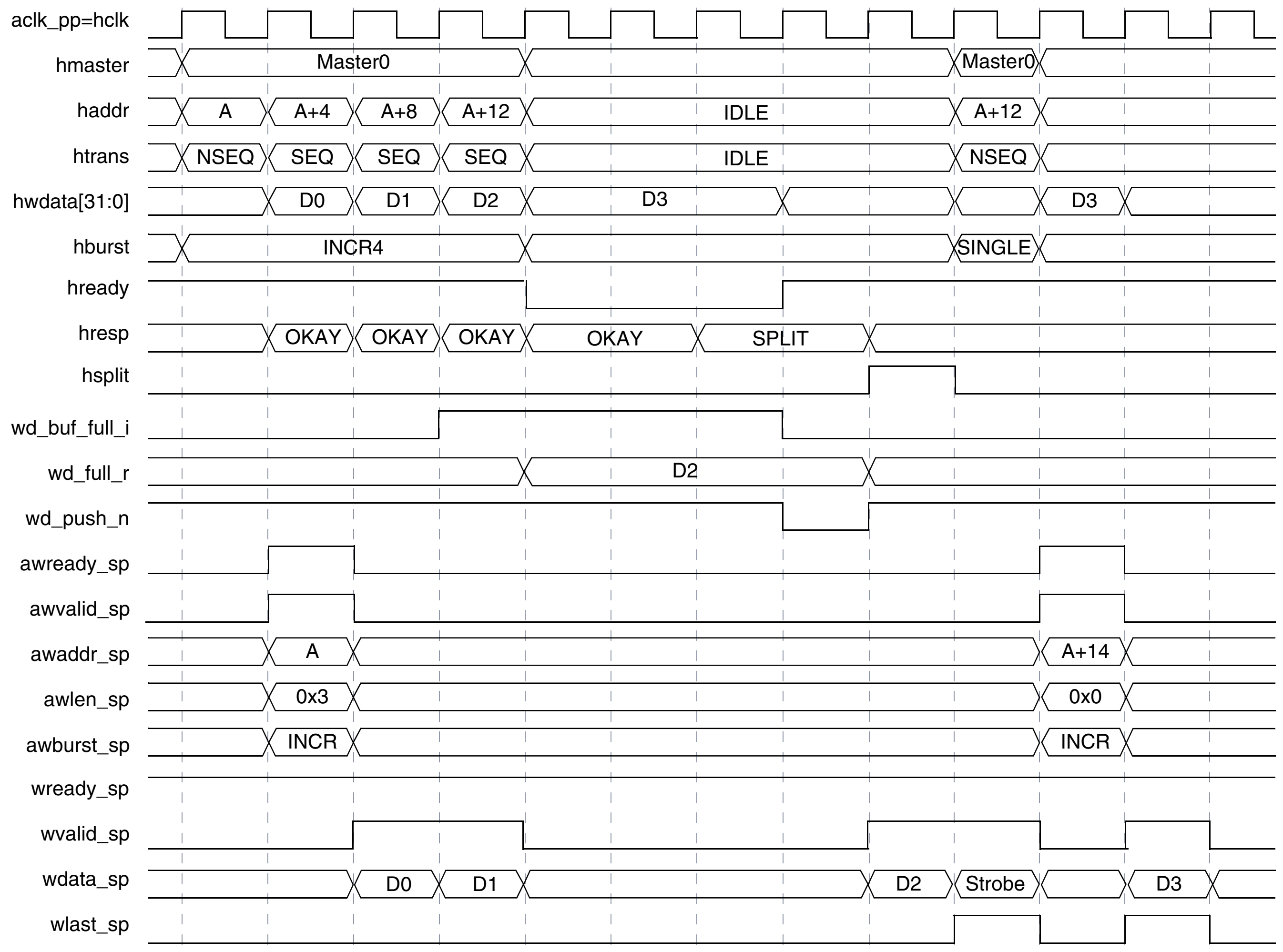
When configured for split mode, the DW_axi_a2x responds by initially driving hready low. If the DW_axi_a2x buffer is still full after a period of time—defined by the A2X_HREADY_LOW_PERIOD parameter—the following occurs:

1. DW_axi_a2x issues a split response for the AHB Master.
2. While issuing a split response, the DW_axi_a2x completes the AHB transaction by strobing the remaining data beats of the current transaction.
3. DW_axi_a2x issues a split response to any new write transaction while the write address or write data buffer remains full.

   An exception to this rule is if an AHB Master is returning for its non-bufferable response. In this case, the AHB write transaction is not split and the DW_axi_a2x returns the response received from the secondary port.
4. Once the buffer is empty, the DW_axi_a2x recalls all AHB masters previously split due to a buffer full condition. The DW_axi_a2x does not provide any priority control for the AHB masters and accepts the first write transaction received on the primary port after recall.

Figure 2-26 shows an example of an AHB INCR4 transaction that gets split due to a buffer full condition.

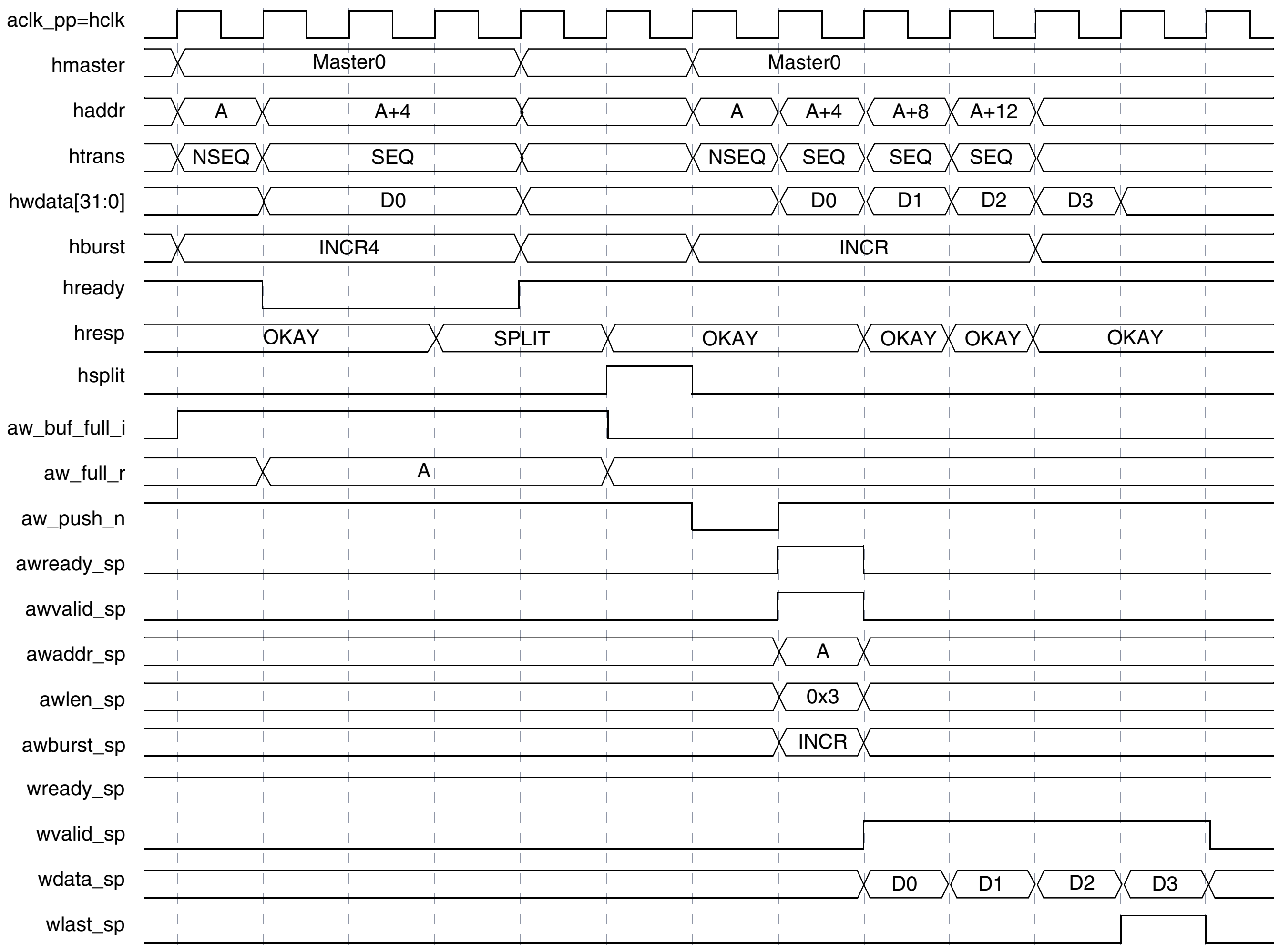
Figure 2-26 AHB Write Split – Buffer Full Condition



On the third data beat, the buffer becomes full and the DW_axi_a2x responds by driving hready low. In this example, the DW_axi_a2x issues a split response after hready is held low for two consecutive cycles. After issuing the split, the DW_axi_a2x strobes the last data beat of the INCR4 transaction and starts a new AXI transaction when the AHB Master returns from recall. The DW_axi_a2x adjusts the length of the transaction so that only remaining data beats are sent on the AXI Secondary Port when the AHB master returns to complete its INCR4 transaction.

Figure 2-27 shows an example of an AHB INCR4 write transaction split due to an address buffer full condition.

Figure 2-27 AW Buffer Full



In this example, the DW_axi_a2x returns a split response after the hready is held low for two consecutive cycles. Once a split response is returned, the DW_axi_a2x does not continue to store the address or the corresponding data. When the buffer becomes non-empty, the DW_axi_a2x recalls the AHB Master from the split.

If after splitting the AHB Master due to a write data or address buffer full, another AHB Master attempts a write and the buffer remains full. The DW_axi_a2x immediately issues a split response to this AHB Master. When the buffer becomes non-full, the DW_axi_a2x recalls all previously split AHB Masters.

An AHB master that was previously split due to a non-bufferable write transaction is recalled when its response is returned on the AXI secondary port. When this AHB Master returns to retrieve its response, the DW_axi_a2x returns the response, but it does not accept a new write transaction for the AHB master if the buffer is full. Thus, the DW_axi_a2x returns a split response to this new AHB write transaction.

In the example above, the AHB Master initially starts the transaction with a INCR4 burst type and when recalled, returns with a burst type of INCR. The DW_axi_a2x does not record any information about the original transaction and treats the recalled transaction as a new transaction.

2.12.1.4 AHB Write Response Mode

The DW_axi_a2x provides three types of write response through the A2X_BRESP_MODE parameter. These modes are:

- Bufferable A2X_BRESP_MODE(0)
- Non-bufferable A2X_BRESP_MODE(1)
- Dynamic A2X_BRESP_MODE(2)

2.12.1.4.1 Bufferable Mode - A2X_BRESP_MODE(0)

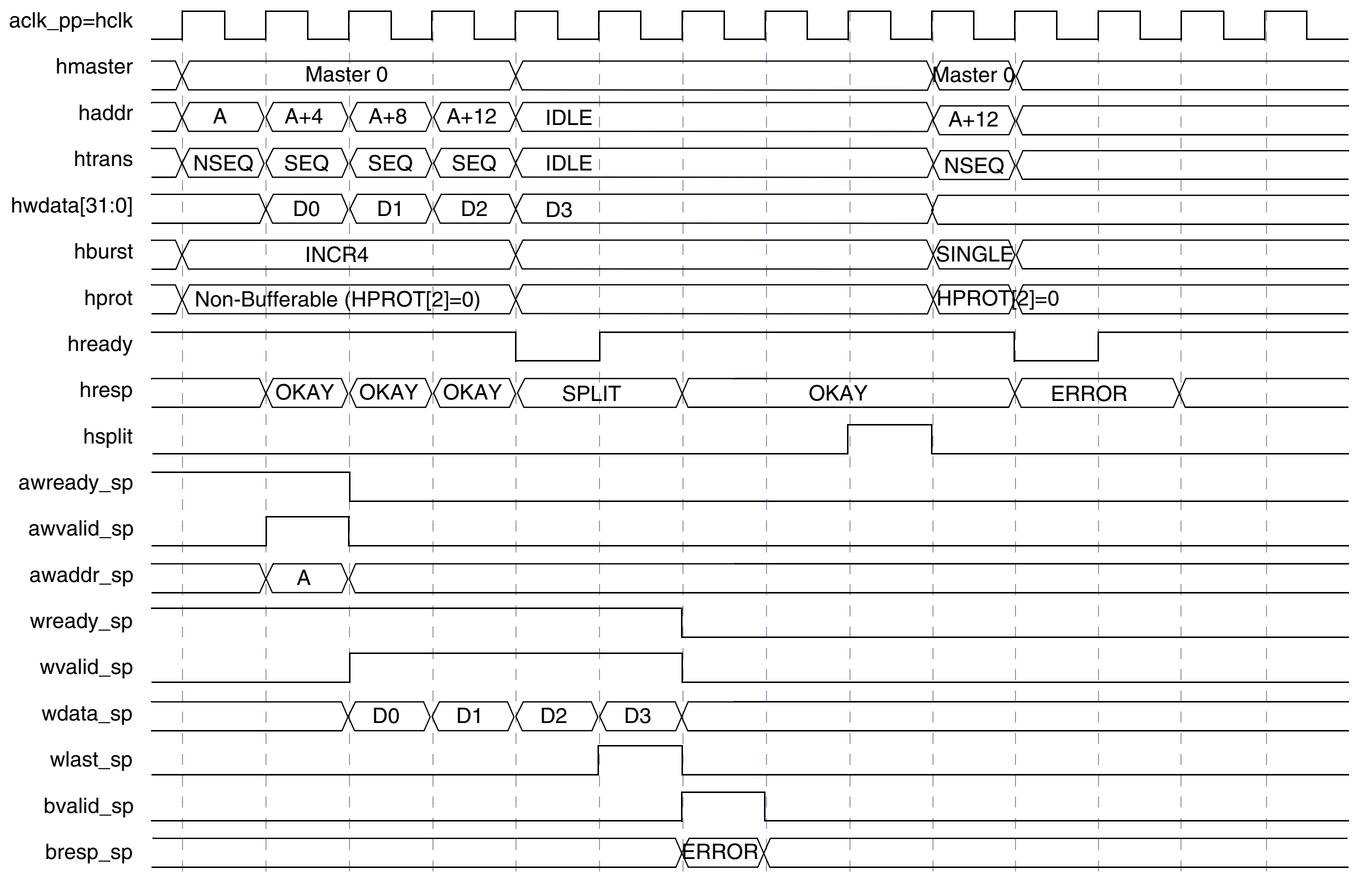
In the bufferable mode of operation, the DW_axi_a2x always responds with an okay response after receiving an AHB Write Transaction.

2.12.1.4.2 Non-Bufferable Mode - A2X_BRESP_MODE(1)

In the non-bufferable mode of operation, the DW_axi_a2x always returns the response received from the secondary port.

Figure 2-28 shows an AHB non-bufferable transaction. In split mode (A2X_AHB_SPLIT_MODE = 1), the DW_axi_a2x splits the last data beat of the AHB transaction. When the DW_axi_a2x receives the burst response for the write transaction, the DW_axi_a2x recalls the master from split. The burst response is then returned to the master when the AHB master returns.

Figure 2-28 AHB Write Protection Transaction (Split Mode)



All non-bufferable AHB INCR transactions are sent with an AXI length of 0. Thus, an AHB INCR transaction is split after the first data beat, and only when the write response is returned for that data transfer is the next INCR transaction accepted.

If the DW_axi_a2x is not configured for split mode (A2X_AHB_SPLIT_MODE = 0), the DW_axi_a2x responds by driving hready low on the last data beat. When the response is available, the DW_axi_a2x asserts hready and returns the response.

2.12.1.4.3 Dynamic Mode – A2X_BRESP_MODE(2)

The DW_axi_a2x always returns an OKAY response for bufferable transactions and the response received from the secondary port for non-bufferable transactions.

In cases where the primary port is broken into multiple secondary port transactions, responses returned on the secondary port are combined into one response on the primary port. If an error is detected on any of the secondary port responses, this error is returned to the primary port; otherwise the last secondary port response is returned.

2.12.2 AHB Reads

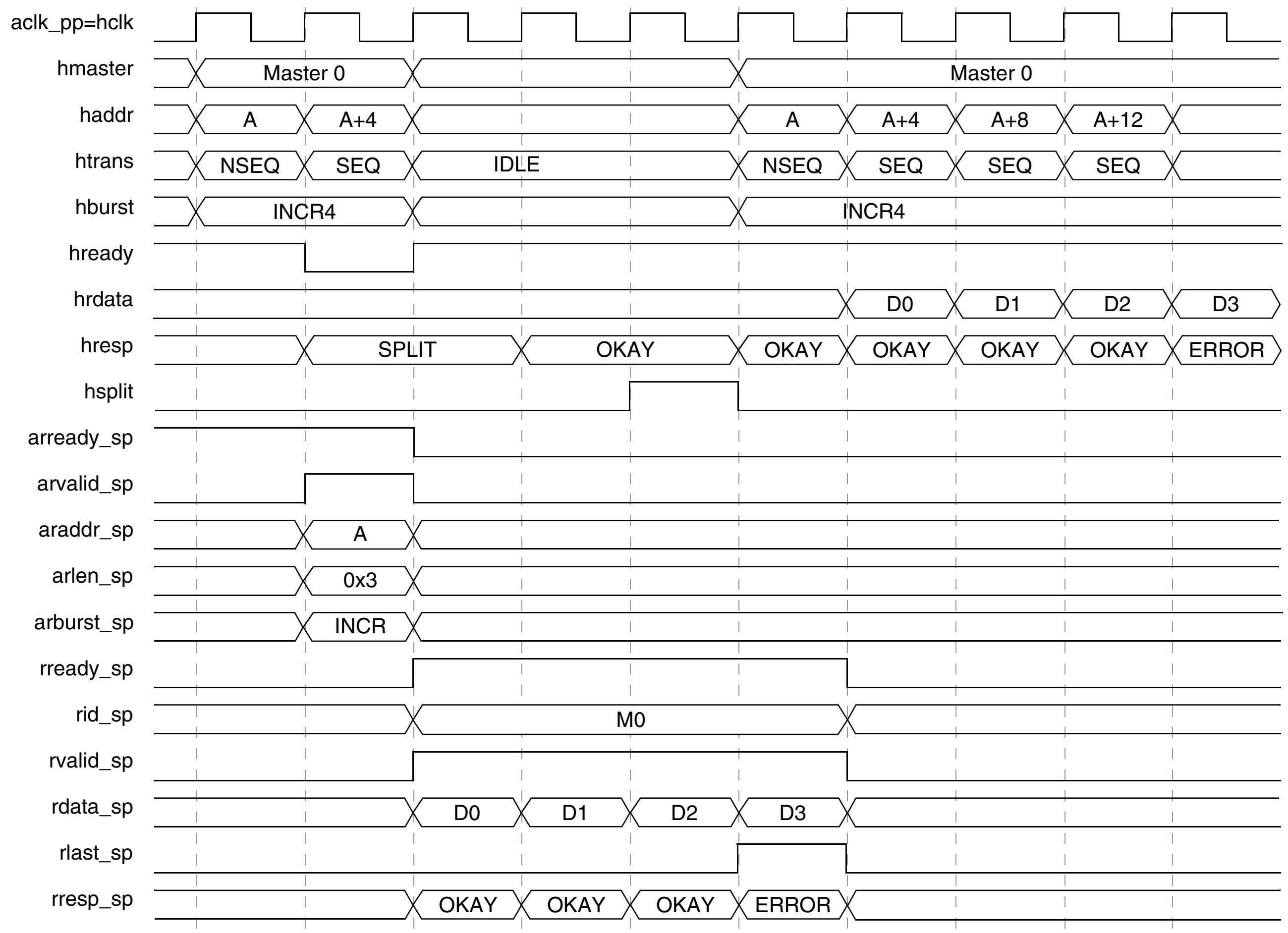
The DW_axi_a2x is capable of translating AHB read transactions to AXI transactions on the DW_axi_a2x secondary port.

2.12.2.1 AHB Defined-Length Reads

Figure 2-29 shows the translation of a fixed-length AHB transaction to an AXI transaction on the DW_axi_a2x secondary port. The read transaction is broken into three phases:

- Address capture
- Recall phase
- Read data phase

Figure 2-29 AHB Read Transactions



In split mode (A2X_AHB_SPLIT_MODE = 1), the DW_axi_a2x returns a split response during the address phase to the AHB master, but captures the address information to send a read address transaction on the DW_axi_a2x secondary port. When the read data is returned on the DW_axi_a2x secondary port, the AHB master is recalled from split (recall phase). Finally, when the AHB master returns from the split, the master read data is placed onto the AHB hrdata bus.

In cases where read data for a different master is received at the head of the data buffer before the read transaction completes for M0 (interleaved), the AHB master M0 is again split until read data for M0 is available again. Since the original read transaction was a fixed-length transaction, the DW_axi_a2x does not issue another AR request until all the read data is returned to the master; that is, rlast is detected on the primary port R channel.

If an error is returned to the AHB master during the data phase of a read transaction, the DW_axi_a2x flushes any remaining data beats from the read data buffer if the AHB master does not continue the transaction; that is, if one of the following occurs:

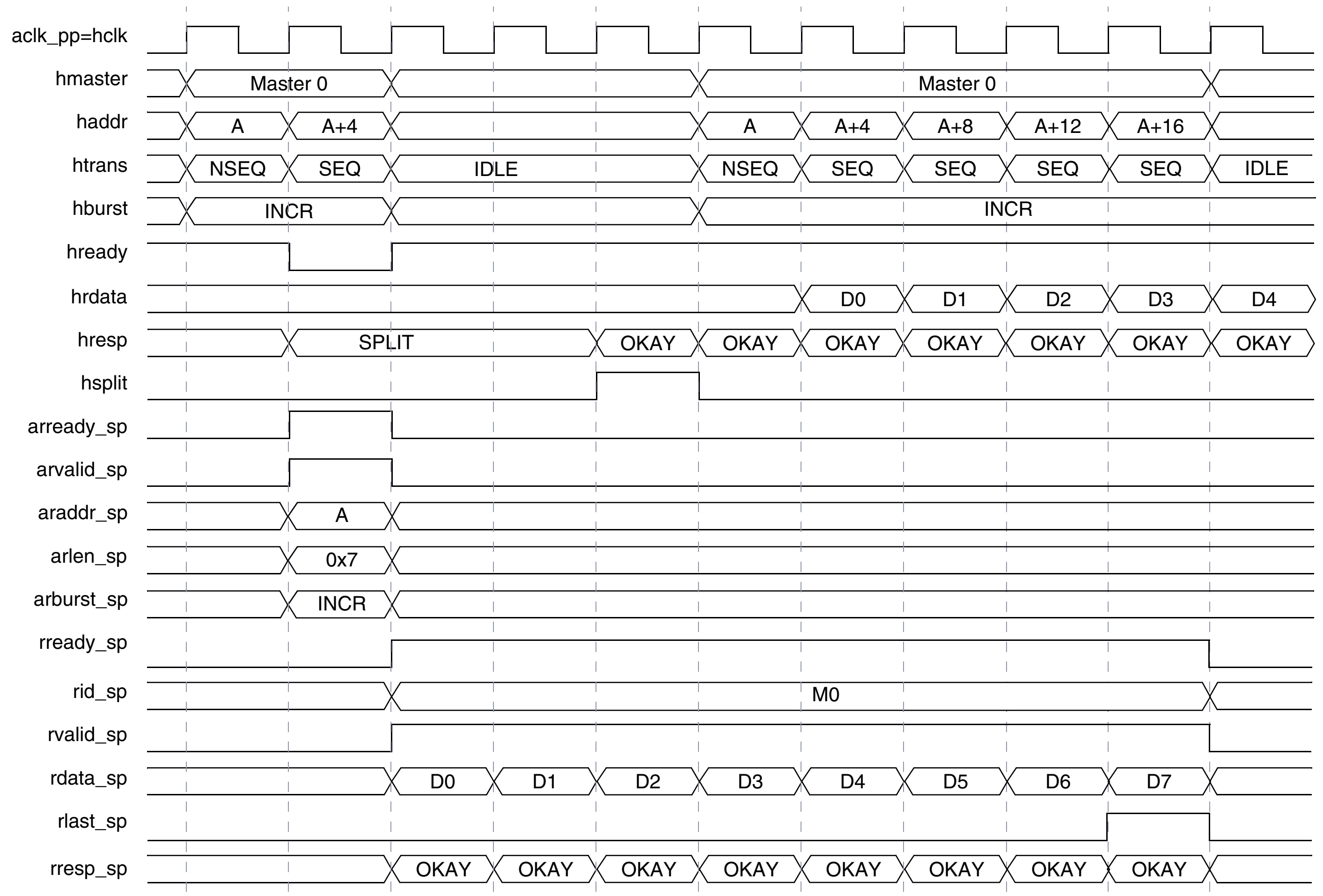
- htrans goes to NSEQ or IDLE
- hsel goes low
- A different AHB Master appears on the bus

In non-split mode (A2X_AHB_SPLIT_MODE = 0), the DW_axi_a2x responds to a read by driving hready low until the read data appears on the DW_axi_a2x_R channel. In this mode of operation, the DW_axi_a2x only issues an address on the AXI secondary port when all the read data for a previous transaction is returned on the AXI secondary port. Thus, when the DW_axi_a2x is flushing read data from the buffer, it may accept another AHB read transaction; however, the DW_axi_a2x does not issue that transaction on the AXI SP until all read data for the previous transaction are returned.

2.12.2.2 AHB INCR Reads

Figure 2-30 shows an AHB INCR read transaction; the protocol is similar to the example illustrated by Figure 2-29, except that the DW_axi_a2x uses the A2X_HINCR_RBCNT_MAX parameter or the hincr_rbcnt_m* port to generate the AXI secondary port read burst length (arlen_sp). In this example, the AXI read burst length is 8, but the AHB master INCR transaction only requests five data transfers. In this case the remaining data beats are flushed out of the read data buffer and guarding is removed.

Figure 2-30 AHB Read INCR Transactions



In cases where read data for a different master is received at the head of the data buffer and the AHB master M0 is still requesting read data, the AHB master M0 is again split until read data for M0 is available again. If the last read data beat is returned and the AHB master M0 is still requesting read data, the AHB master M0 is again split and a new AXI read is generated on the secondary port.

2.12.2.3 Flushing Data from Buffer

The DW_axi_a2x flushes read data from the read data buffer when one of the following occurs:

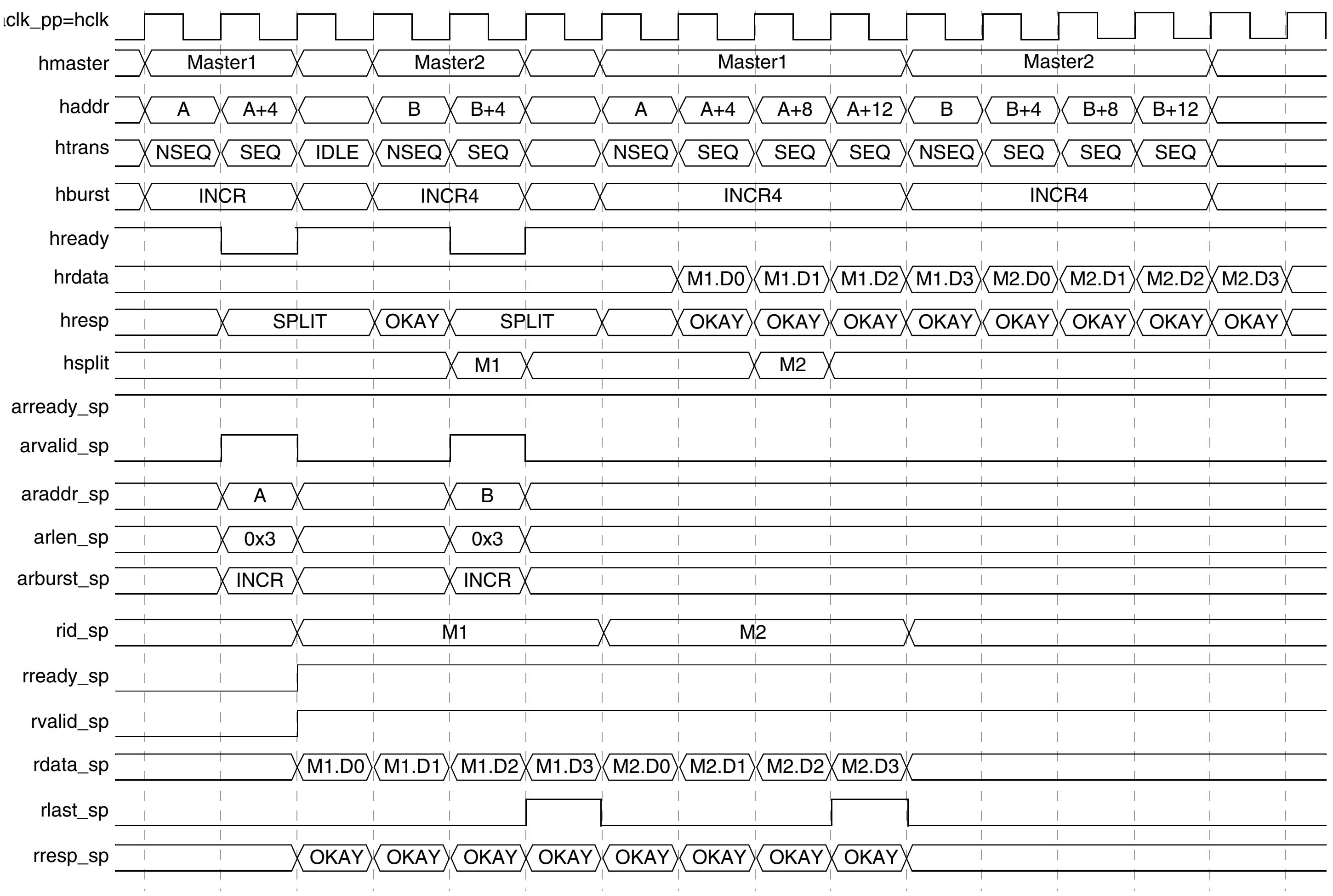
- AHB INCR read is less than read prefetch length (A2X_HINCR_RBCNT_MAX or the value captured on the hincr_rbcnt_m* port).
- Error is returned to AHB Master, and that AHB Master does not continue the read transaction; that is, htrans goes to IDLE or NSEQ.

2.12.2.4 AHB Read Streaming

In AHB Split mode, the DW_axi_a2x has the ability to stream read data from multiple AHB Masters back-to-back.

Figure 2-28 shows an example of the A2X streaming read data from two AHB Masters.

Figure 2-31 AHB Read Streaming



When read data is returned for the AHB Master 1, the DW_axi_a2x recalls that master from split. Once the master returns from split, the DW_axi_a2x starts to return the read data on the AHB read data bus. While returning the read data for AHB Master 1, the DW_axi_a2x issues a recall to AHB Master 2 when returning the second read data beat for Master 1. The AHB Master takes two clock cycles to appear on the AHB bus after being recalled, which gives time for the DW_axi_a2x to return the last data beat for Master 1 before responding with the Master 2 read data.

The DW_axi_a2x can only stream AHB INCR transactions back-to-back if the AHB Master generates a read INCR request equal to the A2X_HINCR_RBCNT_MAX parameter or the value captured on the hincr_rbcnt_m* when initially requesting the read transaction.

2.12.3 AHB Early Burst Termination (EBT)

The DW_axi_a2x is capable of handling early-burst termination (EBT) operations for read and write transactions.

The conditions in which a defined-length transaction can be early-burst terminated are as follows:

- Another AHB Master interrupts the current defined-length transaction before it completes
- HTRANS transitions to IDLE or NSEQ before the transaction is complete
- HSEL goes low

The DW_axi_a2x does not consider AHB INCRs in an EBT condition because AHB INCRs are of an undefined length; thus they cannot be early-burst terminated.

2.12.3.1 EBT Writes

If an EBT condition is detected during a write transaction, the DW_axi_a2x sends the remaining data transfers with strobed data.

If a defined-length write transaction is early-burst terminated, the DW_axi_a2x expects the terminated master to return and complete its transfer. When the master returns, the DW_axi_a2x adjusts the length of the transaction so that only remaining data beats are sent on the AXI secondary port. Thus, if an AHB INCR8 is early-burst terminated after five transfers, the DW_axi_a2x sends the first AXI transaction with an awlen_sp of 7, but with the last three data transfers strobed. When the terminated master returns to complete its transfer, the DW_axi_a2x generates an alen of 2 in order to complete the remaining three transfers from the AHB master.

2.12.3.2 EBT Reads

Only data at the head of the read buffer can be returned on the AHB channel. Thus, if a read is early-burst terminated during the data phase of the fixed read transaction, the terminated master must return to complete the read transaction.

2.12.4 AHB Locking

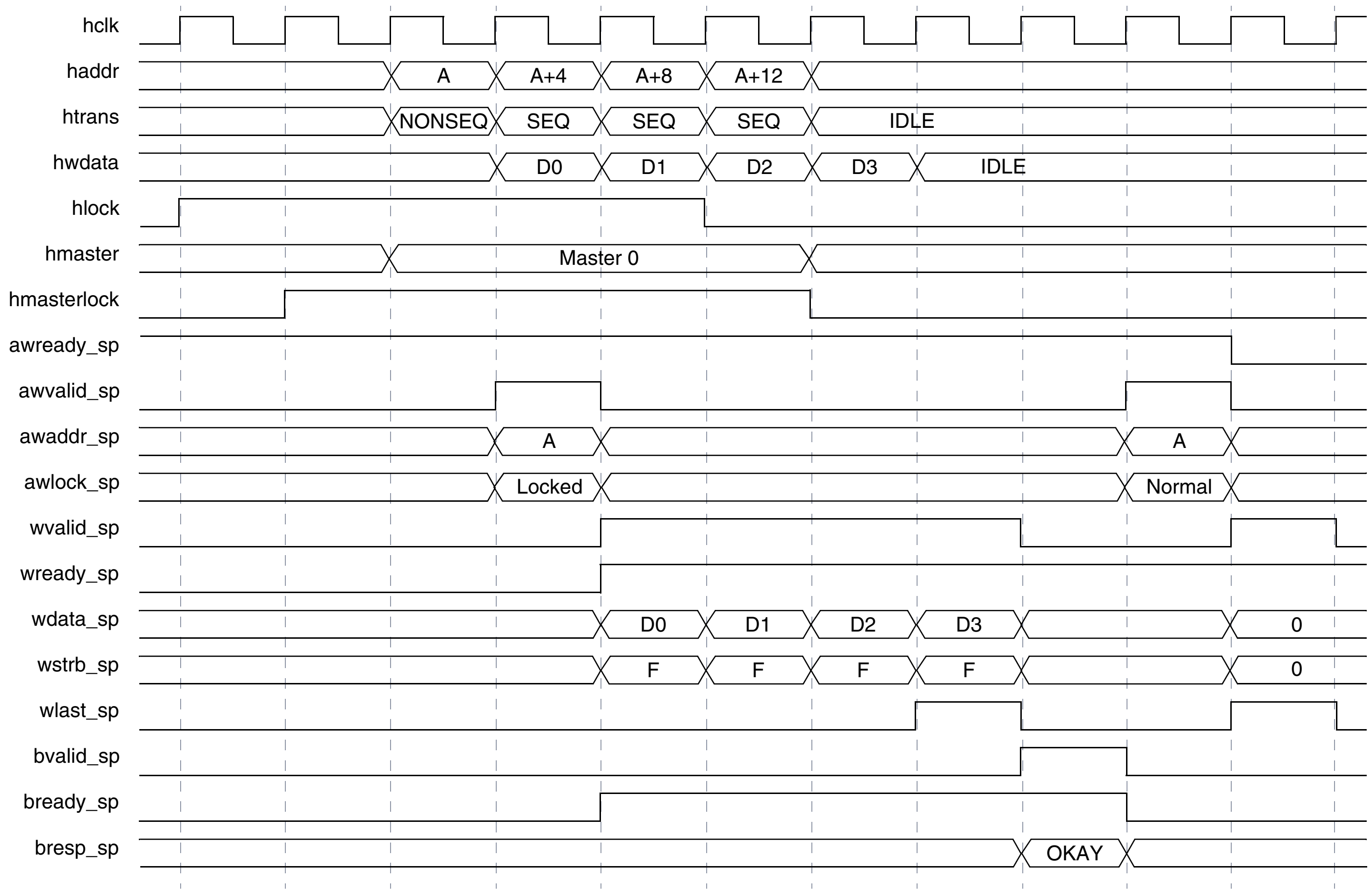
A locked transfer is indicated by the hmasterlock signal. For the AXI, a sequence of locking transactions is completed with an unlocking transaction. However, the DW_axi_a2x cannot predict when the last functional transaction will be in an AHB locked sequence. Therefore, to complete a locked sequence on AXI, the DW_axi_a2x must issue an unlocking transaction in order to finish the sequence of unlocking transactions.

The DW_axi_a2x must ensure that all outstanding transactions have completed on the secondary port before generating an unlock transaction. It must also ensure that the unlock transaction has completed before generating any new transactions.

If a new transaction appears on the AHB bus before an unlock transaction has completed, the DW_axi_a2x responds to this transaction by driving hready low. Only when the unlock transaction has completed does the DW_axi_a2x assert hready.

Figure 2-32 shows an AHB locked access.

Figure 2-32 AHB Locked Access



An unlocking transaction is generated on the falling edge of hmastlock or hsel or if a different master appears on the AHB bus. The unlock transaction is a single write transaction with size of 0x0 and all write strobes turned off. This transaction has no functional effect on the destination slave, other that completing the locked sequence on AXI. The address used for the AXI secondary port unlock transaction is the first successful read/write address used by the DW_axi_a2x locking sequence.

2.12.4.1 AHB-Locking Deadlock

To prevent deadlock in AHB mode, the DW_axi_a2x must accept and store:

- Responses for every AHB master
- All read data for every AHB master

When locking is enabled, the DW_axi_a2x sets the depth of the read data buffer to:

$$\text{number\_of\_masters} \times \text{maximum\_burst\_length}$$

Maximum burst length is defined as 16 for AHB locked configurations. If the A2X_HINCR_RBCNT_MAX parameter is greater than 4, then the maximum burst length equals $2^{A2X\_HINCR\_RBCNT\_MAX}$. This ensures that the DW_axi_a2x can accept all outstanding read data before sending a locking transaction on the secondary port.

When sending a locked transaction on the AHB, the locking transaction must not cross the AXI slave boundary. If during a locked sequence the AHB master crosses the AXI slave boundary without issuing an unlock transaction to that AXI Slave, the DW_axi_a2x will not generate a unlock transaction for that AXI slave.

Note Under the above circumstances, the AXI/AHB bus fabric may enter DEADLOCK.

2.12.5 AHB Boundary Control

The AHB fabric requires that bursts cannot cross a 1 kB address boundary. However, when sending an AHB INCR transaction, the DW_axi_a2x uses one of the following to determine the length of the AXI transaction:

- A2X_HINCR_WBCNT_MAX
- A2X_HINCR_RBCNT_MAX parameter
- hinct_wbcnt_m* or hincr_rbcnt_m* ports

While the generated AHB INCR transaction does not cross the 1 kB boundary, the AXI secondary port transaction *can* cross the AXI 4K boundary. In this case, the DW_axi_a2x adjusts the length of the AXI secondary port transaction so that the transaction does not cross the 1k boundary.

As an example:

- AHB starts an INCR transaction at address 0x3FFA
- A2X_HINCR_WBCNT_MAX has a value of 3 (Length(8) = $\text{Log}_2 3$)

This situation results in the last two transactions crossing the 1K boundary. In this case, the DW_axi_a2x adjusts the secondary port length to 6 so that the transaction only goes up to address 0x3FFF.

2.12.6 AHB Lite

In AHB-Lite mode (A2X_AHB_LITE_MODE), the DW_axi_a2x does not issue split responses. In this mode the hready signal is driven low for all read transactions until the read data is returned, as well as for the last data transfer of a non-bufferable write transaction.

In AHB-Lite mode, the number of AHB Masters that the DW_axi_a2x can support is one.

2.13 Endian Conversion

The DW_axi_a2x provides endian mapping from the AHB/AXI primary port to the AXI secondary port with respect to the transaction size. This section outlines the key features of the DW_axi_a2x endian mapping.

2.13.1 Endian Key Features

Key features of endian conversion are as follows:

- Parameters:
  - A2X_PP_Endian – selects Primary Port endianness
  - A2X_SP_Endian – selects Secondary Port endianness
- Endian transformations supported on Primary Port:
  - Little Endian (LE) – AXI and AHB
  - Big Endian-8 (BE-8) Invariant – AXI Only
  - Big Endian-32 (BE-32) Invariant – AHB Only
  - Big Endian-A (BE-A) Address Invariant – AHB Only
- Endian transformations supported on Secondary Port:
  - Little Endian (LE)
  - Big Endian-8 (BE-8) Invariant – AXI Only

2.13.2 DW_axi_a2x Memory Access Table

Table 2-1 shows the effect of LE, BE-8, BE-32 and BE-A on a 64-bit-wide bus. Table 2-1 also outlines the differences between word access, half-word access, and byte address for the different endian implementations. Definitions for items in Table 2-1 are as follows:

- A*Num* – Byte Access to address[2:0] = *num*
- A*Num*-Byte – Byte of word/half-word access to address[2:0] = *num*
- MS – Most significant byte
- MS-1 – Second most significant byte
- LS+1 – Second least significant byte
- LS – Least significant byte

Table 2-1 Endian Memory Access Table

Pins	Byte Access				Half-Word Access				Word Access			
	LE	BE-8	BE-32	BE-A	LE	BE-8	BE-32	BE-A	LE	BE-8	BE-32	BE-A
63:56	A7	A7	A4	A0	A6:MS	A6:LS	A4:MS	A0	A4:MS	A4:LS	A4:MS	A0
55:48	A6	A6	A5	A1	A6:LS	A6:MS	A4:LS	A1	A4:MS-1	A4:LS+1	A4:MS-1	A1
47:40	A5	A5	A6	A2	A4:MS	A4:LS	A6:MS	A2	A4:LS+1	A4:MS-1	A4:LS+1	A2
39:32	A4	A4	A7	A3	A4:LS	A4:MS	A6:LS	A3	A4:LS	A4:MS	A4:LS	A3
31:24	A3	A3	A0	A4	A2:MS	A2:LS	A0:MS	A4	A0:MS	A0:LS	A0:MS	A4
23:16	A2	A2	A1	A5	A2:LS	A2:MS	A0:LS	A5	A0:MS-1	A0:LS+1	A0:MS-1	A5

Table 2-1 Endian Memory Access Table

Pins	Byte Access				Half-Word Access				Word Access			
	LE	BE-8	BE-32	BE-A	LE	BE-8	BE-32	BE-A	LE	BE-8	BE-32	BE-A
15:8	A1	A1	A2	A6	A0:MS	A0:LS	A2:MS	A6	A0:LS+1	A0:MS-1	A0:LS+1	A6
7:0	A0	A0	A3	A7	A0:LS	A0:MS	A2:LS	A7	A0:LS	A0:MS	A0:LS	A7

Table 2-1 is taken from the *ARM1156T2F-S Technical Reference Manual* and modified to include the AHB address invariance scheme (BA).

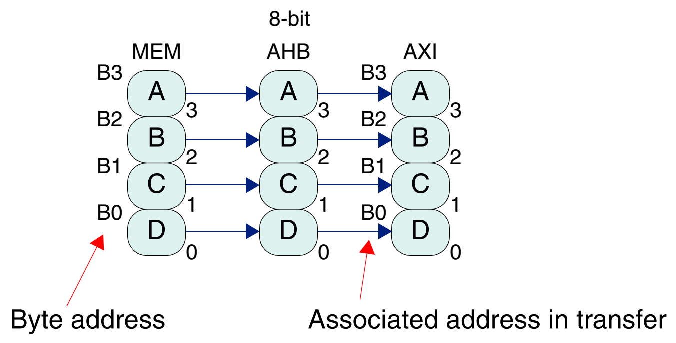
2.13.3 AXI Endian Transformations

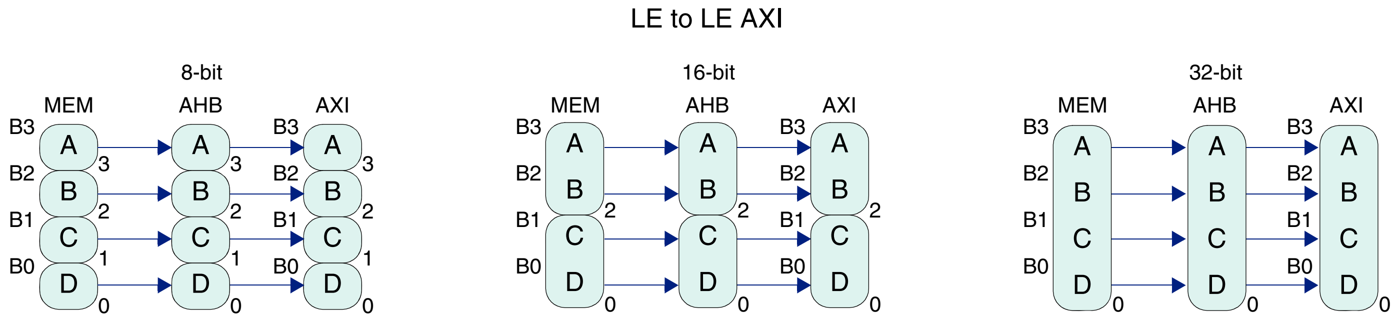
The DW_axi_a2x internally implements endian transformations by always converting the data to Little-Endian (LE) before pushing into the data buffer. Thus, if the primary port is configured for BE-8, BE-32, or BE-A, the data is transformed into a Little Endian representation before pushing into the buffer.

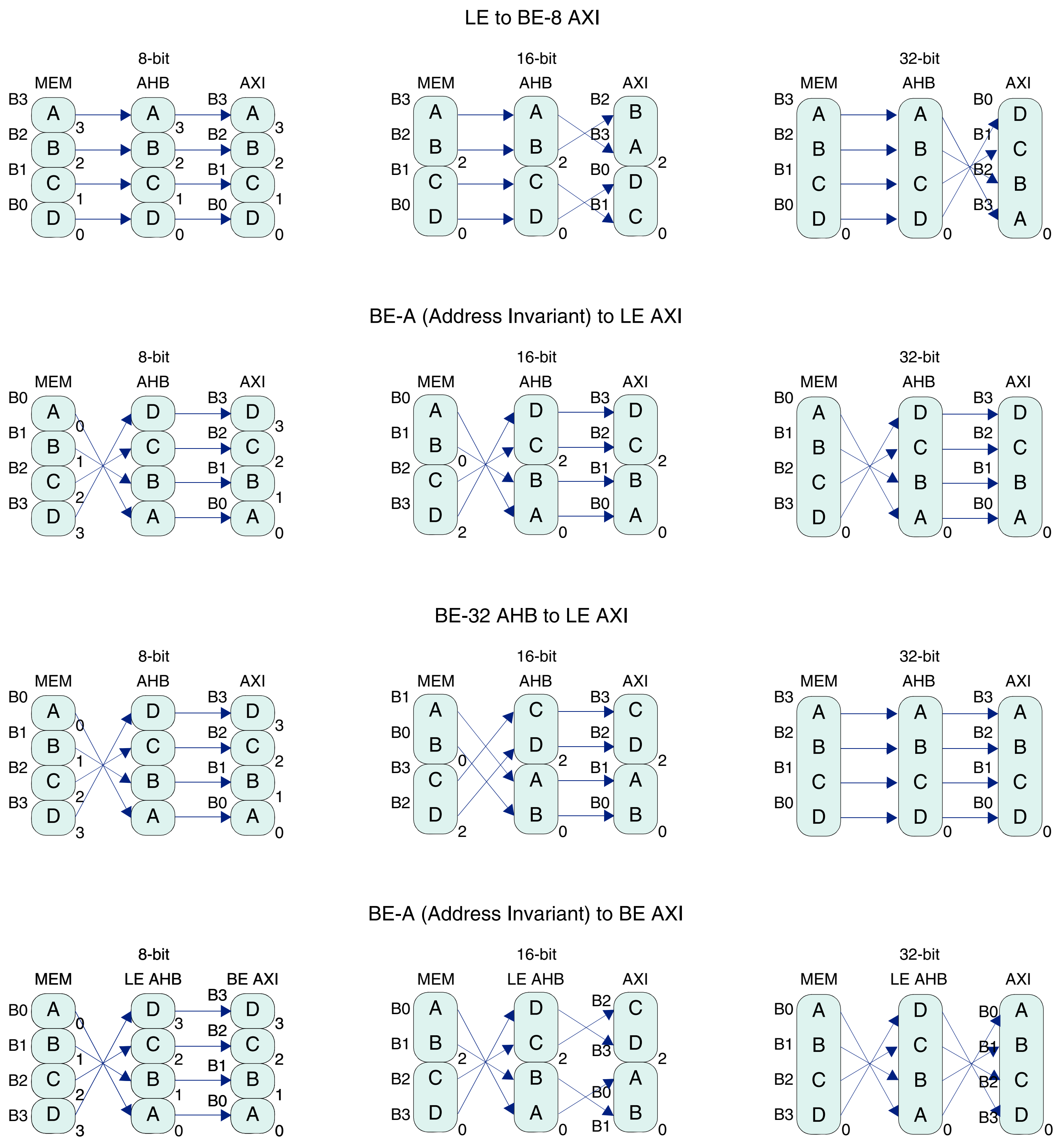
Similarly, if the secondary port is configured for BE-8, BE-32, or BE-A, the data is transformed from Little Endian to the secondary port representation.

The following diagrams show transformations for different endian representations on an equal-sized configuration; that is, primary port data widths match secondary port data widths. Figure 2-33 shows how to interpret the endian transformation diagrams.

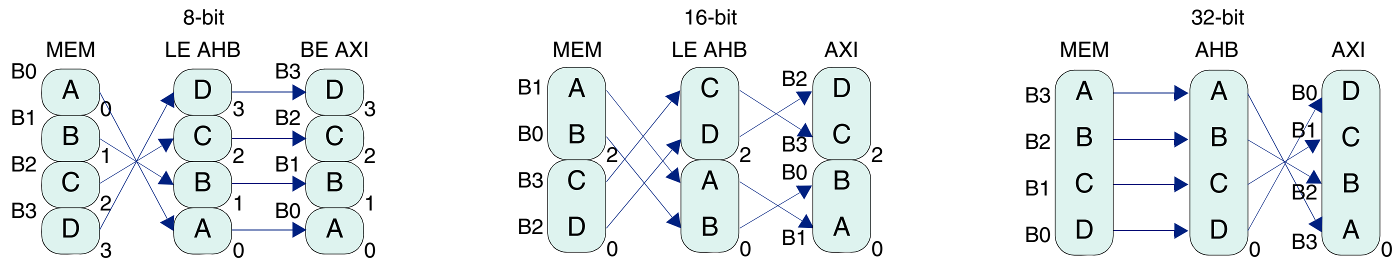
Figure 2-33 Key for Endian Transformation Diagrams



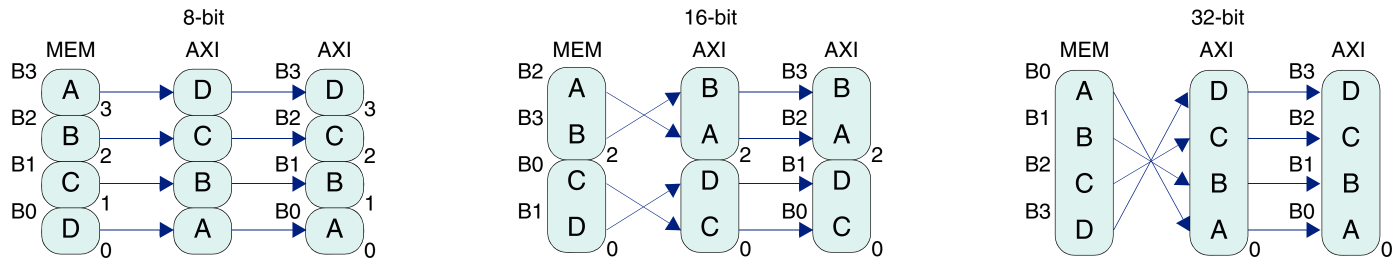



LE to BE-8 AXI
8-bit
16-bit
32-bit
MEM
AHB
AXI
BE-A (Address Invariant) to LE AXI
8-bit
16-bit
32-bit
MEM
AHB
AXI
BE-32 AHB to LE AXI
8-bit
16-bit
32-bit
MEM
AHB
AXI
BE-A (Address Invariant) to BE AXI
8-bit
16-bit
32-bit
MEM
LE AHB
BE AXI
AXI

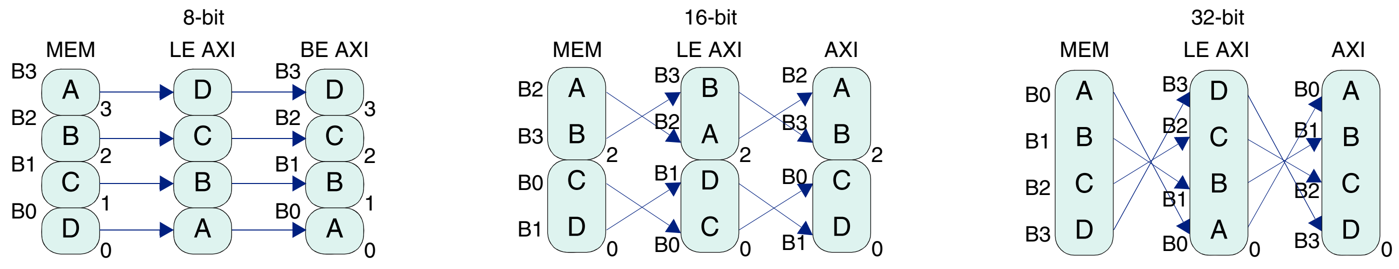
BE-32 AHB to BE-8 AXI



BE-8 AXI to LE AXI



BE-8 AXI to BE-8 AXI

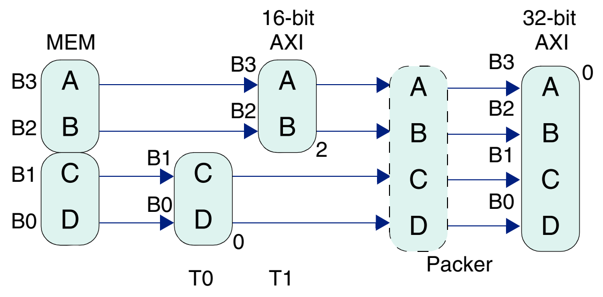


2.13.4 AXI Endianness – Upsized Transactions

The DW_axi_a2x performs primary port endian conversion using the primary port transaction size; similarly, it performs secondary port endian conversion using the secondary port transaction size.

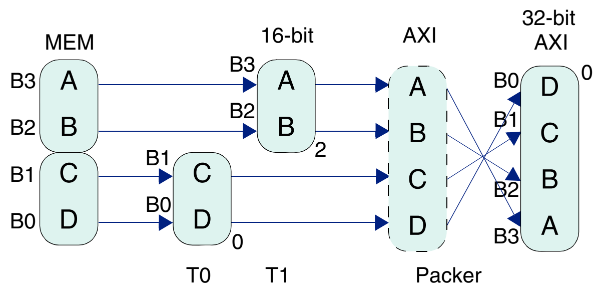
Outlined in the following diagrams are different implementations for endian transformation of an upsized configuration. The timing of the transaction for a 16-bit primary port to 32-bit secondary port is shown as T0 and T1; T0 represents the first transaction, and T1 represents the second transaction. The diagrams do not show the conversions for a primary port BE-32 or BE-A. However, the transformations can be deduced by looking at previous implementations of a BE-32/BE-A to an LE for the conversion before entering the packer and an LE-8/BE-8 after exiting the buffer.

LE AXI to LE AXI (upsizing PP_DW(16) to SP_DW(32))



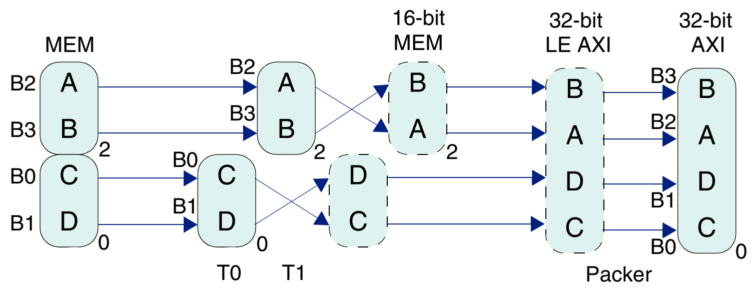
The diagram below shows a DW_axi_a2x configuration with the secondary port configured for Big-Endian AXI (BE-8). The example shows a write transaction with the data packed before the secondary port endian transformation is performed. The secondary port endian conversion is performed on the secondary port data bus using the secondary port transaction size.

LE AXI to BE-8 AXI



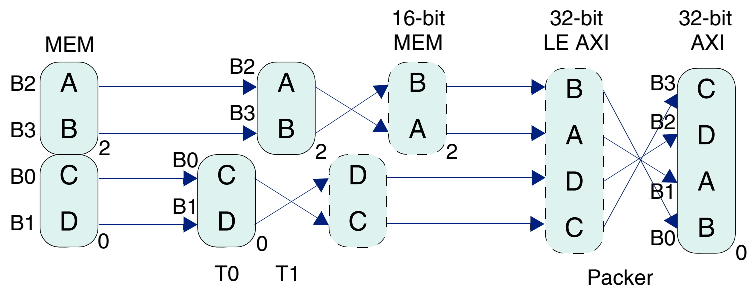
The diagram below shows a DW_axi_a2x configuration with the primary port configured for Big-Endian AXI (BE-8). The example shows a write transaction with the data packed after the primary port endian transformation is performed. The primary port endian conversion is performed on the primary port data bus using the primary port transaction size.

BE-8 AXI to LE AXI



The diagram below shows a DW_axi_a2x configuration with the secondary and primary port configured for Big-Endian AXI (BE-8). The example shows a write transaction with the data packed after the primary port endian transformation is performed. The primary port endian conversion is performed on the primary port data bus using the secondary port transaction size. The secondary port endian conversion is performed on the secondary port data bus using the secondary port transaction size.

BE-8 AXI to BE-8 AXI

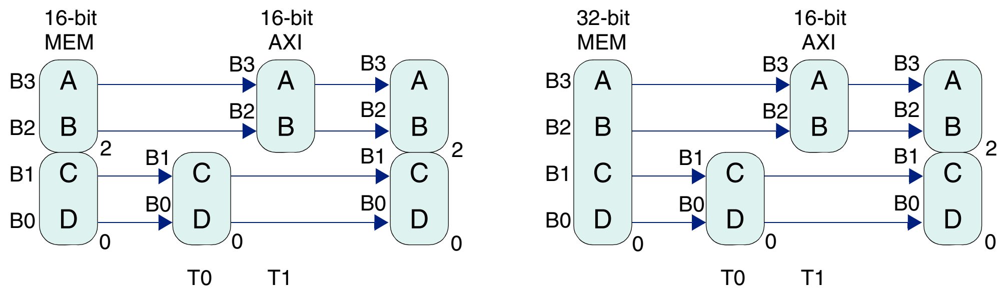


2.13.5 AXI Endianness – Downsized Transactions

The DW_axi_a2x performs primary port conversion using the primary port transaction size; similarly, it performs secondary port conversion using the secondary port transaction size.

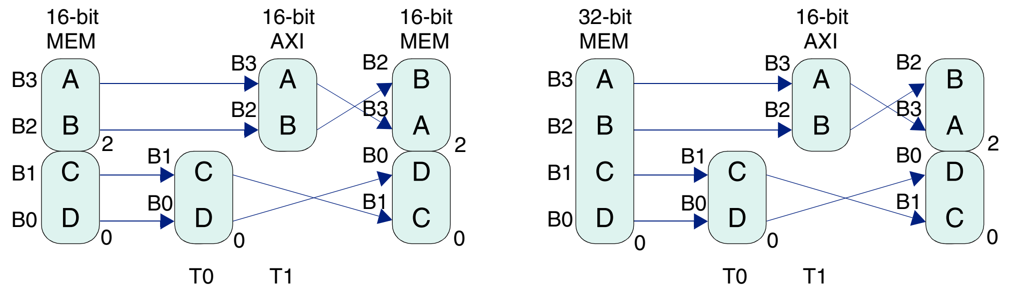
Outlined in the following diagrams are different implementations for endian transformation of a downsized configuration. The timing of the transaction for a 32-bit primary port to a 16-bit secondary port is shown as T0 and T1. T0 represents the first transaction, and T1 represents the second transaction. The diagrams do not show conversions for a primary port BE-32 or BE-A. However, the transformations can be deduced by looking at previous implementations of a BE-32/BE-A to a LE for the conversion before entering the packer, and an LE- BE-32/BE-A after exiting the buffer.

LE AXI to LE AXI (downsizing PP_DW(32) to SP_DW(16))



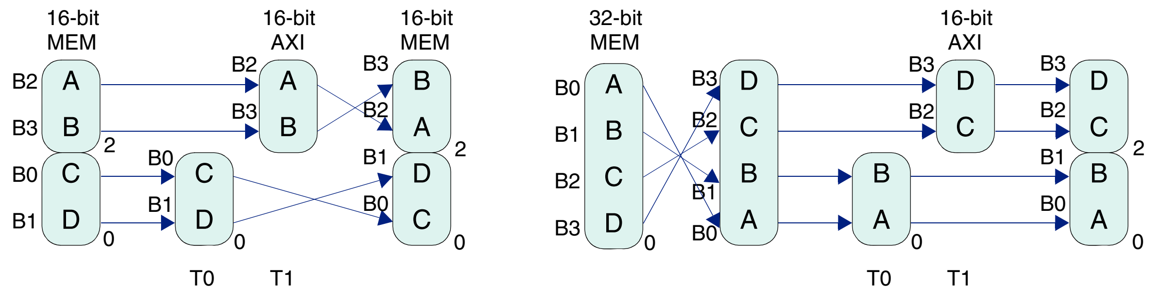
The diagram below shows a DW_axi_a2x configuration with the secondary port configured for Big-Endian AXI (BE-8). The example shows a two-write transaction; the first with a primary port transaction size of 16 bits and the second with a primary port transaction size of 32 bits. Both transactions are unpacked before the secondary port endian mapping transformation is performed using the secondary port transaction size of 16 bits.

LE AXI to BE-8 AXI



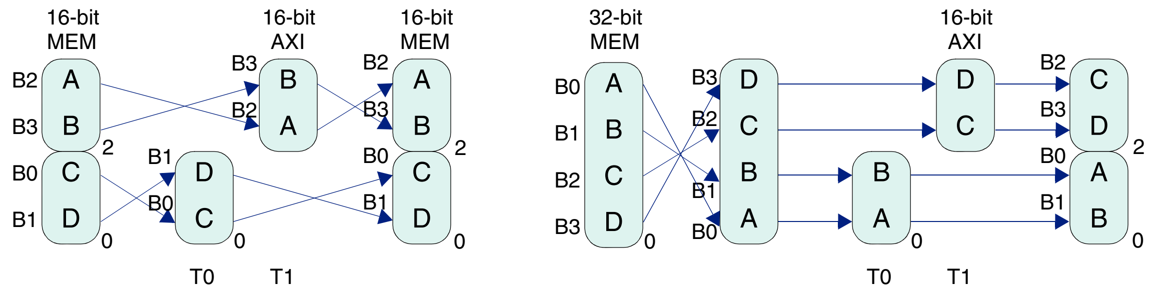
The diagram below shows a DW_axi_a2x configuration with the primary port configured for Big-Endian AXI (BE-8). The example shows a two-write transaction; the first with a primary port transaction size of 16 bits and the second with a primary port transaction size of 32 bits. Both transactions are endian-mapping using the primary port transaction size before entering the DW_axi_a2x data unpacker.

BE-8 AXI to LE AXI



The diagram below shows a DW_axi_a2x configuration with the primary and secondary port configured for Big Endian AXI (BE-8). The example again shows a two-write transaction; the first with a primary port transaction size of 16 bits and the second with a primary port transaction size of 32 bits. Both transactions are endian-mapping using the primary port transaction size before entering the DW_axi_a2x data unpacker. The secondary port endian mapping transformation is performed using the secondary port transaction size of 16 bits after the data has been unpacked.

BE-8 AXI to BE-8 AXI



2.14 Outstanding Transaction Limits

The DW_axi_a2x uses buffers that control AHB-to-AXI conversions and AHB/AXI resized conversions for upsized and downsized configurations. The sections below describe how the buffers are used in the write and read architecture and the impact these buffers have on the AXI secondary and primary port.

2.14.1 Outstanding Write Transaction Limits

Figure 2-34 shows all write buffers used by the DW_axi_a2x. Depending on the configuration, the buffers are removed or instantiated into the DW_axi_a2x architecture.

Figure 2-34 Write Transaction Limits

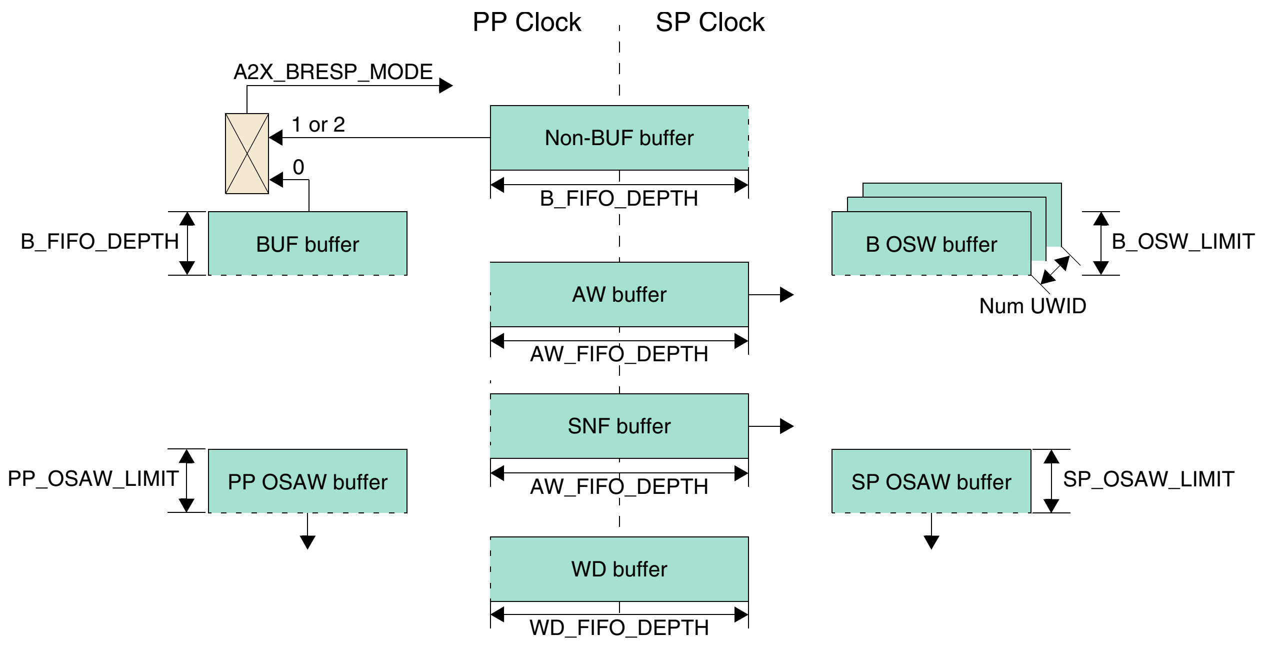


Table 2-2 lists configurations where the write transaction buffers are in use and how they impact the DW_axi_a2x buffers.

Table 2-2 Outstanding Write Transaction Limits

Buffers	Configurations	Buffer Full Condition
BUF buffer	AXI bufferable-only	■ Stalls primary port write data path ■ wready_pp driven low
NBUF buffer	■ AHB/AXI non-bufferable-only ■ AHB Dynamic	■ Stalls secondary port write response channel ■ bready_sp driven low
B OSW buffer	■ AXI resized non-bufferable ■ AHB resized non-bufferable ■ AHB dynamic	■ Stalls secondary port write address channel ■ awready_sp driven low

Table 2-2 Outstanding Write Transaction Limits

Buffers	Configurations	Buffer Full Condition
AW buffer	All configurations	■ Stalls primary port write address path ■ awready_pp driven low in AXI ■ hready driven low or master split in AHB
SNF buffer	Store-forward configurations	■ Stalls primary port write data path ■ wready_pp driven low in AXI ■ hready driven low or master split in AHB
WD buffer	All configurations	■ Stalls primary port write data path ■ wready_pp driven low in AXI ■ hready driven low or master split in AHB
PP OSAW buffer	■ All upsizing configurations ■ All downsizing configurations with store-forward ■ AXI primary port big-endian configurations	■ Stalls primary port write address path ■ awready_pp driven low in AXI ■ hready driven low or master split in AHB
SP OSAW buffer	■ All AHB configurations ■ All downsizing configurations ■ Secondary port big-endian configurations	■ Stalls secondary port write address channel ■ awready_sp driven low ■ Prevents write data been sent before address

2.14.2 Outstanding Read Transaction Limits

Figure 2-35 shows all read buffers used by the DW_axi_a2x. Depending on the configuration, the buffers are removed or instantiated into the DW_axi_a2x architecture.

Figure 2-35 Read Transaction Buffers

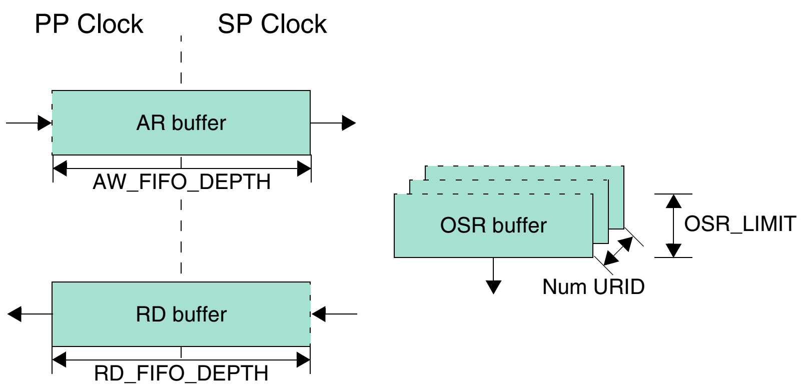


Table 2-3 lists configurations where the read transaction buffers are in use and how they impact the DW_axi_a2x buffers.

Table 2-3 Outstanding Read Transaction Limits

Buffers	Configurations	Buffer Full Condition
AR buffer	All configurations	■ Stalls primary port read address path ■ arready_pp driven low in AXI ■ hready driven low in AHB Non-Split ■ Split response returned in AHB Split mode
RD buffer	All configurations	■ Stalls secondary port read data path ■ rready_sp driven low
OSR buffer	■ All upsizing configurations ■ All downsizing configurations ■ AHB configurations with INCR beat count greater than maximum secondary port length: HINCR_RBCNT > 2^{A2X_BLW} ■ Secondary port endian configurations ■ All Locked configurations	■ Stalls secondary port read address path ■ arready_sp driven low in AXI

2.15 Clocks and Resets

The clocking modes are configured by the coreTools A2X_CLK_MODE parameter.

2.15.1 Synchronous

If the A2X_CLK_MODE is configured for synchronous mode, there is only one clock and reset present in the DW_axi_a2x. Any clock-crossing boundary logic is removed.

The DW_axi_a2x secondary port clock and reset are removed. In AXI configurations, clk_pp and resetn_pp are used and in AHB configurations, hclk and hresetn ports are used.

2.15.2 Quasi-Synchronous

If the A2X_CLK_MODE is configured for quasi-synchronous mode, there are two clocks and two reset pins in the DW_axi_a2x. The clocks are of an integer ratio to each other and are edge-aligned. Since the edges are aligned, there is no risk of metastability.

2.15.3 Asynchronous

If the A2X_CLK_MODE is configured for asynchronous mode, there is a separate clock and reset port for the DW_axi_a2x primary port and secondary port. The secondary port uses clk_sp and resetn_sp; the primary port uses clk_pp and resetn_pp for AXI configurations or hclk and hresetn for AHB configurations. These primary port and secondary port clocks can be driven asynchronously.

When A2X_CLK_MODE is configured in Quasi-synchronous or Asynchronous mode, it is assumed that both the reset inputs are asserted and de-asserted at the same time with de-assertion synchronous to the corresponding clocks. De-assertion may not be exactly at the same time as there may be delays associated with respect to the synchronization to each clock domain.

Note DW_axi_a2x does not support asserting of one reset while the other is de-asserted; doing so results in unpredictable behavior.

2.15.4 Synchronization Depth

The A2X_PP_SYNC_DEPTH and A2X_SP_SYNC_DEPTH parameters set the number of synchronizing register stages in signals that pass between the DW_axi_a2x primary and secondary ports.

- 0 – No synchronization
- 2 – Two-stage synchronization with both stages synchronizing on positive clock edge
- 3 – Three-stage synchronization, with all stages synchronizing on positive clock edge

The number of synchronization stages directly affects the latency through the DW_axi_a2x. For example, if A2X_PP_SYNC_DEPTH = 2 and the channel buffer is empty, a transaction pushed into the write address channel is not driven out from the DW_axi_a2x secondary port until one primary port clock—increment write pointer—plus three secondary port clocks later (two for synchronization plus one for status flag registering).

The DW_axi_a2x uses the A2X_PP_SYNC_DEPTH and A2X_SP_SYNC_DEPTH parameters to configure different numbers of synchronization stages for the primary port and secondary port sides.

2.16 Low-Power Interface

The DW_axi_a2x can be configured to support a low-power handshaking interface, which allows the DW_axi_a2x to inform the low-power controller (LPC) that it has no outstanding transaction. The DW_axi_a2x has a passive low-power interface and only indicates acceptance into low-power when inactive and when there are no outstanding transactions in the DW_axi_a2x.

Note The DW_axi_a2x cannot be forced into low power by the Low Power Controller.

Key features of this interface are:

- Parameters
  - A2X_LOWPWR_IF – adds low-power handshaking signals synchronous to the primary port clock:
    - csysreq
    - csysack
    - cactive
  - A2X_LOWPWR_NOPX_CNT – counts number of cycles DW_axi_a2x has to be inactive before de-asserting cactive signal
- cactive de-assertion:

- Occurs when there are no active transactions and A2X_LOWPWR_NOPX_CNT cycles have elapsed since last active transaction completed
- De-asserts after detecting falling edge of csysreq signal and DW_axi_a2x has been internally reset

- cactive assertion – occurs upon detection of:
  - Rising edge of csysreq signal
  - Rising edge of arvalid_pp, awvalid_pp, or wvalid_pp signals in AXI mode
  - Rising edge of hsel in AHB mode
- Clock gating:
  - When no active transaction exists in DW_axi_a2x, DW_axi_a2x asserts cactive port to indicate that clocks can be taken away from DW_axi_a2x

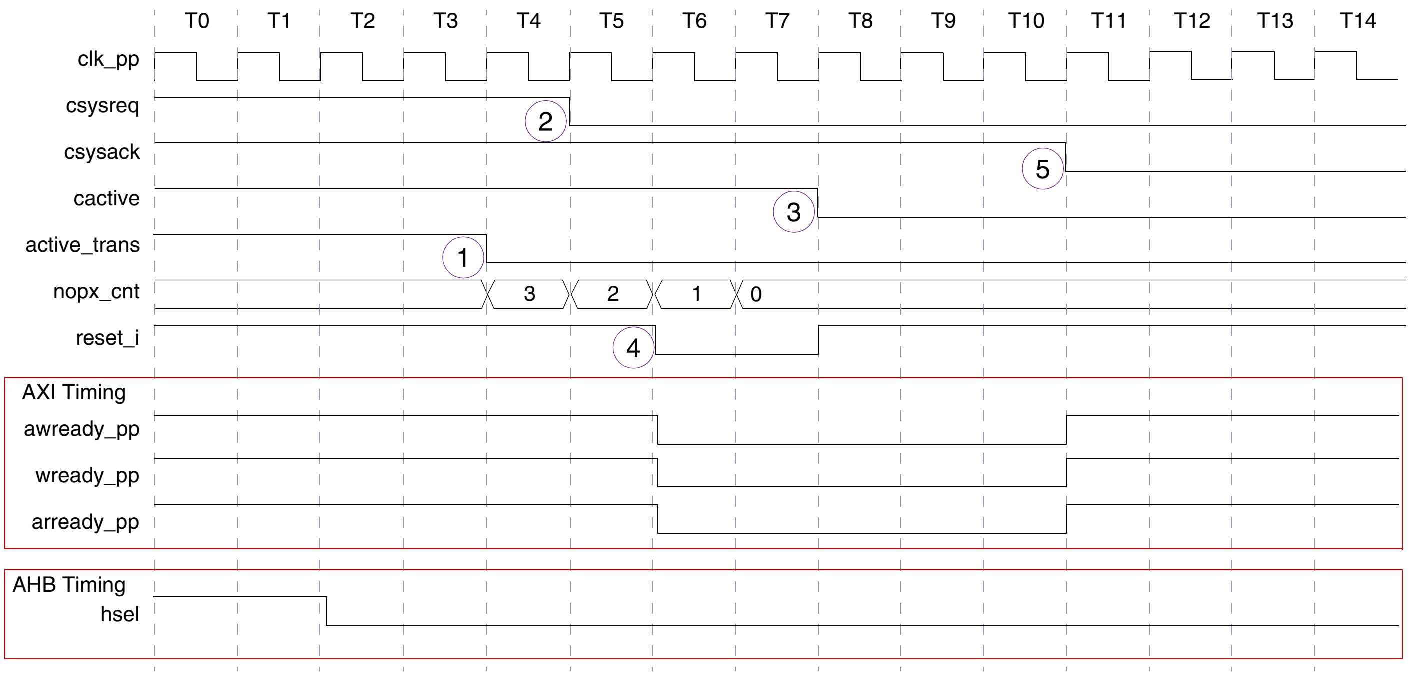
2.16.1 Low-Power Entry

The DW_axi_a2x accepts entry into low-power if the following conditions are true:

- There is no pending transactions in DW_axi_a2x
- There is no outstanding transaction on AXI secondary port
- There is no active transactions on the primary port; that is, awvalid_pp, arvalid_pp, or wvalid_pp are active-high in AXI Mode or hsel in AHB mode

Figure 2-36 illustrates this situation.

Figure 2-36 Low-Power Entry



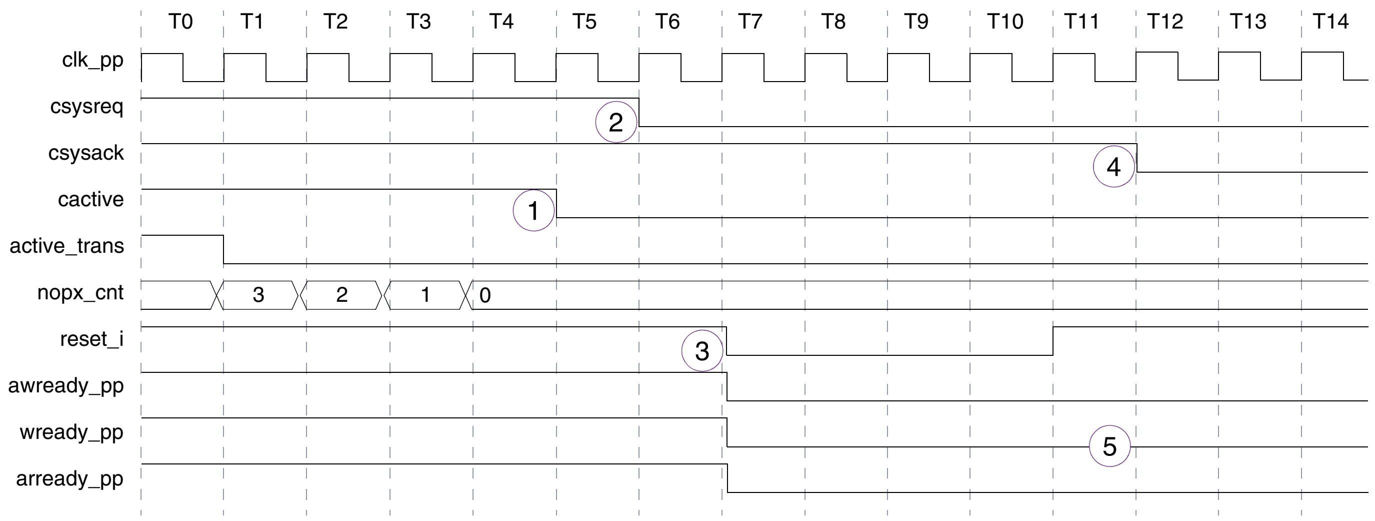
1. At T4, DW_axi_a2x becomes inactive and nopx count starts to decrement
2. At T5, low-power entry is requested and csysreq de-asserted

3. At T8, cactive is driven low to indicate that DW_axi_a2x is inactive
4. At T6, DW_axi_a2x generates internal reset:
   - AXI drives ready signals driven low on primary port in order to prevent any transaction from been granted
   - AHB, hsel remain low, allowing DW_axi_a2x to grant low-power request
5. At T11, DW_axi_a2x has completed reset and acknowledges low-power entry request by driving csysack low

2.16.2 Low-Power Entry – cactive De-Assertion

The DW_axi_a2x enters a low-power state when requested by the LPC and cactive has been de-asserted. However, in this case cactive is driven low before the low-power entry request. Figure 2-37 illustrates this situation.

Figure 2-37 Low-Power Entry – cactive De-Assertion



1. At T5, cactive driven low to indicate that DW_axi_a2x is inactive
2. At T6, DW_axi_a2x receives low-power request csysreq
3. At T7, DW_axi_a2x generates internal reset and drives primary port ready signals low to prevent further AXI transactions to DW_axi_a2x
4. At T12, DW_axi_a2x completes reset and acknowledges low-power request by driving csysack low
5. At T12, primary port ready signals remain low until low-power exit request is generated

2.16.3 Low-Power Rejection

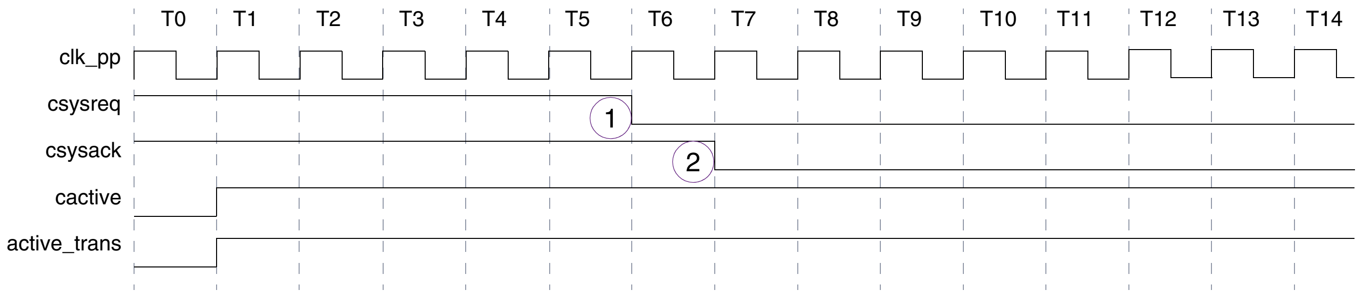
The DW_axi_a2x rejects low power requests by de-asserting csysack and holding cactive high. The DW_axi_a2x rejects entry into low-power if the following conditions are true:

- There are pending transactions in DW_axi_a2x
- There are outstanding transaction on AXI secondary port

- There are active transactions on primary port; that is, awvalid_pp, arvalid_pp, or wvalid_pp in AXI mode or hsel in AHB mode

Figure 2-38 illustrates this situation.

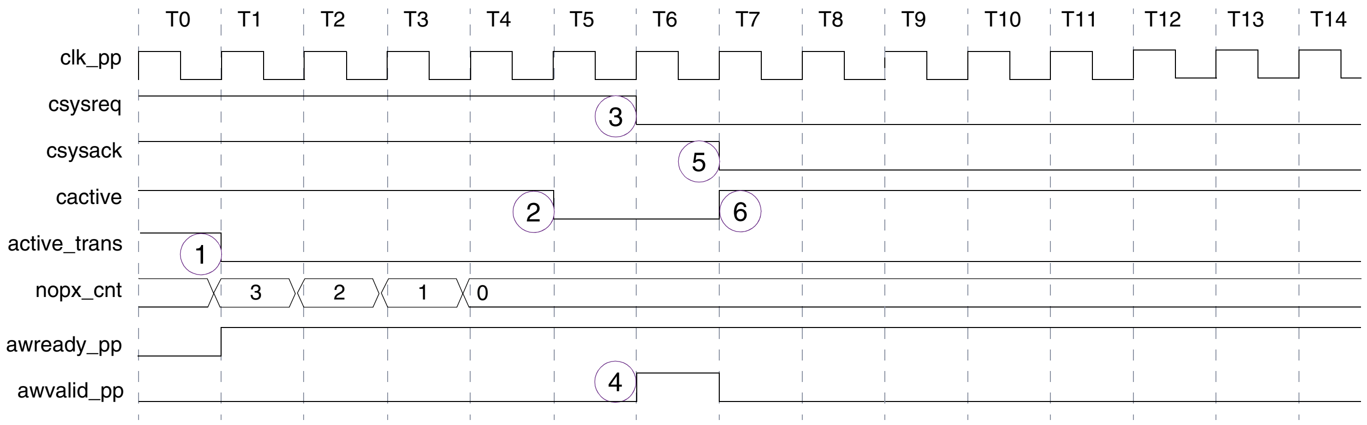
Figure 2-38 Low-Power Rejection – Scenario #1



1. At T6, DW_axi_a2x receives request to enter low-power mode
2. At T7, DW_axi_a2x responds to low-power entry request by de-asserting csysack; cactive held high to indicate DW_axi_a2x is busy

In another scenario, the DW_axi_a2x de-asserts cactive and then rejects a request by re-asserting cactive when the DW_axi_a2x becomes busy again. Figure 2-39 illustrates this situation.

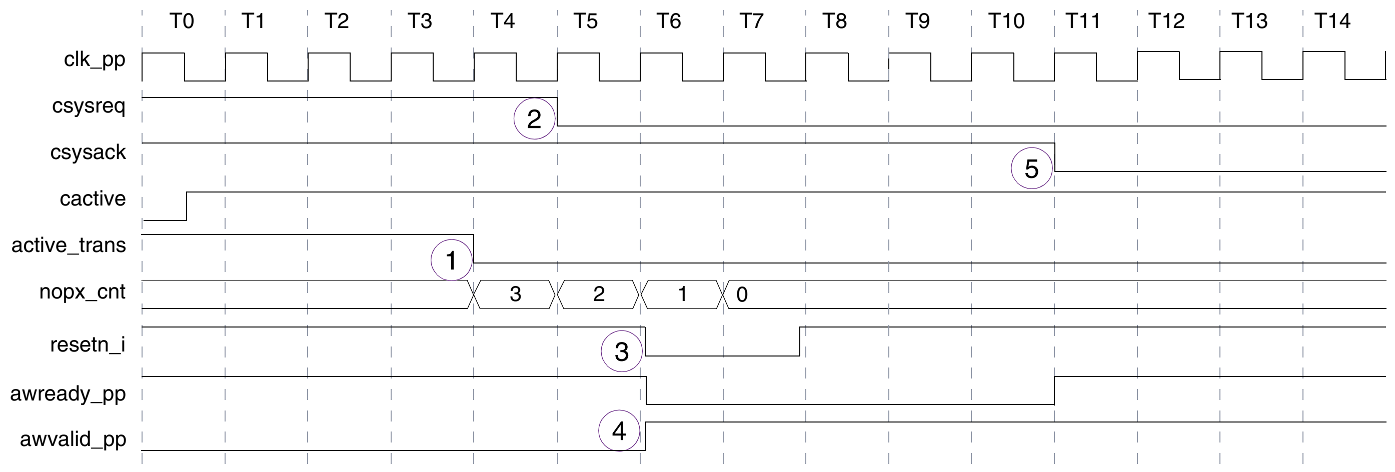
Figure 2-39 Low-Power Rejection – Scenario #2



1. At T1, DW_axi_a2x becomes inactive; nopx count starts to decrement
2. At T5, DW_axi_a2x de-asserts cactive
3. At T6, low-power entry requested; csysreq de-asserted
4. At T6, DW_axi_a2x receives write address request
5. At T7, DW_axi_a2x acknowledges low-power entry request by driving csysack low
6. At T7, DW_axi_a2x drives cactive high to indicate active state

In a different scenario, the DW_axi_a2x rejects a low-power entry request by keeps cactive high. Figure 2-40 illustrates this situation.

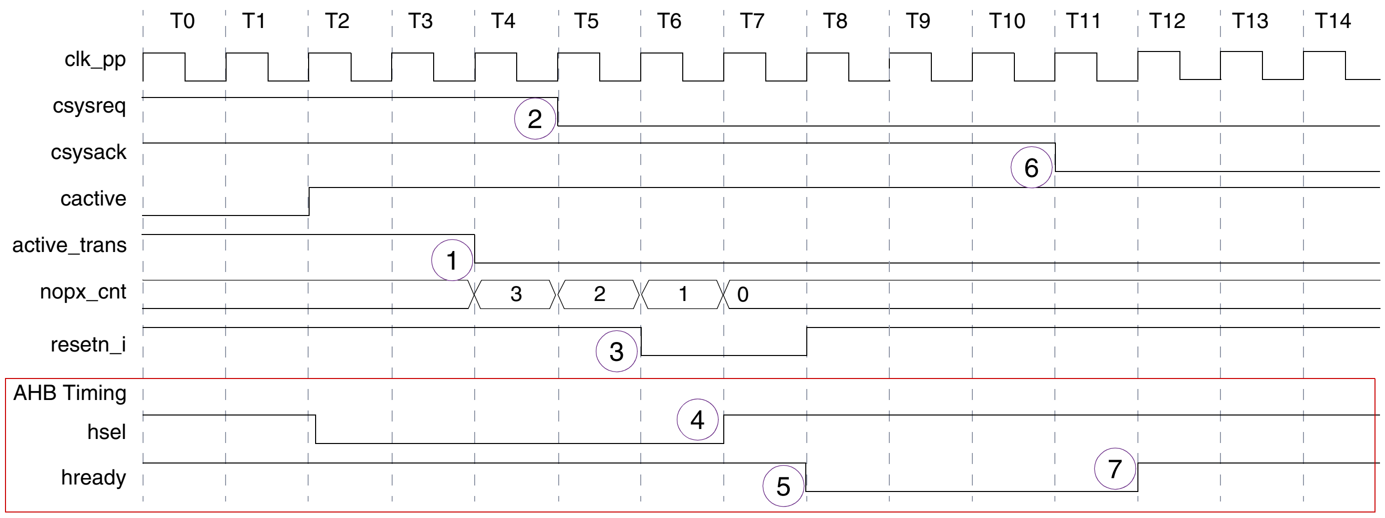
Figure 2-40 Low-Power Rejection – Scenario #3



1. At T4, DW_axi_a2x becomes inactive; nopx count starts to decrement
2. At T5, low-power entry requested; csysreq de-asserted
3. At T6, DW_axi_a2x generates internal reset and drives AXI ready signals low on primary port to prevent any transaction from been granted
4. At T6, DW_axi_a2x receives valid address from primary port Write Address channel
5. At T11, DW_axi_a2x has completed reset and acknowledges low-power entry request by driving csysack low; low-power entry request rejected by keeping cactive high

In yet another scenario, the DW_axi_a2x receives a valid transaction for the AHB primary port and drives hready low. After completing a reset and acknowledging a low-power request, he DW_axi_a2x rejects the low-power entry request by keeping cactive high and accepting the AHB transaction. Figure 2-41 illustrates this situation.

Figure 2-41 Low-Power Rejection – Scenario #4



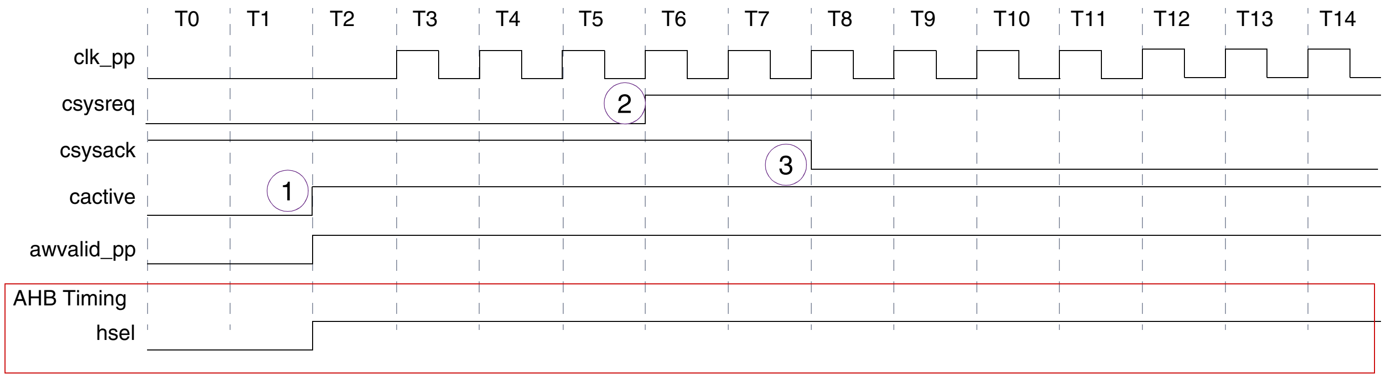
1. At T4, DW_axi_a2x becomes inactive; nopx count starts to decrement

2. At T5, low-power entry requested; csysreq de-asserted
3. At T6, DW_axi_a2x generates internal reset
4. At T7, DW_axi_a2x receives valid transaction for AHB primary port
5. At T8, DW_axi_a2x drives hready low in response to AHB transaction
6. At T11, DW_axi_a2x has completed reset and acknowledges low-power entry request by driving csysack low; low-power entry request rejected by keeping cactive high
7. At T12, AHB transaction accepted by driving hready high

2.16.4 Low-Power Exit

The DW_axi_a2x exits a low-power state when requested by the LPC and cactive has been asserted. Figure 2-42 illustrates this situation.

Figure 2-42 Low-Power Exit



1. At T2, awvalid_pp for AXI or hel for AHB is asserted; cactive combinatorially asserted to indicate DW_axi_a2x must exit low-power
2. At T6, DW_axi_a2x receives request to exit low-power
3. At T8, DW_axi_a2x acknowledges low-power exit request by driving csysack low

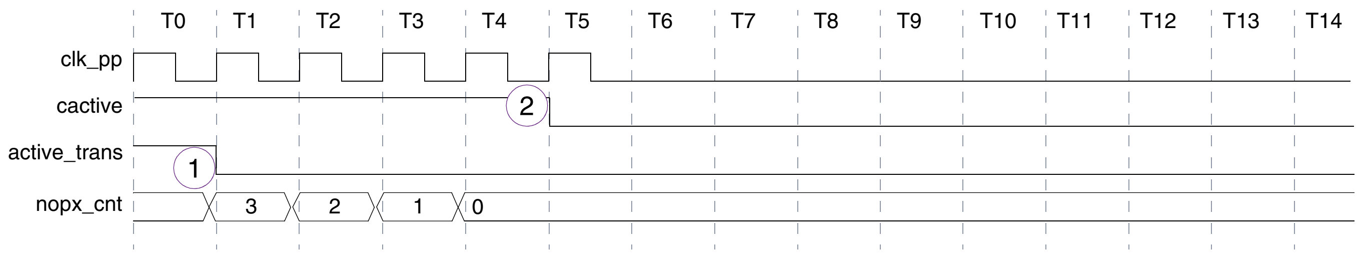
2.17 Clock Gating

The cactive output is driven from the internal active_trans signal and the hsel input signal in AHB Mode or the awvalid, wvalid, and arvalid signals in AXI mode.

2.17.1 Gating DW_axi_a2x Clocks

In Figure 2-43, the DW_axi_a2x de-asserts cactive when the clock is not longer required.

Figure 2-43 Gating the DW_axi_a2x Clocks

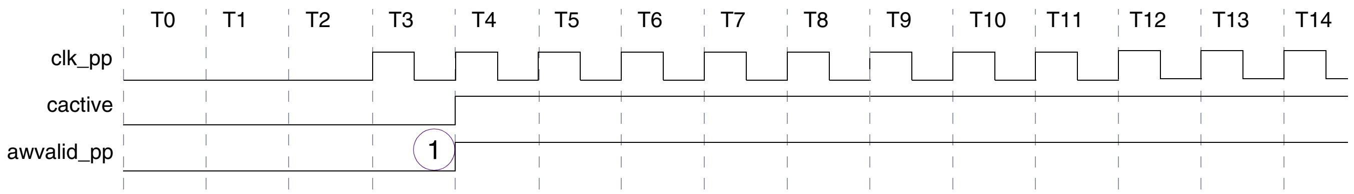


1. At T1, DW_axi_a2x becomes inactive and nopx count starts to decrement
2. At T5, DW_axi_a2x de-asserts cactive indicating that clock is no longer required

2.17.2 Enabling DW_axi_a2x Clocks

In Figure 2-44, the DW_axi_a2x asserts cactive when the clock is required.

Figure 2-44 Enabling DW_axi_a2x Clocks



1. At T4, DW_axi_a2x receives request from AW channel and asserts clock gate to indicate need for clock

The clock gating output port cactive is combinatorially driven by:

- AXI – awvalid, wvalid, and arvalid primary port inputs
- AHB – hsel input port

If any of these input ports are asserted, the DW_axi_a2x asserts the cactive output port to indicate that a clock is required.

3 Parameter Descriptions

This chapter details all the configuration parameters. **You can use the coreConsultant GUI configuration reports to determine the actual configured state of the controller.** Some expressions might refer to TCL functions or procedures (sometimes identified as **<functionof>)** that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the controller in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

The parameter descriptions in this chapter include the **Enabled:** attribute which indicates the values required to be set on other parameters before you can change the value of this parameter.

These tables define all of the ~~user~~ configuration options for this component.

3.1 A2X Configuration Parameters

Table 3-1 A2X Configuration Parameters

Label	Description
	A2X Configuration
Select AXI Interface Type?	Select AXI Interface Type as AXI3, AXI4, or ACE-Lite. By default, DW_axi_a2x supports the AXI3 interface. ■ When A2X_PP_MODE = 0: AXI3 (0) and AXI4 (1) are supported. ■ When A2X_PP_MODE = 1: AXI3 (0), AXI4 (1) and ACELITE (2) are supported. **Values:** ■ AXI3 (0) ■ AXI4 (1) ■ ACELITE (2) **Default Value:** AXI3 **Enabled:** Always **Parameter Name:** A2X_AXI_INTERFACE_TYPE
Primary Port Mode	Denotes the mode in which the DW_axi_a2x operates (either connecting to another AXI bus fabric or to an AHB bus fabric). ■ AHB interface - Connects to AHB bus fabric ■ AXI interface - Connects to AXI bus fabric **Values:** ■ AHB (0) ■ AXI (1) **Default Value:** AXI **Enabled:** Always **Parameter Name:** A2X_PP_MODE
Write Buffer Mode	Sets the DW_axi_a2x write buffer mode. **Values:** ■ Cut-Through (0) ■ Store-Forward (1) **Default Value:** Cut-Through **Enabled:** Always **Parameter Name:** A2X_WBUF_MODE

Table 3-1 A2X Configuration Parameters (Continued)

Label	Description
Read Buffer Mode	Sets the DW_axi_a2x read buffer mode. In AHB mode with Locked support enabled, the Read Buffer Mode (A2X_RBUF_MODE) is hard-coded to Cut-Through. In this mode, the read data buffer is large enough to accept any outstanding read data transactions. Thus, Store-Forward is not required in this configuration. For all other modes, the A2X_RBUF_MODE parameter is user-configurable. **Values:** ■ Cut-Through (0) ■ Store-Forward (1) **Default Value:** Cut-Through **Enabled:** ((A2X_PP_MODE==1) || ((A2X_PP_MODE==0) && (A2X_LOCKED==0)))? 1 : 0 **Parameter Name:** A2X_RBUF_MODE
Primary Port Endianess	Specifies the endian type of the DW_axi_a2x Primary Port. **Values:** ■ (LE) Little Endian (0) ■ (BE-32) Big Endian (1) ■ (BE-A) Big Endian (2) ■ (BE-8) Big Endian (3) **Default Value:** (LE) Little Endian **Enabled:** Always **Parameter Name:** A2X_PP_ENDIAN
Secondary Port Endianess	Specifies the endian type of the DW_axi_a2x Secondary Port. **Values:** ■ (LE) Little Endian (0) ■ (BE-8) Big Endian (3) **Default Value:** (LE) Little Endian **Enabled:** Always **Parameter Name:** A2X_SP_ENDIAN
Include QoS Signals?	If true, the primary port master and secondary port slave write and read address channels have QoS signals (awqos_*/arqos_*) in the I/O. **Values:** ■ No (0) ■ Yes (1) **Default Value:** No **Enabled:** (((A2X_AXI_INTERFACE_TYPE==1) || (A2X_AXI_INTERFACE_TYPE==2)) && (A2X_PP_MODE==1)) **Parameter Name:** A2X_INC_QOS

Table 3-1 A2X Configuration Parameters (Continued)

Label	Description
Include REGION Signals?	If true, the primary port master and secondary port slave write and read address channels have Region signals (awregion_*/arregion_*) in the I/O. **Values:** ■ No (0) ■ Yes (1) **Default Value:** No **Enabled:** (((A2X_AXI_INTERFACE_TYPE==1) || (A2X_AXI_INTERFACE_TYPE==2)) && (A2X_PP_MODE==1)) **Parameter Name:** A2X_INC_REGION

3.2 AHB Configuration Parameters

Table 3-2 AHB Configuration Parameters

Label	Description
	AHB Configuration
AHB-Lite Mode	Configures the DW_axi_a2x for AHB-Lite mode. In this mode: ■ AHB Split responses are not supported. ■ DW_axi_a2x supports only one AHB Master in AHB-Lite mode. ■ DW_axi_a2x responds to AHB reads and non-bufferable writes by driving hready low. **Values:** ■ Disable (0) ■ Enable (1) **Default Value:** Disable **Enabled:** A2X_PP_MODE==0 **Parameter Name:** A2X_AHB_LITE_MODE
AHB Split Mode	Configures the DW_axi_a2x for Split mode. ■ 0: DW_axi_a2x responds to AHB reads and non-bufferable writes by driving hready low. ■ 1: DW_axi_a2x responds to AHB reads and non-bufferable writes with Split response. **Values:** ■ Disable (0) ■ Enable (1) **Default Value:** (A2X_AHB_LITE_MODE==0)? 1 : 0 **Enabled:** ((A2X_PP_MODE==0) && (A2X_AHB_LITE_MODE==0)) **Parameter Name:** A2X_AHB_SPLIT_MODE
Number of AHB Masters	Defines the number of active AHB Masters that the DW_axi_a2x can support. The number of AHB masters specified by this parameter does not include the dummy master; the DW_axi_a2x assumes that the dummy master is HMASTER zero. **Values:** 1, ..., 15 **Default Value:** (A2X_AHB_LITE_MODE==1)? 1 : 8 **Enabled:** ((A2X_PP_MODE==0) && (A2X_AHB_LITE_MODE==0)) **Parameter Name:** A2X_NUM_AHBM

Table 3-2 AHB Configuration Parameters (Continued)

Label	Description
AHB Write Hready Low Period	Defines the number of clock cycles for which the DW_axi_a2x drives hready low before issuing a split response to a write transaction. The DW_axi_a2x drives hready low when it cannot accept a write transaction due to a Buffer Full condition. **Values:** 10, ..., 200 **Default Value:** 100 **Enabled:** ((A2X_PP_MODE==0) && (A2X_AHB_SPLIT_MODE==1)) **Parameter Name:** A2X_HREADY_LOW_PERIOD
Hardcode AHB INCR Ports	Enables/disables the DW_axi_a2x Read and Write Beat Counter Ports, namely hincr_rbcnt_m and hincr_wbcnt_m. ■ 0: Generates hincr_rbcnt and hincr_wbcnt ports ■ 1: Removes all hincr_rbcnt and hincr_wbcnt ports The number of ports are based on the A2X_SINGLE_RBCNT and A2X_SINGLE_WBCNT parameters. **Values:** ■ No (0) ■ Yes (1) **Default Value:** Yes **Enabled:** A2X_PP_MODE==0 **Parameter Name:** A2X_HINCR_HCBCNT
Single AHB Write INCR Ports	Determines the number of AHB Write INCR Ports (hincr_wbcnt_m). ■ 0: Generates an AHB Write INCR Port for each AHB Master. The DW_axi_a2x uses the hincr_wbcnt corresponding to the AHB Master for an AHB INCR transaction. ■ 1: Generates only one AHB Write INCR Port (hincr_wbcnt_m1). The DW_axi_a2x uses this port for all AHB INCR transactions. In Non-Bufferable mode, all AHB INCR transactions are treated as AHB Singles. This parameter is in use only when the DW_axi_a2x is configured for AHB Mode and the HINCR ports are not hardcoded, and the DW_axi_a2x is not an AHB Lite Mode or Non-Bufferable Mode configuration; that is, either: ■ A2X_PP_MODE=0 and A2X_HINCR_HCBCNT=0 and A2X_AHB_LITE_MODE=0 ■ A2X_PP_MODE=0 and A2X_HINCR_HCBCNT=0 and A2X_BRESP_MODE=1 **Values:** ■ No (0) ■ Yes (1) **Default Value:** Yes **Enabled:** ((A2X_HINCR_HCBCNT==1) || (A2X_AHB_LITE_MODE==1) || (A2X_BRESP_MODE==1)) ? 0 : 1 **Parameter Name:** A2X_SINGLE_WBCNT

Table 3-2 AHB Configuration Parameters (Continued)

Label	Description
Single AHB Read INCR Ports	Determines the number of AHB Read INCR Ports (hincr_rbcnt_m). ■ 0: Generates an AHB Read INCR Port for each AHB Master. The DW_axi_a2x uses the hincr_rbcnt corresponding to the AHB Master for an AHB INCR transaction. ■ 1: Generates only one AHB Read INCR Port (hincr_rbcnt_m1). The DW_axi_a2x uses this port for all AHB INCR transactions. This parameter is in use only when the DW_axi_a2x is configured for AHB Mode, and the HINCR ports are not hardcoded, and the DW_axi_a2x is not an AHB Lite Mode configuration, that is: A2X_PP_MODE=0, A2X_HINCR_HCBCNT=0 and A2X_AHB_LITE_MODE=0. **Values:** ■ No (0) ■ Yes (1) **Default Value:** Yes **Enabled:** ((A2X_HINCR_HCBCNT==1) || (A2X_AHB_LITE_MODE==1)) ? 0 : 1 **Parameter Name:** A2X_SINGLE_RBCNT
Write AHB INCR Max Length	Sets the maximum number of primary port beats for an AHB write INCR transaction. This is a Log2-encoded value that determines the number of AHB INCR data beats to accept before generating a new AXI address. ■ 0 - 1 AHB data transfer ■ 1 - 2 AHB data transfers ■ 2 - 4 AHB data transfers ■ 3 - 8 AHB data transfers ■ 4 - 16 AHB data transfers ■ 5 - 32 AHB data transfers ■ 6 - 64 AHB data transfers ■ 7 - 128 AHB data transfers ■ 8 - 256 AHB data transfers ■ 9 - 512 AHB data transfers ■ 10 - 1024 AHB data transfers This value cannot exceed a maximum of 1K bytes, that is: (AHB_data_transfers * number_of_Primary_Port_data_bytes) A minimum restriction on this parameter applies for upsizing configurations (A2X_PP_DW<A2X_SP_DW). The minimum allowed value is determined by the resize ratio, that is: A2X_PP_DW/A2X_SP_DW. **Values:** A2X_HINCR_WBCNT_MIN, ..., 10 **Default Value:** A2X_HINCR_WBCNT_MIN **Enabled:** A2X_PP_MODE==0 **Parameter Name:** A2X_HINCR_WBCNT_MAX

Table 3-2 AHB Configuration Parameters (Continued)

Label	Description
AHB Read INCR Max Prefetch	Sets the maximum number of primary port beats for an AHB read INCR transaction. This is a Log2-encoded value that determines the number of AHB Read INCR data beats to prefetch. ■ 0 - Prefetch 1 AHB data transfer ■ 1 - Prefetch 2 AHB data transfers ■ 2 - Prefetch 4 AHB data transfers ■ 3 - Prefetch 8 AHB data transfers ■ 4 - Prefetch 16 AHB data transfers ■ 5 - Prefetch 32 AHB data transfers ■ 6 - Prefetch 64 AHB data transfers ■ 7 - Prefetch 128 AHB data transfers ■ 8 - Prefetch 256 AHB data transfers ■ 9 - Prefetch 512 AHB data transfers ■ 10 - Prefetch 1024 AHB data transfers This value cannot exceed a maximum of 1K bytes. To calculate the maximum number of bytes: (AHB_data_transfers * number_of_primary_port_data_bytes) In AHB Locked mode, this value cannot exceed the DW_axi_a2x maximum prefetch of 128. In this mode, the Read Data buffer depth is set to: (number_of_masters * maximum_Prefetch) For upsizing configurations (A2X_PP_DW<A2X_SP_DW), a minimum restriction on this parameter applies. The minimum allowed value is determined by the resize ratio, that is: A2X_PP_DW/A2X_SP_DW. **Values:** A2X_HINCR_RBCNT_MIN, ..., (A2X_LOCKED==0)? 10 : 7 **Default Value:** A2X_HINCR_RBCNT_MIN **Enabled:** A2X_PP_MODE==0 **Parameter Name:** A2X_HINCR_RBCNT_MAX

3.3 Primary Port Widths Parameters

Table 3-3 Primary Port Widths Parameters

Label	Description
	Primary Port Widths
Primary Port ID Width	Specifies the ID width on the DW_axi_a2x Primary Port. **Values:** 1, ..., ~~12~~16 **Default Value:** 4 **Enabled:** (A2X_PP_MODE==0) ? 0 : 1 **Parameter Name:** A2X_PP_IDW
Primary Port Address Width	Specifies the Address Bus Width of the Primary Port. **Values:** 32, ..., 64 **Default Value:** 32 **Enabled:** Always **Parameter Name:** A2X_PP_AW
Primary Port Burst Width	Specifies the width of the Primary awlen/arlen port. ■ Minimum supported AXI length is 16 (2^4) ■ Maximum supported AXI length is 256 (2^8) With Locked mode on, a BLW of 4 is supported. **Values:** 4, 5, 6, 7, 8 **Default Value:** 4 **Enabled:** (A2X_PP_MODE==0) ? 0 : 1 **Parameter Name:** A2X_PP_BLW
Primary Port Data Width	Specifies the Primary Port Data Bus Width. **Values:** 8, 16, 32, 64, 128, 256, 512, 1024 **Default Value:** 32 **Enabled:** Always **Parameter Name:** A2X_PP_DW

3.4 Secondary Port Widths Parameters

Table 3-4 Secondary Port Widths Parameters

Label	Description
	Secondary Port Widths
Secondary Port ID Width	Specifies the ID width on the DW_axi_a2x Secondary Port. **Values:** 1, ..., ~~12~~16 **Default Value:** A2X_PP_IDW **Enabled:** Always **Parameter Name:** A2X_SP_IDW
Secondary Port Address Width	Specifies the Address Bus Width of the Secondary Port. **Values:** 32, ..., 64 **Default Value:** A2X_PP_AW **Enabled:** Always **Parameter Name:** A2X_SP_AW
Secondary Port Burst Width	Specifies the width of the Secondary awlen/arlen port. ■ Minimum supported AXI length is 16 (2^4) ■ Maximum supported AXI length is 256 (2^8) With Locked mode on, a BLW of 4 is supported. **Values:** 4, 5, 6, 7, 8 **Default Value:** 4 **Enabled:** Always **Parameter Name:** A2X_SP_BLW
Secondary Port Data Width	Specifies the Secondary Port Data Bus Width. **Values:** 8, 16, 32, 64, 128, 256, 512, 1024 **Default Value:** 32 **Enabled:** Always **Parameter Name:** A2X_SP_DW

3.5 Sideband/User Signal Width Parameters

Table 3-5 Sideband/User Signal Width Parameters

Label	Description
	Sideband/User Signal Width
Write Address Sideband/User Signal Width	Specifies the DW_axi_a2x Write Address Sideband (AXI3)/User Signal (AXI4/ACE-Lite) Bus Width. When set to 0, the write address sideband (AXI3)/user (AXI4/ACE-Lite) ports are removed. **Values:** 0, ..., 256 **Default Value:** (A2X_PP_MODE==0) ? A2X_HASBW : 0 **Enabled:** A2X_PP_MODE==1 **Parameter Name:** A2X_AWSBW
Read Address Sideband/User Width	Specifies the DW_axi_a2x Read Address Sideband (AXI3)/User Signal (AXI4/ACE-Lite) Bus Width. When set to 0, the Read address sideband (AXI3)/user (AXI4/ACE-Lite) ports are removed. **Values:** 0, ..., 256 **Default Value:** (A2X_PP_MODE==0) ? A2X_HASBW : 0 **Enabled:** A2X_PP_MODE==1 **Parameter Name:** A2X_ARSBW
Write Data Sideband/User Width	Specifies the DW_axi_a2x Write Data Sideband (AXI3)/User Signal (AXI4/ACE-Lite) Bus Width. When set to 0, the Write Data sideband (AXI3)/user (AXI4/ACE-Lite) ports are removed. **Values:** 0, ..., 256 **Default Value:** 0 **Enabled:** A2X_PP_MODE==1 **Parameter Name:** A2X_WSBW
Read Data Sideband/User Width	Specifies the DW_axi_a2x Read Data Sideband (AXI3)/User Signal (AXI4/ACE-Lite) Bus Width. When set to 0, the Read Data sideband (AXI3)/user (AXI4/ACE-Lite) ports are removed. **Values:** 0, ..., 256 **Default Value:** 0 **Enabled:** A2X_PP_MODE==1 **Parameter Name:** A2X_RSBW
Write Response Sideband/User Bus Width	Specifies the DW_axi_a2x Write Response Sideband (AXI3)/User Signal (AXI4/ACE-Lite) Bus Width. When set to 0, the Write Response sideband (AXI3)/user (AXI4/ACE-Lite) ports are removed. **Values:** 0, ..., 256 **Default Value:** 0 **Enabled:** (A2X_PP_MODE==1) && (A2X_BRESP_MODE!=0) **Parameter Name:** A2X_BSBW

3.6 Clocking Parameters

Table 3-6 Clocking Parameters

Label	Description
Clock Mode	Selects the relationship between the Primary Port clock (AXI or AHB) and the Secondary Port clock (AXI). **Values:** ■ Synchronous (0) ■ Quassi-Synchronous (1) ■ Asynchronous (2) **Default Value:** Synchronous **Enabled:** Always **Parameter Name:** A2X_CLK_MODE
Primary Port Synchronization Depth	Defines the number of synchronization register stages in the internal channel buffers for signals passing from the DW_axi_a2x Primary Port to the DW_axi_a2x Secondary Port. ■ 0 - No synchronization stages ■ ~~1 - Two-stage synchronization:~~ - ~~First stage - Negative edge~~ - ~~Second stage - Positive edge~~ ■ 2 - Two-stage synchronization, both stages positive edge ■ 3 - Three-stage synchronization, all stages positive edge If one port has a synchronization depth of 0, the other port must also be 0. **Values:** 0, ~~1,~~ 2, 3 **Default Value:** (A2X_CLK_MODE==2) ? 2 : 0 **Enabled:** A2X_CLK_MODE==2 **Parameter Name:** A2X_PP_SYNC_DEPTH
Secondary Port Synchronization Depth	Defines the number of synchronization register stages in the internal channel buffers for signals passing from the DW_axi_a2x Secondary Port to the DW_axi_a2x Primary Port. ■ 0 - No synchronization stages ■ ~~1 - Two-stage synchronization:~~ - ~~First stage - Negative edge~~ - ~~Second stage - Positive edge~~ ■ 2 - Two-stage synchronization, both stages positive edge ■ 3 - Three-stage synchronization, all stages positive edge If one port has a synchronization depth of 0, the other port must also be 0. **Values:** 0, ~~1,~~ 2, 3 **Default Value:** (A2X_CLK_MODE==2) ? 2 : 0 **Enabled:** A2X_CLK_MODE==2 **Parameter Name:** A2X_SP_SYNC_DEPTH

3.7 Locking Parameters

Table 3-7 Locking Parameters

Label	Description
DW_axi_a2x Locked	Supports the following types of locked transactions: ■ AHB to AXI ■ AXI to AXI **Values:** ■ Disabled (0) ■ Enabled (1) **Default Value:** Disabled **Enabled:** A2X_AXI_INTERFACE_TYPE==0 **Parameter Name:** A2X_LOCKED
Locked Read Data Buffer Depth	Sets the depth of the Locked Read Data Channel buffer in AHB mode. The buffer does not exist in AXI mode, or in AHB mode without Locked support. **Values:** 2, ..., A2X_LKMODE_MAX_PREFETCH **Default Value:** 4 **Enabled:** 0 **Parameter Name:** A2X_LK_RD_FIFO_DEPTH

3.8 A2X Write Configuration Parameters

Table 3-8 A2X Write Configuration Parameters

Label	Description
A2X Write Configuration	
Write Buffer Response Mode	Selects the Write Response mode. ■ Bufferable Mode (available in AXI3 and AHB Mode) - Returns OKAY responses after last write data beat is received from primary port write data channel. ■ Non-Bufferable - Returns response received on Secondary Port ■ Dynamic Mode (available only in AHB mode): - Returns response for non-bufferable transaction - Responses for bufferable transactions ignored and not returned **Note:** To support AXI exclusive access, it is recommended to set the write response mode to non-bufferable. In AXI4 configurations, only Non-Bufferable mode is supported. ■ When A2X_PP_MODE = 0: Bufferable, Non-Bufferable and Dynamic modes are supported. ■ When A2X_PP_MODE = 1: - If A2X_AXI_INTERFACE_TYPE = AXI3: If A2X_LOCKED = 1, only Non-Bufferable mode is supported. If A2X_LOCKED = 0, Non-Bufferable and Bufferable mode are supported. - If A2X_AXI_INTERFACE_TYPE = AXI4/ACELITE: Only Non-Bufferable mode is supported. **Values:** ■ Bufferable Only (0) ■ Non-Bufferable Only (1) ■ Dynamic (2) **Default Value:** Non-Bufferable Only **Enabled:** (A2X_PP_MODE==0)? 1 : (A2X_LOCKED==0)? 1 : 0 **Parameter Name:** A2X_BRESP_MODE

Table 3-8 A2X Write Configuration Parameters (Continued)

Label	Description
Write Buffer Response Order	Selects the order in which write responses are returned to the DW_axi_a2x Secondary Port. ■ In-Order - Write responses returned in the same order as sent on Secondary Port Write Address channel. ■ Out-Of-Order - Write response may return from Secondary Port in different order than sent on Secondary Port Write Address channel. This parameter is in use only for Non-Bufferable or Dynamic Response mode (A2X_BRESP_MODE=1/2). **Values:** ■ In-Order (0) ■ Out-of-Order (1) **Default Value:** In-Order **Enabled:** (A2X_BRESP_MODE==0)? 0 : 1 **Parameter Name:** A2X_BRESP_ORDER
Number of UWID	Selects the number of unique Write IDs for which the DW_axi_a2x may have outstanding on the AXI secondary port. When responses are returned in order, there is no restriction on the number of unique write IDs. In this case, this parameter is not used, and the only restriction placed on the number of outstanding write transactions is the A2X_OSAW_LIMIT parameter. This parameter is enabled for Out-of-Order Write Responses for: ■ AHB and AXI upsized and downsized configurations ■ AHB equal-sized Dynamic mode This parameter is disabled for: ■ AHB and AXI Bufferable Response mode ■ AXI equal-sized configurations For more information on usage for this parameter, refer to "Outstanding Transaction Limits" in the DesignWare DW_axi_a2x Databook. **Values:** 1, ..., (A2X_PP_MODE==0)? A2X_NUM_AHBM : 16 **Default Value:** 1 **Enabled:** ((A2X_PP_MODE==0) && (A2X_AHB_LITE_MODE==1))? 0 : ((A2X_OSW_EN==1) && (A2X_BRESP_ORDER==1)) ? 1 : 0 **Parameter Name:** A2X_NUM_UWID
Number of Outstanding Secondary Port Writes	Defines the maximum number of outstanding Secondary Port write transactions per ID when responses are returned out-of-order. For more information on usage for this parameter, refer to "Outstanding Transaction Limits" in the DesignWare DW_axi_a2x Databook. **Values:** 3, ..., 64 **Default Value:** (A2X_OSW_EN==1)? 16 : 3 **Enabled:** (A2X_OSW_EN==1)? 1 : 0 **Parameter Name:** A2X_B_OSW_LIMIT_P1

Table 3-8 A2X Write Configuration Parameters (Continued)

Label	Description
Number of Outstanding Primary Port Writes	Defines the maximum number of outstanding Primary Port write addresses that the DW_axi_a2x can accept before receiving write data. For more information on usage for this parameter, refer to "Outstanding Transaction Limits" in the DesignWare DW_axi_a2x Databook. **Values:** 3, ..., 64 **Default Value:** (A2X_OSW_EN==1)? 16 : 3 **Enabled:** (A2X_PP_ENDIAN!=0)? 1 : (A2X_PP_DW<A2X_SP_DW)? 1 : ((A2X_PP_DW>A2X_SP_DW) && (A2X_WBUF_MODE==1))? 1 : 0 **Parameter Name:** A2X_PP_OSAW_LIMIT_P1
Number of Secondary Port Write Addresses	Defines the maximum number of Secondary Port write addresses that the DW_axi_a2x can send before sending data. This parameter is enabled for all configurations except AXI equal-sized configurations. For more information on usage for this parameter, refer to "Outstanding Transaction Limits" in the DesignWare DW_axi_a2x Databook. **Values:** 3, ..., 64 **Default Value:** 3 **Enabled:** (A2X_PP_MODE==0)? 1 : (A2X_OSW_EN==1)? 1 : 0 **Parameter Name:** A2X_SP_OSAW_LIMIT_P1

3.9 A2X Read Configuration Parameters

Table 3-9 A2X Read Configuration Parameters

Label	Description
A2X Read Configuration	
Read Data Interleaving	Enables support for Read Data interleaving. **Values:** ■ Disabled (0) ■ Enabled (1) **Default Value:** Disabled **Enabled:** Always **Parameter Name:** A2X_READ_INTLEV
Read Data Order	Selects the Secondary Port Read Data order. ■ In-Order - Read data is returned in same order as sent on Secondary Port channel. ■ Out-Of-Order - Read data may return from Secondary Port in different order than sent on Secondary Port write address channel. **Values:** ■ In-Order (0) ■ Out-of-Order (1) **Default Value:** (A2X_READ_INTLEV==1)? 1 : 0 **Enabled:** A2X_READ_INTLEV==0 **Parameter Name:** A2X_READ_ORDER
Number of URID	Selects the number of unique Read IDs for which the DW_axi_a2x may have outstanding transactions. When read data are returned in order, there is no restriction on the number of unique Read IDs. Enabled under the following conditions: ■ In Out-Of-Order or Interleaved configurations with: - AHB/AXI upsizing or downsizing configurations - AHB configurations with A2X_HINCR_RBCNT > 2^A2X_SP_BLW ■ In In-Order configurations, this parameter is hardcoded to 1 for: - AHB/AXI upsizing or downsizing configurations - AHB Configurations with A2X_HINCR_RBCNT > 2^A2X_SP_BLW For more information on usage for this parameter, refer to "Outstanding Transaction Limits" in the DesignWare DW_axi_a2x Databook. **Values:** 1, ..., (A2X_PP_MODE==0)? A2X_NUM_AHBM : 16 **Default Value:** 1 **Enabled:** ((A2X_OSR_EN==1) && ((A2X_READ_ORDER==1) || (A2X_READ_INTLEV==1)))? 1 : 0 **Parameter Name:** A2X_NUM_URID

Table 3-9 A2X Read Configuration Parameters (Continued)

Label	Description
Number of Outstanding Reads	Selects the maximum number of outstanding Secondary Port read transactions when read data are returned in order. Also selects the maximum number of outstanding Secondary Port read transactions per ID when read data are returned out-of-order. For more information on usage for this parameter, refer to "Outstanding Transaction Limits" in the DesignWare DW_axi_a2x Databook. **Values:** 3, ..., 64 **Default Value:** (A2X_OSR_EN==1)? 16 : 3 **Enabled:** ((A2X_LOCKED==1) || (A2X_OSR_EN==1))? 1 : 0 **Parameter Name:** A2X_OSR_LIMIT_P1

3.10 Buffer Depths Parameters

Table 3-10 Buffer Depths Parameters

Label	Description
Buffer Depths	
Write Address Buffer Depth	Sets the depth of the Write Address Channel buffer. **Values:** 2, ..., 32 **Default Value:** ((A2X_PP_MODE==0) && (A2X_BRESP_MODE==1) && (A2X_AHB_SPLIT_MODE==0))? 2 :4 **Enabled:** ((A2X_PP_MODE==0) && (A2X_BRESP_MODE==1) && (A2X_AHB_SPLIT_MODE==0))? 0 : 1 **Parameter Name:** A2X_AW_FIFO_DEPTH
Write Data Buffer Depth	Sets the depth of the Write Data Channel buffer. **Values:** 2, ..., 512 **Default Value:** [<functionof> A2X_WBUF_MODE A2X_SNF_AWLEN_DFLT] **Enabled:** Always **Parameter Name:** A2X_WD_FIFO_DEPTH
Write Response Buffer Depth	Sets the depth of the Write Response Channel buffer. In the AHB mode, this parameter is hardcoded to 2. **Values:** 2, ..., 32 **Default Value:** 2 **Enabled:** (A2X_PP_MODE==0) ? 0 : 1 **Parameter Name:** A2X_BRESP_FIFO_DEPTH
Read Address Buffer Depth	Sets the depth of the Read Address Channel buffer. **Values:** 2, ..., 32 **Default Value:** (A2X_PP_MODE==0)? 2 : 4 **Enabled:** ((A2X_PP_MODE==0) && (A2X_AHB_SPLIT_MODE==0))? 0 : 1 **Parameter Name:** A2X_AR_FIFO_DEPTH
Read Data Buffer Depth	Sets the depth of the Read Data Channel buffer. In the AHB mode with Locked support, the buffer depth is set to: (number_of_AHB_Masters * maximum_burst_length) This ensures that the DW_axi_a2x can store all outstanding Read transactions before issuing an AHB Locked transaction on the Secondary Port AXI channel. **Values:** 2, ..., 2048 **Default Value:** [<functionof> A2X_PP_MODE A2X_LOCKED A2X_NUM_AHBM A2X_LKMODE_MAX_PREFETCH A2X_RBUF_MODE A2X_SNF_ARLEN_DFLT] **Enabled:** ((A2X_LOCKED==1) && (A2X_PP_MODE==0)) ? 0 : 1 **Parameter Name:** A2X_RD_FIFO_DEPTH

3.11 AXI Low Power Mode Parameters

Table 3-11 AXI Low Power Mode Parameters

Label	Description
	AXI Low Power Mode
Low Power Interface	When enabled, the DW_axi_a2x supports Low-Power mode. **Values:** ■ Disable (0) ■ Enable (1) **Default Value:** Disable **Enabled:** Always **Parameter Name:** A2X_LOWPWR_IF
Low Power NOPX Count	Defines the number of clock cycles the DW_axi_a2x must be inactive before de-asserting CACTIVE. **Values:** 0, ..., 15 **Default Value:** 2 **Enabled:** (A2X_LOWPWR_IF==1)? 1 : 0 **Parameter Name:** A2X_LOWPWR_NOPX_CNT
Low Power Reset Count	Defines the number of clock cycles to hold the DW_axi_a2x in reset after a low-power request. **Values:** 0, ..., 15 **Default Value:** 2 **Enabled:** (A2X_LOWPWR_IF==1)? 1 : 0 **Parameter Name:** A2X_LOWPWR_RST_CNT

3.12 AXI SP Timing Mode Parameters

Table 3-12 AXI SP Timing Mode Parameters

Label	Description
	AXI SP Timing Mode
SP Read Address Channel Pipeline	Optional. Secondary Port Read Address Channel timing pipeline. **Values:** ■ Disable (0) ■ Enable (1) **Default Value:** Disable **Enabled:** Always **Parameter Name:** A2X_AR_SP_PIPELINE
SP Write Address Channel Pipeline	Optional. Secondary Port Write Address Channel timing pipeline. **Values:** ■ Disable (0) ■ Enable (1) **Default Value:** Disable **Enabled:** Always **Parameter Name:** A2X_AW_SP_PIPELINE

3.13 Assembler configuration Parameters

Table 3-13 Assembler configuration Parameters

Label	Description
Auto-Connect Split Mode	Controls the connection of the A2X_AHB_SPLIT_MODE parmaeter to the Configure Interface Parameter SplitCapable value is set to 1. **Values:** 0, 1 **Default Value:** 1 **Enabled:** This parameter is used in the coreAssembler. **Parameter Name:** A2X_AUTO_LINK_SPLIT_MODE

4 Signal Descriptions

This chapter details all possible I/O signals in the controller. For configurable IP titles, your actual configuration might not contain all of these signals.

Inputs are on the left of the signal diagrams; outputs are on the right.

Attention: For configurable IP titles, do not use this document to determine the exact I/O footprint of the controller. It is for reference purposes only.

When you configure the controller in coreConsultant, you must access the I/O signals for your actual configuration at workspace/report/IO.html or workspace/report/IO.xml after you have completed the report creation activity. That report comes from the exact same source as this chapter but removes all the I/O signals that are not in your actual configuration. This does not apply to non-configurable IP titles. In addition, all parameter expressions are evaluated to actual values. Therefore, the widths might change depending on your actual configuration.

Some expressions might refer to TCL functions or procedures (sometimes identified as **<functionof>)** that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the controller in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

In addition to describing the function of each signal, the signal descriptions in this chapter include the following information:

Active State: Indicates whether the signal is active high or active low. When a signal is not intended to be used in a particular application, then this signal needs to be tied or driven to the inactive state (opposite of the active state).

Registered: Indicates whether or not the signal is registered directly inside the IP boundary without intervening logic (excluding simple buffers). A value of No does not imply that the signal is not synchronous, only that there is some combinatorial logic between the signal's origin or destination register and the boundary of the controller. A value of N/A indicates that this information is not provided for this IP title.

Synchronous to: Indicates which clocks in the IP sample this input (drive for an output) when considering all possible configurations. A particular configuration might not have all of the clocks listed. This clock might not be the same as the clock that your application logic should use to clock (sample/drive) this pin. For more details, consult the clock section in the databook.

Exists: Names of configuration parameters that populate this signal in your configuration.

Validated by: Assertion or de-assertion of signals that validates the signal being described.

Attributes used with Synchronous To

- Clock name - The name of the clock that samples an input or drive and output.
- None - This attribute may be used for clock inputs, hard-coded outputs, feed-through (direct or combinatorial), dangling inputs, unused inputs and asynchronous outputs.
- Asynchronous - This attribute is used for asynchronous inputs and asynchronous resets.

The I/O signals are grouped as follows:

- Low Power Signals on page 103
- Debug Signals on page 104
- AHB Signals on page 105
- AHB INCR Slave Interface Signals on page 110
- AXI Primary Write Address Signals on page 112
- AXI Primary Write Data Signals on page 117
- AXI Primary Write Response Signals on page 119
- AXI Primary Read Address Signals on page 121
- AXI Primary Read Data Signals on page 126
- AXI Secondary Clock and Reset Signals on page 129
- AXI Secondary Write Address Signals on page 130
- AXI Secondary Write Data Signals on page 136
- AXI Secondary Write Response Signals on page 139
- AXI Secondary Read Address Signals on page 141
- AXI Secondary Read Data Signals on page 146

4.1 Low Power Signals

Table 4-1 Low Power Signals

Port Name	I/O	Description
csysreq	I	System low-power request from system clock controller. ■ De-asserted by system low power controller (LPC) to initiate entry into a low power state. ■ Asserted to initiate exit from a low power state. **Exists:** (A2X_LOWPWR_IF== 1) **Synchronous To:** (A2X_PP_MODE == 0) ? "hclk" : "clk_pp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
csysack	O	Low-power request acknowledgment. ■ De-asserted by DW_axi_a2x to acknowledge request to enter low-power state ■ Asserted by DW_axi_a2x to acknowledge request to exit low-power state **Exists:** (A2X_LOWPWR_IF== 1) **Synchronous To:** (A2X_PP_MODE == 0) ? "hclk" : "clk_pp" **Registered:** Yes **Power Domain:** SINGLE_DOMAIN **Active State:** High
cactive	O	Clock active request. De-asserted by DW_axi_a2x to tell system low-power controller (LPC) that clock can be removed. ■ 1: peripheral clock required ■ 0: peripheral clock not required **Note**: Clock can be removed on negative edge after cactive and csysack signals are sampled low (0). **Exists:** (A2X_LOWPWR_IF== 1) **Synchronous To:** (A2X_PP_MODE == 0) ? "hclk" : "clk_pp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

4.2 Debug Signals

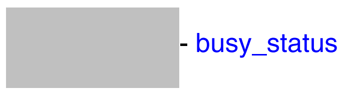


Table 4-2 Debug Signals

Port Name	I/O	Description
busy_status	O	A2X Busy Status. Indicates the status of the A2X. ■ Busy (1) ■ Not-Busy(0) **Exists:** Always **Synchronous To:** (A2X_LOWPWR_IF == 1) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp"): "None" **Registered:** (A2X_LOWPWR_IF == 1) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.3 AHB Signals

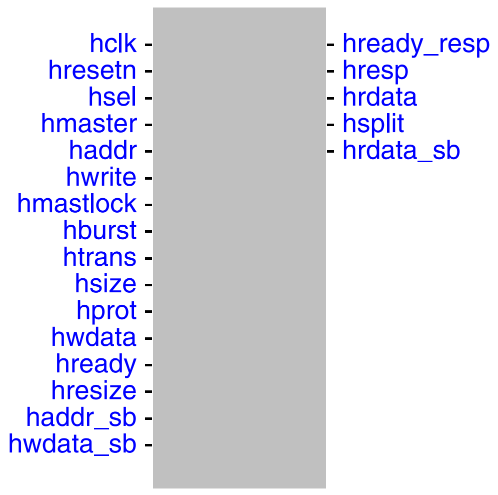


Table 4-3 AHB Signals

Port Name	I/O	Description
hclk	I	AHB Clock. This clock times all AHB bus transfers. All signal timings are related to the rising edge of hclk. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hresetn	I	AHB Asynchronous Reset. The bus reset signal is active low and is used to reset the system and the bus on the DesignWare Synthesizable Components interface. Asynchronous assertion, synchronous de-assertion. The reset must be deasserted synchronously after the rising edge of hclk. DW_axi_a2x does not contain logic to perform this synchronization, so it must be provided externally. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** Low

Table 4-3 AHB Signals (Continued)

Port Name	I/O	Description
hsel	I	AHB Peripheral Select. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hmaster[(A2X_IDW-1):0]	I	AHB Master Identification. Indicates which master currently has ownership of the address and control bus. This is generated by the arbiter. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
haddr[(A2X_PP_AWIDTH-1):0]	I	AHB Address Bus. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hwrite	I	AHB Transfer Write Control. When high, this signal indicates a write transfer. When low, this signal indicates a read transfer. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
hmastlock	I	AHB Locked Transfer Control. Asserted by bus master to indicate that it wishes to carry out a locked transaction. This signal is unused when the secondary interface is configured as AXI4/ACELITE. In those configurations, it is included for interface consistency only. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-3 AHB Signals (Continued)

Port Name	I/O	Description
hburst[2:0]	I	AHB Transfer Type. Indicates if the transfer constitutes part of a burst. Each master in the system has its own hburst bus. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
htrans[1:0]	I	AHB Transfer Control. Indicates the type of transfer being performed. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hsize[2:0]	I	AHB Transfer Size. Indicates size of transfer. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hprot[3:0]	I	AHB Transfer Protection Control. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hwdata[(A2X_PP_DWIDTH-1):0]	I	AHB Transfer Write Data. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hready	I	AHB Ready Response from selected slave. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-3 AHB Signals (Continued)

Port Name	I/O	Description
hready_resp	O	AHB Ready Response from DW_axi_a2x. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** Yes **Power Domain:** SINGLE_DOMAIN **Active State:** High
hresp[1:0]	O	AHB Transfer Response from DW_axi_a2x. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** Yes **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hrdata[(A2X_PP_DWIDTH-1):0]	O	AHB Transfer Read Data. The read data bus is used to transfer data from the DW_axi_a2x to the AHB bus master during read operations. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** hclk **Registered:** Yes **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hsplit[15:0]	O	AHB Transfer Split. This bus indicates to the arbiter which master may proceed to complete a split transaction. **Exists:** (A2X_PP_MODE == 0) **Synchronous To:** (A2X_AHB_SPLIT_MODE == 1) ? "hclk":"None" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hresize	I	Non-AHB Protocol Signal. When asserted high, the DW_axi_a2x upsizes the transaction; when low, the DW_axi_a2x does not resize the transaction. **Exists:** (A2X_PP_MODE == 0) && (A2X_UPSIZE == 1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
haddr_sb[(A2X_INT_AWSBW-1):0]	I	Non-AHB Protocol Signal. Address sideband bus. **Exists:** (A2X_PP_MODE == 0) && (A2X_HASBW != 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-3 AHB Signals (Continued)

Port Name	I/O	Description
hwdata_sb[(A2X_INT_WSBW-1):0]	I	Non-AHB Protocol Signal. Write data sideband bus. **Exists:** (A2X_PP_MODE == 0) && (A2X_WSBW != 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hrdata_sb[(A2X_INT_RSBW-1):0]	O	Non-AHB Protocol Signal. Read data sideband bus. **Exists:** (A2X_PP_MODE == 0) && (A2X_RSBW != 0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.4 AHB INCR Slave Interface Signals

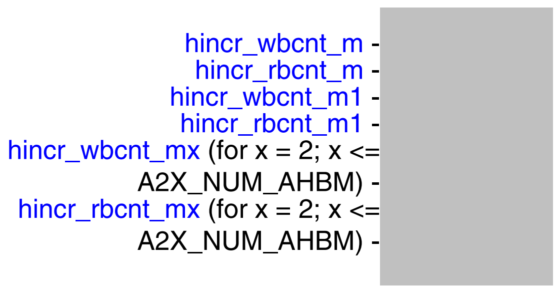

hincr_wbcnt_m -
hincr_rbcnt_m -
hincr_wbcnt_m1 -
hincr_rbcnt_m1 -
hincr_wbcnt_mx (for x = 2; x <= A2X_NUM_AHBM) -
hincr_rbcnt_mx (for x = 2; x <= A2X_NUM_AHBM) -

Table 4-4 AHB INCR Slave Interface Signals

Port Name	I/O	Description
hincr_wbcnt_m[(A2X_HINCR_LEN_W-1):0]	I	AHB Master Write INCR Transaction Length. **Exists:** (A2X_PP_MODE == 0) && (A2X_HINCR_HCBCNT==0) && (A2X_SINGLE_WBCNT==1) && (A2X_BRESP_MODE!=1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hincr_rbcnt_m[(A2X_HINCR_LEN_W-1):0]	I	AHB Master Read INCR Transaction Length. **Exists:** (A2X_PP_MODE == 0) && (A2X_HINCR_HCBCNT==0) && (A2X_SINGLE_RBCNT==1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hincr_wbcnt_m1[(A2X_HINCR_LEN_W-1):0]	I	AHB Master 1 Write INCR Transaction Length. **Exists:** (A2X_PP_MODE == 0) && (A2X_HINCR_HCBCNT==0) && (A2X_SINGLE_WBCNT==0) && (A2X_BRESP_MODE!=1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hincr_rbcnt_m1[(A2X_HINCR_LEN_W-1):0]	I	AHB Master 1 Read INCR Transaction Length. **Exists:** (A2X_PP_MODE == 0) && (A2X_HINCR_HCBCNT==0) && (A2X_SINGLE_RBCNT==0) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-4 AHB INCR Slave Interface Signals (Continued)

Port Name	I/O	Description
hincr_wbcnt_mx[(A2X_HINCR_LEN_W-1):0] (for x = 2; x <= A2X_NUM_AHBM)	I	AHB Master x Write INCR Transaction Length. **Exists:** (A2X_PP_MODE == 0) && (A2X_HINCR_HCBCNT==0) && (A2X_SINGLE_WBCNT==0) && (A2X_NUM_AHBM>=x) && (A2X_BRESP_MODE!=1) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
hincr_rbcnt_mx[(A2X_HINCR_LEN_W-1):0] (for x = 2; x <= A2X_NUM_AHBM)	I	AHB Master x Read INCR Transaction Length. **Exists:** (A2X_PP_MODE == 0) && (A2X_HINCR_HCBCNT==0) && (A2X_SINGLE_RBCNT==0) && (A2X_NUM_AHBM>=x) **Synchronous To:** hclk **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.5 AXI Primary Write Address Signals

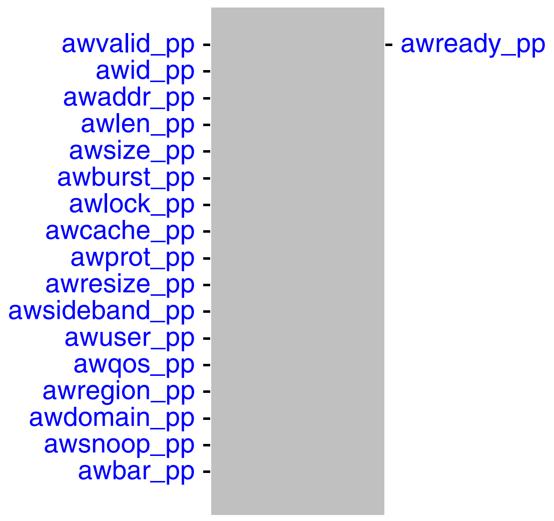


Table 4-5 AXI Primary Write Address Signals

Port Name	I/O	Description
awready_pp	O	AXI Write Address Ready. Indicates that the slave is ready to accept an address and associated control signals. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
awvalid_pp	I	Primary Port AXI Write Address Valid. Indicates that valid write address and control information are available. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
awid_pp[(A2X_IDW-1):0]	I	Primary Port AXI Write Address ID. Identification tag for the write address group of signals. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-5 AXI Primary Write Address Signals (Continued)

Port Name	I/O	Description
awaddr_pp[(A2X_PP_AWIDTH-1):0]	I	Primary Port AXI Write Address. Specifies the address of the AXI write burst transaction. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awlen_pp[(A2X_BLWIDTH-1):0]	I	Primary Port AXI Write Burst Length. The number of transfers in a burst associated with the write address. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awsize_pp[2:0]	I	Primary Port AXI Write Burst Size. The size of each transfer in a burst. Byte lane strobes indicate exactly which byte lanes to update. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awburst_pp[1:0]	I	Primary Port AXI Write Burst Type. Coupled with the size, burst type details how the address for each transfer within a burst is calculated. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awlock_pp[(A2X_INT_LTW-1):0]	I	AXI Write Lock Type. Provides additional information about the atomic characteristics of the transfer. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-5 AXI Primary Write Address Signals (Continued)

Port Name	I/O	Description
awcache_pp[3:0]	I	Primary Port AXI Write Cache Type. Indicates the bufferable, cacheable/modifiable, write-through, write-back, and allocate attributes of the transaction. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awprot_pp[2:0]	I	Primary Port AXI Write Protection Type. Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awresize_pp	I	Primary Port Write Channel Non-AXI Protocol Signal. ■ When asserted high, DW_axi_a2x upsizes the transaction if: - A2X_AXI_INTERFACE_TYPE = AXI3; OR - A2X_AXI_INTERFACE_TYPE = AXI4/ACELITE and modifiable bit (awcache_pp[1]) is high ■ When low, the DW_axi_a2x does not resize the transaction. **Exists:** (A2X_PP_MODE == 1) && (A2X_UPSIZE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
awsideband_pp[(A2X_INT_AWSBW-1):0]	I	Optional. Primary Port Write address Sideband Bus. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named awsideband_pp ■ When the AXI4/ACELITE interface is enabled, this signal is named awuser_pp. **Exists:** (A2X_AXI_INTERFACE_TYPE == 0) && (A2X_PP_MODE == 1) && (A2X_AWSBW != 0) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-5 AXI Primary Write Address Signals (Continued)

Port Name	I/O	Description
awuser_pp[(A2X_INT_AWSBW-1):0]	I	Optional. Primary Port Write address User Signals. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named awsideband_pp ■ When the AXI4/ACELITE interface is enabled, this signal is named awuser_pp. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_PP_MODE == 1) && (A2X_AWSBW != 0) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awqos_pp[(A2X_INT_QOSW-1):0]	I	Quality of Service identifier, conveying priority information associated with each transaction at the write address channel of the master port. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_INC_QOS == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awregion_pp[(A2X_INT_REGIONW-1):0]	I	Primary Port Write Channel Region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_INC_REGION == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awdomain_pp[(A2X_INT_DOMAINW-1):0]	I	This signal indicates the shareability domain of a write transaction (Primary Port). **Exists:** (A2X_AXI_INTERFACE_TYPE == 2) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-5 AXI Primary Write Address Signals (Continued)

Port Name	I/O	Description
awsnoop_pp[(A2X_INT_WSNOOPW-1):0]	I	This signal indicates the transaction type for shareable write transactions (Primary Port). **Exists:** (A2X_AXI_INTERFACE_TYPE == 2) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awbar_pp[(A2X_INT_BARW-1):0]	I	This signal indicates a write barrier transaction (Primary Port). **Exists:** (A2X_AXI_INTERFACE_TYPE == 2) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.6 AXI Primary Write Data Signals

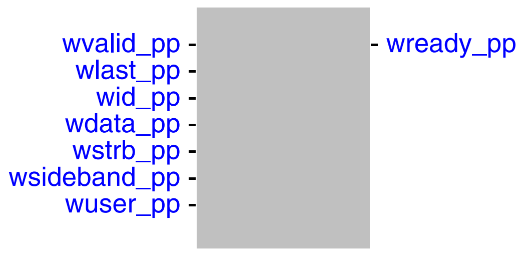


Table 4-6 AXI Primary Write Data Signals

Port Name	I/O	Description
wready_pp	O	Primary Port AXI Write Ready. Indicates that the slave can accept the write data. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
wvalid_pp	I	Primary Port AXI Write Valid. Indicates that valid write data and strobes are available. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
wlast_pp	I	Primary Port AXI Write Last. Indicates the last transfer in a write burst. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
wid_pp[(A2X_IDW-1):0]	I	AXI Write ID. ID tag of the write data transfer. Must match the awid_pp value of the write transaction. **Exists:** (A2X_AXI_INTERFACE_TYPE == 0) && (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-6 AXI Primary Write Data Signals (Continued)

Port Name	I/O	Description
wdata_pp[(A2X_PP_DWIDTH-1):0]	I	Primary Port AXI Write Data. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
wstrb_pp[(A2X_PP_WSTRB_DW-1):0]	I	Primary Port AXI Write Strobe. Indicates which byte lanes to update in memory. There is one data strobe for each eight bits of the write bus; that is, wstrb[i] corresponds to wdata[8n+7 : 8n]. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
wsideband_pp[(A2X_INT_WSBW-1):0]	I	Optional. Primary Port Write data Sideband Bus. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named wsideband_pp. ■ When the AXI4/ACELITE interface is enabled, this signal is named wuser_pp. **Exists:** (A2X_AXI_INTERFACE_TYPE == 0) && (A2X_PP_MODE == 1) && (A2X_WSBW != 0) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
wuser_pp[(A2X_INT_WSBW-1):0]	I	Optional. Primary Port Write data User Signals. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named wsideband_pp. ■ When the AXI4/ACELITE interface is enabled, this signal is named wuser_pp. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_PP_MODE == 1) && (A2X_WSBW != 0) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.7 AXI Primary Write Response Signals

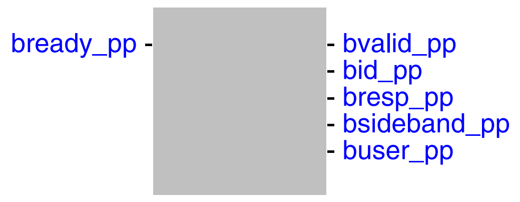


Table 4-7 AXI Primary Write Response Signals

Port Name	I/O	Description
bready_pp	I	Primary Port AXI Write Response Ready. Indicates that the master can accept the write response information. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
bvalid_pp	O	Primary Port AXI Write Response Valid. Indicates that a valid write response is available. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** ((A2X_CLK_MODE == 0) || ((A2X_PP_MODE == 1) && (A2X_BRESP_MODE == 0))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
bid_pp[(A2X_IDW-1):0]	O	Primary Port AXI Write Response ID. Must match the awid value of the write transaction to which the slave is responding. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** ((A2X_CLK_MODE == 0) || ((A2X_PP_MODE == 1) && (A2X_BRESP_MODE == 0))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
bresp_pp[1:0]	O	Primary Port AXI Write Response. Indicates the status of the write transaction. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** ((A2X_CLK_MODE == 0) && (A2X_BRESP_MODE != 0)) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-7 AXI Primary Write Response Signals (Continued)

Port Name	I/O	Description
bsideband_pp[(A2X_INT_BSBW-1):0]	O	Optional. Primary Port Write Response Sideband Bus. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named bsideband_pp. ■ When the AXI4/ACELITE interface is enabled, this signal is named buser_pp. **Exists:** (A2X_AXI_INTERFACE_TYPE == 0) && (A2X_PP_MODE == 1) && (A2X_BSBW != 0) **Synchronous To:** clk_pp **Registered:** ((A2X_CLK_MODE == 0) && (A2X_BRESP_MODE != 0)) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
buser_pp[(A2X_INT_BSBW-1):0]	O	Optional. Primary Port Write Response User Signals. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named bsideband_pp. ■ When the AXI4/ACELITE interface is enabled, this signal is named buser_pp. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_PP_MODE == 1) && (A2X_BSBW != 0) **Synchronous To:** clk_pp **Registered:** ((A2X_CLK_MODE == 0) && (A2X_BRESP_MODE != 0)) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.8 AXI Primary Read Address Signals

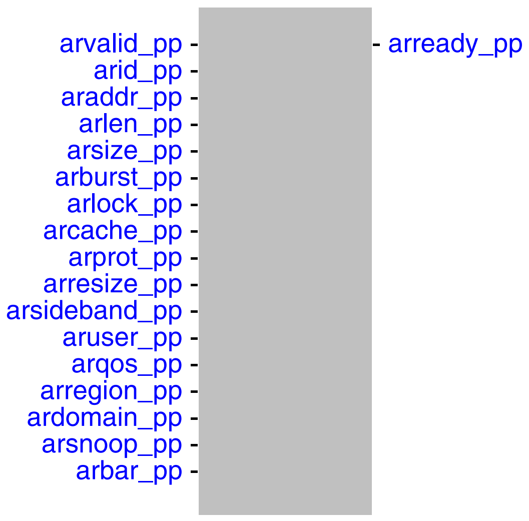


Table 4-8 AXI Primary Read Address Signals

Port Name	I/O	Description
arready_pp	O	Primary Port AXI Read Address Ready. Indicates that the slave is ready to accept an address and associated control signals. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
arvalid_pp	I	Primary Port AXI Read Address Valid. Indicates that valid read address and control information are available. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
arid_pp[(A2X_IDW-1):0]	I	Primary Port AXI Read Address ID. Identification tag for the read address group of signals. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-8 AXI Primary Read Address Signals (Continued)

Port Name	I/O	Description
araddr_pp[(A2X_PP_AWIDTH-1):0]	I	Primary Port AXI Read Address. Specifies the address of an AXI read burst transaction. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arlen_pp[(A2X_BLWIDTH-1):0]	I	Primary Port AXI Read Burst Length. The number of transfers in a burst associated with the read address. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arsize_pp[2:0]	I	Primary Port AXI Read Burst Size. The size of each transfer in a burst. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arburst_pp[1:0]	I	Primary Port AXI Read Burst Type. Coupled with the size, burst type details how the address for each transfer within a burst is calculated. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arlock_pp[(A2X_INT_LTW-1):0]	I	Primary Port AXI Read Lock Type. Provides additional information about the atomic characteristics of the transfer. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-8 AXI Primary Read Address Signals (Continued)

Port Name	I/O	Description
arcache_pp[3:0]	I	Primary Port AXI Read Cache Type. Provides additional information about the cacheable characteristics of the transfer. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arprot_pp[2:0]	I	Primary Port AXI Read Protection Type. Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arresize_pp	I	Secondary Port Read Channel Non-AXI Protocol Signal. ■ When asserted high, DW_axi_a2x upsizes the transaction if: - A2X_AXI_INTERFACE_TYPE = AXI3; OR - A2X_AXI_INTERFACE_TYPE = AXI4/ACELITE and modifiable bit (awcache_pp[1]) is high ■ When low, the DW_axi_a2x does not resize the transaction. **Exists:** (A2X_PP_MODE == 1) && (A2X_UPSIZE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
arsideband_pp[(A2X_INT_ARSBW-1):0]	I	Optional. Primary Port Read Address Sideband bus. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named arsideband_pp. ■ When the AXI4/ACELITE interface is enabled, this signal is named aruser_pp. **Exists:** (A2X_AXI_INTERFACE_TYPE == 0) && (A2X_PP_MODE == 1) && (A2X_ARSBW != 0) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-8 AXI Primary Read Address Signals (Continued)

Port Name	I/O	Description
aruser_pp[(A2X_INT_ARSBW-1):0]	I	Optional. Primary Port Read Address User Signals. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named arsideband_pp. ■ When the AXI4/ACELITE interface is enabled, this signal is named aruser_pp. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_PP_MODE == 1) && (A2X_ARSBW != 0) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arqos_pp[(A2X_INT_QOSW-1):0]	I	Quality of Service identifier, conveying priority information associated with each transaction at the read address channel of the master port. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_INC_QOS == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arregion_pp[(A2X_INT_REGIONW-1):0]	I	Primary Port Read Channel Region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_INC_REGION == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
ardomain_pp[(A2X_INT_DOMAINW-1):0]	I	This signal indicates the shareability domain of a read transaction (Primary Port). **Exists:** (A2X_AXI_INTERFACE_TYPE == 2) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-8 AXI Primary Read Address Signals (Continued)

Port Name	I/O	Description
arsnoop_pp[(A2X_INT_RSNOOPW-1):0]	I	This signal indicates the transaction type for shareable read transactions (Primary Port). **Exists:** (A2X_AXI_INTERFACE_TYPE == 2) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arbar_pp[(A2X_INT_BARW-1):0]	I	This signal indicates a read barrier transaction (Primary Port). **Exists:** (A2X_AXI_INTERFACE_TYPE == 2) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.9 AXI Primary Read Data Signals

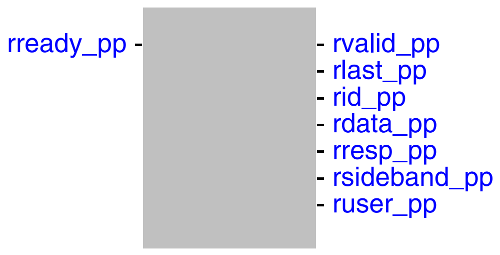


Table 4-9 AXI Primary Read Data Signals

Port Name	I/O	Description
rready_pp	I	Primary Port AXI Read Ready. Indicates that the slave can accept the read data. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
rvalid_pp	O	Primary Port AXI Read Valid. Indicates that valid read data is available. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** (A2X_CLK_MODE == 0)? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
rlast_pp	O	Primary Port AXI Read Last. Indicates the last transfer in a read burst. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** ((A2X_CLK_MODE == 0) && ((A2X_PP_DW > A2X_SP_DW) || (A2X_PP_DW == A2X_SP_DW)))? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-9 AXI Primary Read Data Signals (Continued)

Port Name	I/O	Description
rid_pp[(A2X_IDW-1):0]	O	Primary Port AXI Read ID. ID tag of the read data transfer. Must match the arid value of the read transaction. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** ((A2X_CLK_MODE == 0) && ((A2X_PP_DW > A2X_SP_DW) || (A2X_PP_DW == A2X_SP_DW) || ((A2X_PP_DW < A2X_SP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
rdata_pp[(A2X_PP_DWIDTH-1):0]	O	Primary Port AXI Read Data. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** ((A2X_CLK_MODE == 0) && ((A2X_PP_DW > A2X_SP_DW) || (A2X_PP_DW == A2X_SP_DW)))? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
rresp_pp[1:0]	O	Primary Port AXI Read Response. Indicates the status of the read transfer. **Exists:** (A2X_PP_MODE == 1) **Synchronous To:** clk_pp **Registered:** ((A2X_CLK_MODE == 0) && ((A2X_PP_DW > A2X_SP_DW) || (A2X_PP_DW == A2X_SP_DW) || ((A2X_PP_DW < A2X_SP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
rsideband_pp[(A2X_INT_RSBW-1):0]	O	Optional. Primary Port Read data sideband bus. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named rsideband_pp. ■ When the AXI4/ACELITE interface is enabled, this signal is named ruser_pp. **Exists:** (A2X_AXI_INTERFACE_TYPE == 0) && (A2X_PP_MODE == 1) && (A2X_RSBW != 0) **Synchronous To:** clk_pp **Registered:** ((A2X_CLK_MODE == 0) && ((A2X_PP_DW > A2X_SP_DW) || (A2X_PP_DW == A2X_SP_DW) || ((A2X_PP_DW < A2X_SP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-9 AXI Primary Read Data Signals (Continued)

Port Name	I/O	Description
ruser_pp[(A2X_INT_RSBW-1):0]	O	Optional. Primary Port Read data User Signals. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named rsideband_pp. ■ When the AXI4/ACELITE interface is enabled, this signal is named ruser_pp. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_PP_MODE == 1) && (A2X_RSBW != 0) **Synchronous To:** clk_pp **Registered:** ((A2X_CLK_MODE == 0) && ((A2X_PP_DW > A2X_SP_DW) || (A2X_PP_DW == A2X_SP_DW) || ((A2X_PP_DW < A2X_SP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.10 AXI Secondary Clock and Reset Signals

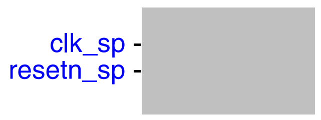


Table 4-10 AXI Secondary Clock and Reset Signals

Port Name	I/O	Description
clk_sp	I	Secondary Port Clock signal. All Secondary Port Signals are synchronous to this clock. **Exists:** (A2X_CLK_MODE != 0) **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** N/A; all signals sampled on rising edge of clock
resetn_sp	I	Secondary Port Asynchronous Reset. Active-low pin that asynchronously resets the Secondary Port logic to its default state. Asynchronous assertion, synchronous de-assertion. The reset must be synchronously de-asserted after rising edge of clk_sp. DW_axi_a2x does not contain logic to perform this synchronization, so it must be provided externally. **Exists:** (A2X_CLK_MODE != 0) **Synchronous To:** None **Registered:** N/A **Power Domain:** SINGLE_DOMAIN **Active State:** Low

4.11 AXI Secondary Write Address Signals

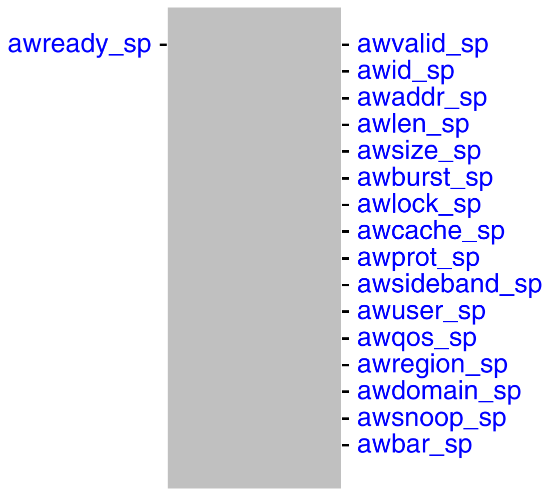


Table 4-11 AXI Secondary Write Address Signals

Port Name	I/O	Description
awready_sp	I	AXI Write Address Ready. Indicates that the master is ready to accept an address and associated control signals. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
awvalid_sp	O	Secondary Port AXI Write Address Valid. Indicates that valid write address and control information are available. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** (A2X_AW_SP_PIPELINE == 1) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-11 AXI Secondary Write Address Signals (Continued)

Port Name	I/O	Description
awid_sp[(A2X_SP_IDW-1):0]	O	Secondary Port AXI Write Address ID. Identification tag for the write address group of signals. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awaddr_sp[(A2X_SP_AWIDTH-1):0]	O	Secondary Port AXI Write Address. Specifies the address of the secondary port AXI write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in a burst. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awlen_sp[(A2X_SP_BLWIDTH-1):0]	O	Secondary Port AXI Write Burst Length. The number of transfers in a burst associated with the write address. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awsize_sp[2:0]	O	Secondary Port AXI Write Burst Size. The size of each transfer in a burst. Byte lane strobes indicate exactly which byte lanes to update. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-11 AXI Secondary Write Address Signals (Continued)

Port Name	I/O	Description
awburst_sp[1:0]	O	Secondary Port AXI Write Burst Type. Coupled with the size, burst type details how the address for each transfer within a burst is calculated. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awlock_sp[(A2X_INT_LTW-1):0]	O	AXI Write Lock Type. Provides additional information about the atomic characteristics of the transfer. ■ 2 bits when A2X_AXI_INTERFACE_TYPE=AXI3 ■ 1 bit when A2X_AXI_INTERFACE_TYPE=AXI4/ACELITE **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_PP_DW)== (A2X_SP_DW))&& (A2X_LOCKED == 0) && (A2X_PP_MODE == 1))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awcache_sp[3:0]	O	Secondary Port AXI Write Cache Type. Indicates the bufferable, cacheable/modifiable, write-through, write-back, and allocate attributes of the transaction. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-11 AXI Secondary Write Address Signals (Continued)

Port Name	I/O	Description
awprot_sp[2:0]	O	Secondary Port AXI Write Protection Type. Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awsideband_sp[(A2X_INT_AWSBW-1):0]	O	Optional. Secondary Port Write address Sideband Bus. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named awsideband_sp ■ When the AXI4/ACELITE interface is enabled, this signal is named awuser_sp. **Exists:** (A2X_AXI_INTERFACE_TYPE == 0) && (A2X_AWSBW != 0) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awuser_sp[(A2X_INT_AWSBW-1):0]	O	Optional. Secondary Port Write address User Signals. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named awsideband_sp ■ When the AXI4/ACELITE interface is enabled, this signal is named awuser_sp. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_AWSBW != 0) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-11 AXI Secondary Write Address Signals (Continued)

Port Name	I/O	Description
awqos_sp[(A2X_INT_QOSW-1):0]	O	Quality of Service identifier, conveying priority information associated with each transaction at the write address channel of the secondary port. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_INC_QOS == 1) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awregion_sp[(A2X_INT_REGIONW-1):0]	O	Secondary Port Write Channel Region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_INC_REGION == 1) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awdomain_sp[(A2X_INT_DOMAINW-1):0]	O	This signal indicates the shareability domain of a write transaction (Secondary Port). **Exists:** (A2X_AXI_INTERFACE_TYPE == 2) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
awsnoop_sp[(A2X_INT_WSNOOPW-1):0]	O	This signal indicates the transaction type for shareable write transactions (Secondary Port). **Exists:** (A2X_AXI_INTERFACE_TYPE == 2) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-11 AXI Secondary Write Address Signals (Continued)

Port Name	I/O	Description
awbar_sp[(A2X_INT_BARW-1):0]	O	This signal indicates a write barrier transaction (Secondary Port). **Exists:** (A2X_AXI_INTERFACE_TYPE == 2) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AW_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1))))? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.12 AXI Secondary Write Data Signals

wready_sp
wvalid_sp
wlast_sp
wid_sp
wdata_sp
wstrb_sp
wsideband_sp
wuser_sp

Table 4-12 AXI Secondary Write Data Signals

Port Name	I/O	Description
wready_sp	I	Secondary Port AXI Write Ready. Indicates that the slave can accept the write data. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
wvalid_sp	O	Secondary Port AXI Write Valid. Indicates that valid write data and strobes are available. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
wlast_sp	O	Secondary Port AXI Write Last. Indicates the last transfer in a write burst. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE ==1) && (A2X_CLK_MODE == 0)) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-12 AXI Secondary Write Data Signals (Continued)

Port Name	I/O	Description
wid_sp[(A2X_SP_IDW-1):0]	O	AXI Write ID. Identification tag of the write data transfer. Must match the awid_sp value of the write transaction. **Exists:** (A2X_AXI_INTERFACE_TYPE == 0) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** (((A2X_SP_DW == A2X_PP_DW) || ((A2X_PP_MODE==0) || (A2X_PP_DW>A2X_SP_DW) || (A2X_PP_DW<A2X_SP_DW))) && (A2X_CLK_MODE == 0)) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
wdata_sp[(A2X_SP_DWIDTH-1):0]	O	Secondary Port AXI Write Data. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((((A2X_SP_DW == A2X_PP_DW) || (A2X_PP_DW<A2X_SP_DW)) && (A2X_PP_MODE==1) && (A2X_SP_ENDIAN == 0)) && (A2X_CLK_MODE == 0)) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
wstrb_sp[(A2X_SP_WSTRB_DW-1):0]	O	Secondary Port AXI Write Strobe. Indicates which byte lanes to update in memory. There is one data strobe for each eight bits of the write bus; that is, wstrb[i] corresponds to wdata[8n+7 : 8n]. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((((A2X_SP_DW == A2X_PP_DW) || (A2X_PP_DW<A2X_SP_DW)) && (A2X_PP_MODE==1)) && (A2X_CLK_MODE == 0)) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-12 **AXI Secondary Write Data Signals (Continued)**

Port Name	I/O	Description
wsideband_sp[(A2X_INT_WSBW-1):0]	O	Optional. Secondary Port Write data Sideband Bus. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. When the AXI3 interface is enabled, this signal is named wsideband_sp. When the AXI4/ACELITE interface is enabled, this signal is named wuser_sp. **Exists:** (A2X_AXI_INTERFACE_TYPE == 0) && (A2X_WSBW != 0) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** (((A2X_SP_DW == A2X_PP_DW) || ((A2X_PP_MODE==0) || (A2X_PP_DW>A2X_SP_DW) || (A2X_PP_DW<A2X_SP_DW))) && (A2X_CLK_MODE == 0)) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
wuser_sp[(A2X_INT_WSBW-1):0]	O	Optional. Secondary Port Write data User Signals. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. When the AXI3 interface is enabled, this signal is named wsideband_sp. When the AXI4/ACELITE interface is enabled, this signal is named wuser_sp. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_WSBW != 0) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** (((A2X_SP_DW == A2X_PP_DW) || ((A2X_PP_MODE==0) || (A2X_PP_DW>A2X_SP_DW) || (A2X_PP_DW<A2X_SP_DW))) && (A2X_CLK_MODE == 0)) ? "Yes" : "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.13 AXI Secondary Write Response Signals

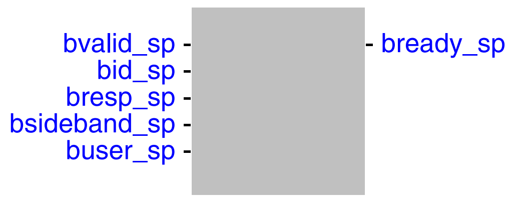


Table 4-13 AXI Secondary Write Response Signals

Port Name	I/O	Description
bready_sp	O	Secondary Port AXI Write Response Ready. Indicates that the master can accept the write response information. **Exists:** Always **Synchronous To:** (A2X_BRESP_MODE == 1) ?((A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp") : "None" **Registered:** ((A2X_CLK_MODE == 0) && ((A2X_BRESP_MODE==1) && (A2X_PP_DW == A2X_SP_DW)))? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
bvalid_sp	I	Secondary Port AXI Write Response Valid. Indicates that a valid write response is available. **Exists:** Always **Synchronous To:** ((A2X_BRESP_MODE == 0) && (A2X_LOCKED == 0) && (A2X_LOWPWR_IF == 0)) ?"None" :((A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp") **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
bid_sp[(A2X_SP_IDW-1):0]	I	Secondary Port AXI Write Response ID. Must match the awid value of the write transaction to which the slave is responding. **Exists:** Always **Synchronous To:** (A2X_BRESP_MODE != 0) ?((A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp") : "None" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-13 AXI Secondary Write Response Signals (Continued)

Port Name	I/O	Description
bresp_sp[1:0]	I	Secondary Port AXI Write Response. Indicates the status of the write transaction. **Exists:** Always **Synchronous To:** (A2X_BRESP_MODE != 0) ?((A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp") : "None" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
bsideband_sp[(A2X_INT_BSBW-1):0]	I	Optional. Secondary Port Write Response Sideband bus. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named bsideband_sp. ■ When the AXI4/ACELITE interface is enabled, this signal is named buser_sp. **Exists:** (A2X_AXI_INTERFACE_TYPE == 0) && (A2X_BSBW != 0) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
buser_sp[(A2X_INT_BSBW-1):0]	I	Optional. Secondary Port Write Response User Signals. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named bsideband_sp. ■ When the AXI4/ACELITE interface is enabled, this signal is named buser_sp. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_BSBW != 0) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.14 AXI Secondary Read Address Signals

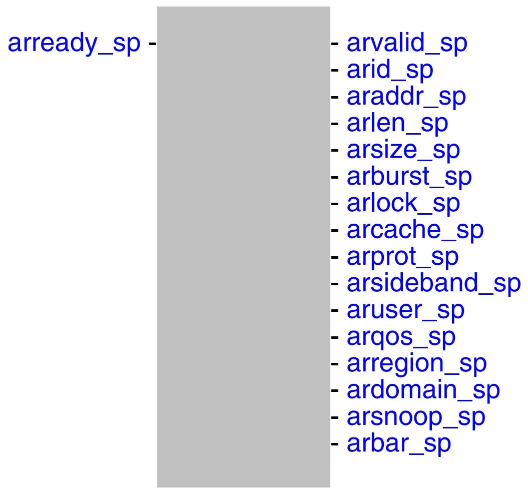


Table 4-14 AXI Secondary Read Address Signals

Port Name	I/O	Description
arready_sp	I	Secondary Port AXI Read Address Ready. Indicates that the slave is ready to accept an address and associated control signals. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
arvalid_sp	O	Secondary Port AXI Read Address Valid. Indicates that valid read address and control information are available. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** (A2X_AR_SP_PIPELINE == 1) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-14 AXI Secondary Read Address Signals (Continued)

Port Name	I/O	Description
arid_sp[(A2X_SP_IDW-1):0]	O	Secondary Port AXI Read Address ID. Identification tag for the read address group of signals. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
araddr_sp[(A2X_SP_AWIDTH-1):0]	O	Secondary Port AXI Read Address. Specifies the address of the secondary port AXI read burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in a burst. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arlen_sp[(A2X_SP_BLWIDTH-1):0]	O	Secondary Port AXI Read Burst Length. The number of transfers in a burst associated with the read address. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arsize_sp[2:0]	O	Secondary Port AXI Read Burst Size. The size of each transfer in a burst. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-14 AXI Secondary Read Address Signals (Continued)

Port Name	I/O	Description
arburst_sp[1:0]	O	Secondary Port AXI Read Burst Type. Coupled with the size, burst type details how the address for each transfer within a burst is calculated. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arlock_sp[(A2X_INT_LTW-1):0]	O	Secondary Port AXI Read Lock Type. Provides additional information about the atomic characteristics of the transfer. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)) && (A2X_LOCKED == 0))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arcache_sp[3:0]	O	Secondary Port AXI Read Cache Type. Provides additional information about the cacheable characteristics of the transfer. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arprot_sp[2:0]	O	Secondary Port AXI Read Protection Type. Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-14 AXI Secondary Read Address Signals (Continued)

Port Name	I/O	Description
arsideband_sp[(A2X_INT_ARSBW-1):0]	O	Optional. Secondary Port Read address sideband bus. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named arsideband_sp. ■ When the AXI4/ACELITE interface is enabled, this signal is named aruser_sp. **Exists:** (A2X_AXI_INTERFACE_TYPE == 0) && (A2X_ARSBW != 0) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
aruser_sp[(A2X_INT_ARSBW-1):0]	O	Optional. Secondary Port Read address User Signals. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named arsideband_sp. ■ When the AXI4/ACELITE interface is enabled, this signal is named aruser_sp. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_ARSBW != 0) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_PP_DW)== (A2X_SP_DW)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arqos_sp[(A2X_INT_QOSW-1):0]	O	Quality of Service identifier, conveying priority information associated with each transaction at the read address channel of the slave port. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_INC_QOS == 1) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-14 AXI Secondary Read Address Signals (Continued)

Port Name	I/O	Description
arregion_sp[(A2X_INT_REGIONW-1):0]	O	Secondary Port Read Channel Region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_INC_REGION == 1) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
ardomain_sp[(A2X_INT_DOMAINW-1):0]	O	This signal indicates the shareability domain of a read transaction (Secondary Port). **Exists:** (A2X_AXI_INTERFACE_TYPE == 2) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arsnoop_sp[(A2X_INT_RSNOOPW-1):0]	O	This signal indicates the transaction type for shareable read transactions (Secondary Port). **Exists:** (A2X_AXI_INTERFACE_TYPE == 2) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
arbar_sp[(A2X_INT_BARW-1):0]	O	This signal indicates a read barrier transaction (Secondary Port). **Exists:** (A2X_AXI_INTERFACE_TYPE == 2) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_AR_SP_PIPELINE == 1) || ((A2X_CLK_MODE == 0) && ((A2X_SP_DW == A2X_PP_DW) && (A2X_PP_MODE == 1)))) ? "Yes": "No" **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

4.15 AXI Secondary Read Data Signals

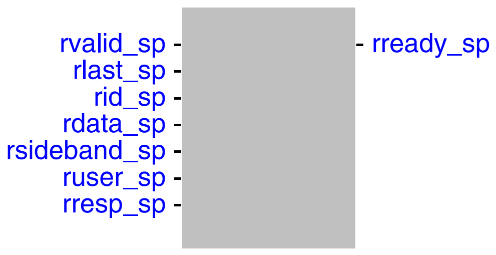


Table 4-15 AXI Secondary Read Data Signals

Port Name	I/O	Description
rready_sp	O	Secondary Port AXI Read Ready. Indicates that the slave can accept the read data. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** ((A2X_CLK_MODE == 0) && (((A2X_PP_MODE==0) || (A2X_PP_ENDIAN == 0)) && (A2X_SP_ENDIAN ==0) && (A2X_SP_DW == A2X_PP_DW) && (((A2X_LOCKED==0)? A2X_SP_BLW : A2X_PP_BLW)>(((A2X_PP_MODE==1)? A2X_PP_BLW : (A2X_LOCKED==0)? A2X_SP_BLW : A2X_PP_BLW)<A2X_HINCR_RBCNT_MAX)))) ? "Yes" :"No" **Power Domain:** SINGLE_DOMAIN **Active State:** High
rvalid_sp	I	Secondary Port AXI Read Valid. Indicates that valid read data is available. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High
rlast_sp	I	Secondary Port AXI Read Last. Indicates the last transfer in a read burst. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** High

Table 4-15 AXI Secondary Read Data Signals (Continued)

Port Name	I/O	Description
rid_sp[(A2X_SP_IDW-1):0]	I	Secondary Port AXI Read ID. Identification tag of the read data transfer. Must match the arid value of the read transaction. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
rdata_sp[(A2X_SP_DWIDTH-1):0]	I	Secondary Port AXI Read Data. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
rsideband_sp[(A2X_INT_RSBW-1):0]	I	Optional. Secondary Port Read Data Sideband Bus. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named rsideband_sp. ■ When the AXI4/ACELITE interface is enabled, this signal is named ruser_sp. **Exists:** (A2X_AXI_INTERFACE_TYPE == 0) && (A2X_RSBW != 0) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A
ruser_sp[(A2X_INT_RSBW-1):0]	I	Optional. Secondary Port Read Data User Signals. The signal name changes between the AXI3 and the AXI4/ACELITE configurations. ■ When the AXI3 interface is enabled, this signal is named rsideband_sp. ■ When the AXI4/ACELITE interface is enabled, this signal is named ruser_sp. **Exists:** (A2X_AXI_INTERFACE_TYPE != 0) && (A2X_RSBW != 0) **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

Table 4-15 AXI Secondary Read Data Signals (Continued)

Port Name	I/O	Description
rresp_sp[1:0]	I	Secondary Port AXI Read Response. Indicates the status of the read transfer. **Exists:** Always **Synchronous To:** (A2X_CLK_MODE == 0) ? ((A2X_PP_MODE == 0) ? "hclk" : "clk_pp") : "clk_sp" **Registered:** No **Power Domain:** SINGLE_DOMAIN **Active State:** N/A

5 Internal Parameter Descriptions

Provides a description of the internal parameters that might be indirectly referenced in expressions in the Signals, Parameters, or Registers chapters. These parameters are not visible in the coreConsultant GUI and most of them are derived automatically from visible parameters. **You must not set any of these parameters directly.**

Some expressions might refer to TCL functions or procedures (sometimes identified as **function_of**) that coreConsultant uses to make calculations. The exact formula used by these TCL functions is not provided in this chapter. However, when you configure the core in coreConsultant, all TCL functions and parameters are evaluated completely; and the resulting values are displayed where appropriate in the coreConsultant GUI reports.

Table 5-1 Internal Parameters

Parameter Name	Equals To
A2X_BLW	(A2X_PP_MODE ==1) ? A2X_PP_BLW : (A2X_LOCKED ==0) ? A2X_SP_BLW : A2X_PP_BLW
A2X_BLWIDTH	A2X_BLW
A2X_HASBW	0
A2X_HINCR_LEN_W	4
A2X_HINCR_RBCNT_MIN	(A2X_PP_DW<A2X_SP_DW) ? A2X_RS_RATIO_LOG2 : 0
A2X_HINCR_WBCNT_MIN	(A2X_PP_DW<A2X_SP_DW) ? A2X_RS_RATIO_LOG2 : 0
A2X_IDW	A2X_PP_IDW
A2X_INT_ARSBW	{(A2X_ARSBW ==0) ? 1 : A2X_ARSBW }
A2X_INT_AWSBW	{(A2X_AWSBW ==0) ? 1 : A2X_AWSBW }
A2X_INT_BARW	(A2X_INTERFACE_TYPE ! =2) ? 1 : 2
A2X_INT_BSBW	(A2X_BSBW ==0) ? 1 : A2X_BSBW
A2X_INT_DOMAINW	(A2X_INTERFACE_TYPE ! =2) ? 1 : 2

Table 5-1 Internal Parameters (Continued)

Parameter Name	Equals To
A2X_INTERFACE_TYPE	A2X_AXI_INTERFACE_TYPE
A2X_INT_HASBW	(A2X_HASBW ==0) ? 1 : A2X_HASBW
A2X_INT_LTW	{(A2X_AXI_INTERFACE_TYPE ==0) ? 2 : 1}
A2X_INT_QOSW	(A2X_QOS ==0) ? 1 : 4
A2X_INT_REGIONW	(A2X_REGION ==0) ? 1 : 4
A2X_INT_RSBW	(A2X_RSBW ==0) ? 1 : A2X_RSBW
A2X_INT_RSNOOPW	(A2X_INTERFACE_TYPE ! =2) ? 1 : 4
A2X_INT_WSBW	(A2X_WSBW ==0) ? 1 : A2X_WSBW
A2X_INT_WSNOOPW	(A2X_INTERFACE_TYPE ! =2) ? 1 : 3
A2X_LKMODE_MAX_PREFETCH	{[~~function_of:~~::DW_axi_a2x::calcRLK_MAXPREFETCH A2X_HINCR_RBCNT_MAX A2X_PP_MODE]}
A2X_OSAW_LIMIT	(A2X_SP_OSAW_LIMIT_P1-1)
A2X_OSR_EN	{((A2X_PP_MODE ==0) && (A2X_AHB_SPLIT_MODE ==0)) ? 0 : ((A2X_PP_MODE ==0) && (A2X_HINCR_RBCNT_MAX>A2X_BLW)) ? 1 : (A2X_PP_ENDIAN ==3) ? 1 : (A2X_SP_ENDIAN ==3) ? 1 : (A2X_PP_DW ==A2X_SP_DW) ? 0 : 1 }
A2X_OSW_EN	(A2X_BRESP_MODE ==0) ? 0 : ((A2X_SP_DW ==A2X_PP_DW) && (A2X_BRESP_MODE ==1)) ? 0 : 1
A2X_PP_AWIDTH	A2X_PP_AW
A2X_PP_DWIDTH	A2X_PP_DW
A2X_PP_NUM_BYTES	{ [~~function_of:~~::DW_axi_a2x::calcNumDiv8 A2X_PP_DW] }
A2X_PP_WSTRB_DW	A2X_PP_NUM_BYTES
A2X_QOS	A2X_INC_QOS
A2X_REGION	A2X_INC_REGION
A2X_RS_RATIO	{ [~~function_of:~~::DW_axi_a2x::calcRSRatio A2X_PP_DW A2X_SP_DW] }
A2X_RS_RATIO_LOG2	{ [~~function_of:~~::DW_axi_a2x::calcNumLog2 A2X_RS_RATIO] }
A2X_SNF_ARLEN_DFLT	A2X_BLW
A2X_SNF_AWLEN_DFLT	A2X_BLW

Table 5-1 Internal Parameters (Continued)

Parameter Name	Equals To
A2X_SP_AWIDTH	A2X_SP_AW
A2X_SP_BLWIDTH	A2X_SP_BLW
A2X_SP_DWIDTH	A2X_SP_DW
A2X_SP_NUM_BYTES	{ [~~function_of:~~::DW_axi_a2x::calcNumDiv8 A2X_SP_DW] }
A2X_SP_WSTRB_DW	A2X_SP_NUM_BYTES
A2X_UPSIZE	{(A2X_PP_DW<A2X_SP_DW) ? 1 : 0}

6 Integration Considerations

After you have configured, tested, and synthesized your component with the coreTools flow, you can integrate the component into your own design environment

6.1 Performance

This section discusses performance and the hardware configuration parameters, that affect the performance of the DW_axi_a2x.

6.1.1 Power Consumption, Frequency, Area, and DFT Coverage

Table 6-1 provides information about the synthesis results (power consumption, frequency and area) and DFT coverage of the DW_axi_a2x using the industry standard 16nm technology library.

Table 6-1 Synthesis Results for DW_axi_a2x

Configuration	Operating Frequency	Gate Count	Power Consumption		Tetramax (%)		SpyGlass StuckAtCov (%)
			Static Power	Dynamic Power	StuckAt Test	Transition	
Default Configuration	clk_pp = 400 MHz	21032	0.916 mW	20 nW	99.94	99.89	99.8
Typical Configuration 1 A2X_AXI_INTERFACE_TYPE = 1 A2X_CLK_MODE = 0 A2X_NUM_URID = 4 A2X_OSR_LIMIT_P1 = 16 A2X_PP_DW = 16 A2X_PP_MODE = 1 A2X_READ_ORDER = 1 A2X_SP_DW = 32 A2X_SP_OSAW_LIMIT_P1 = 16	clk_pp = 400 MHz	37224	1.49 mW	30 nW	99.97	99.88	99.6

Table 6-1 Synthesis Results for DW_axi_a2x

Configuration	Operating Frequency	Gate Count	Power Consumption		Tetramax (%)		SpyGlass StuckAtCov (%)
			Static Power	Dynamic Power	StuckAt Test	Transition	
Typical Configuration 2 A2X_AXI_INTERFACE_TYPE = 1 A2X_CLK_MODE = 2 A2X_NUM_URID = 4 A2X_OSR_LIMIT_P1 = 16 A2X_PP_DW = 32 A2X_PP_MODE = 1 A2X_WBUF_MODE = 1 A2X_RBUF_MODE = 1 A2X_READ_INTLEV = 1 A2X_READ_ORDER = 1 A2X_SP_DW = 16 A2X_BRESP_ORDER = 1 A2X_SP_OSAW_LIMIT_P1 = 16	clk_pp = 400 MHz clk_sp = 400MHz	40772	1.58mW	30 nW	99.95	99.7	99.5
Typical Configuration 3 A2X_AXI_INTERFACE_TYPE = 0 A2X_CLK_MODE = 2 A2X_NUM_URID = 4 A2X_OSR_LIMIT_P1 = 16 A2X_PP_DW = 16 A2X_PP_MODE = 0 A2X_READ_ORDER = 1 A2X_BRESP_MODE = 2 A2X_SP_DW = 32 A2X_SP_OSAW_LIMIT_P1 = 16	hclk = 300MHz clk_sp = 400 MHz	47848	1.4 mW	40 nW	99.96	99.43	99.8

Table 6-1 Synthesis Results for DW_axi_a2x

Configuration	Operating Frequency	Gate Count	Power Consumption		Tetramax (%)		SpyGlass StuckAtCov (%)
			Static Power	Dynamic Power	StuckAt Test	Transition	
Typical Configuration 4 A2X_AXI_INTERFACE_TYPE = 0 A2X_BRESP_MODE = 0 A2X_BRESP_ORDER = 1 A2X_CLK_MODE = 0 A2X_NUM_URID = 4 A2X_NUM_UWID = 4 A2X_OSR_LIMIT_P1 = 16 A2X_PP_DW = 32 A2X_PP_MODE = 0 A2X_READ_ORDER = 1 A2X_SP_DW = 16 A2X_SP_OSAW_LIMIT_P1 = 16	hclk = 300 MHz	45953	1.23 mW	40 nW	99.98	99.7	99.5

6.1.2 Latency Calculations

The latency from the DW_axi_a2x primary port channel to the secondary port channel is dependent on the type of configuration selected and the number of synchronization stages. Table 2-2 and Table 2-3 on page 71 gives details of when the write and read data buffers exist in the DW_axi_a2x. These buffers have a direct impact on the latency of the DW_axi_a2x.

Note The values "+1*_clk_sp_" and "+1*_clk_pp_" in the following equations are the effect of the status flag registering in the channel buffers.

The synchronization and pipeline latency parameters used in the equations below are calculated as follows:

- If the DW_axi_a2x is configured for synchronization mode, sp_sync_latency equals 0; otherwise sp_sync_latency equals (*A2X_SP_SYNC_DEPTH* **clk_sp* + 1**clk_sp*).
- If the DW_axi_a2x is configured for synchronization mode, pp_sync_latency equals 0; otherwise pp_sync_latency equals (*A2X_PP_SYNC_DEPTH* **clk_pp* + 1**clk_pp*).
- If the DW_axi_a2x AW secondary port channel is configured for pipelining, the aw_pipe_latency equals 1**clk_sp*; otherwise aw_pipe_latency equals 0.
- If the DW_axi_a2x Read Address secondary port channel is configured for pipelining, the ar_pipe_latency equals 1**clk_sp*; otherwise ar_pipe_latency equals 0.
- If the primary port buffers from Table 2-2 exist, the following latencies apply:
  - pp_osaw_latency equals 1**clk_pp*

  - pp_buf_latency equals 1**clk_pp*
- If the secondary port buffers from Table 2-2 exist, the following latencies apply:
  - sp_osaw_latency equals 1**clk_sp*
  - sp_osw_latency equals 1**clk_sp*
- If the secondary port buffers from Table 2-3 exist, the following latency applies:
  - sp_osr_latency equals 1**clk_sp*

The latency from respective Primary Port to Secondary Port address channel valid signals is aw_addr_ch_latency and ar_addr_ch_latency:

- aw_addr_ch_latency = 1**clk_pp* + *sp_sync_latency* + *aw_pipe_latency*
- ar_addr_ch_latency = 1**clk_pp* + *sp_sync_latency* + *ar_pipe_latency*

The latency from awvalid_pp to wvalid_sp—earliest time the DW_axi_a2x can forward a write beat issued in the same cycle as the corresponding command—is aw_to_w_latency:

- aw_to_w_latency = *aw_addr_ch_latency* - *aw_pipe_latency* + *sp_osaw_latency*

The latency from arvalid_m to rready_s-earliest time the DW_axi_a2x can accept a beat of read data on the secondary port after the DW_axi_a2x sends the read command is ar_to_r_latency:

- ar_to_r_latency = *ar_addr_ch_latency* - *ar_pipe_latency* + *sp_osr_latency*

The latency from wvalid_pp to wvalid_sp is w_to_w_latency:

- w_to_w_latency = 1**clk_pp* + *sp_sync_latency*

The latencies from rvalid_sp to rvalid_pp and bvalid_sp to bvalid_pp are:

- b_to_b_latency = 1**clk_s* + *pp_sync_latency*
- r_to_r_latency = 1**clk_s* + *pp_sync_latency*

7 Verification

This chapter provides an overview of the packaged verification environment contained within the installation of the DW_axi_a2x DesignWare IIP. This verification environment can be invoked directly from the coreConsultant/coreAssembler GUI to validate the user's configuration of the IIP.

Attention The verification environment is coded in SystemVerilog, and it therefore requires a SystemVerilog licensed simulator to compile and run.

7.1 Overview of Packaged Tests

A single basic testcase is packaged with the DW_axi_a2x IIP. This testcase is designed to sanity check the user's configuration of the IIP. The packaged tests are not designed to cover all functional features of the IIP or to hit 100% of the code coverage targets.

Packaged tests can be executed from the Setup and Run Simulations activity in the coreTools GUI. The output log files and wave dump files are generated in the *workspace*/sim/test_100_basic directory. The generated log files can also be directly viewed from the coreTool GUI.

7.1.1 test_100_basic

The test_100_basic testcase contains two separate Verilog tasks:

- Task to generate stimulus for configurations that have an AXI primary port
- Task to generate stimulus for configurations that have an AHB primary port

Each task generates 1000 AMBA transactions and applies them to the DW_axi_a2x bridge. Transaction attributes and data fields are randomized by the testcase. Checkers and protocol monitors in the verification environment trap any violations in the simulation and report to the simulation log file.

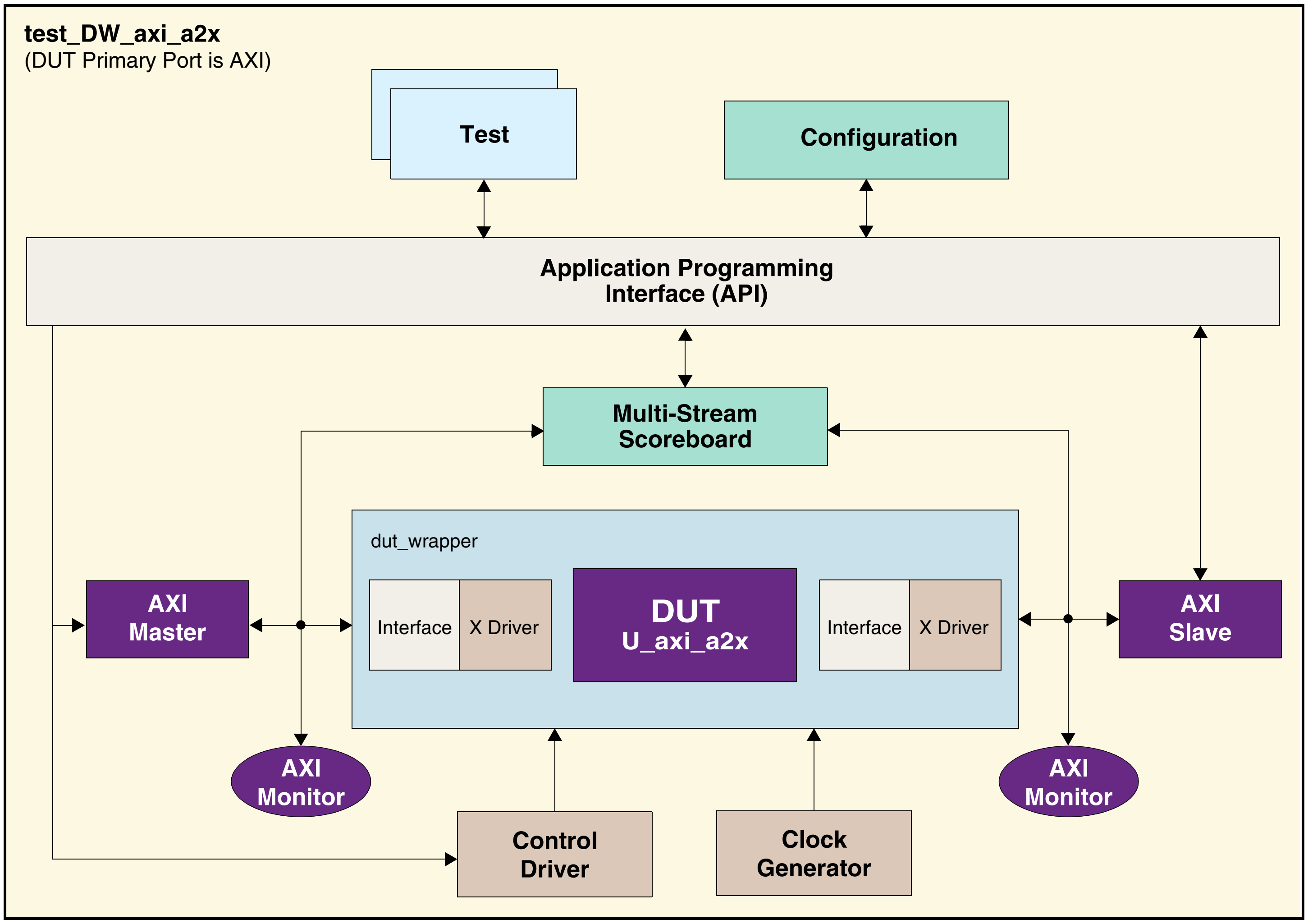
Users can use this testcase as a template for generating directed stimulus at the user site.

7.2 Overview of the Verification Environment

A single verification environment is used to verify all configurations of the DW_axi_a2x. The testbench files use Verilog `ifdef directives to match the configuration of the IIP under test. All testbench environment files—with the exception of DesignWare VIP files—are contained in the *workspace*/sim/testbench directory in plain-text format. The diagrams below show the architecture of the verification environment for both AHB and AXI primary port configurations.

Figure 7-1 illustrates the DW_axi_a2x architecture of the verification environment for the AXI primary port configuration.

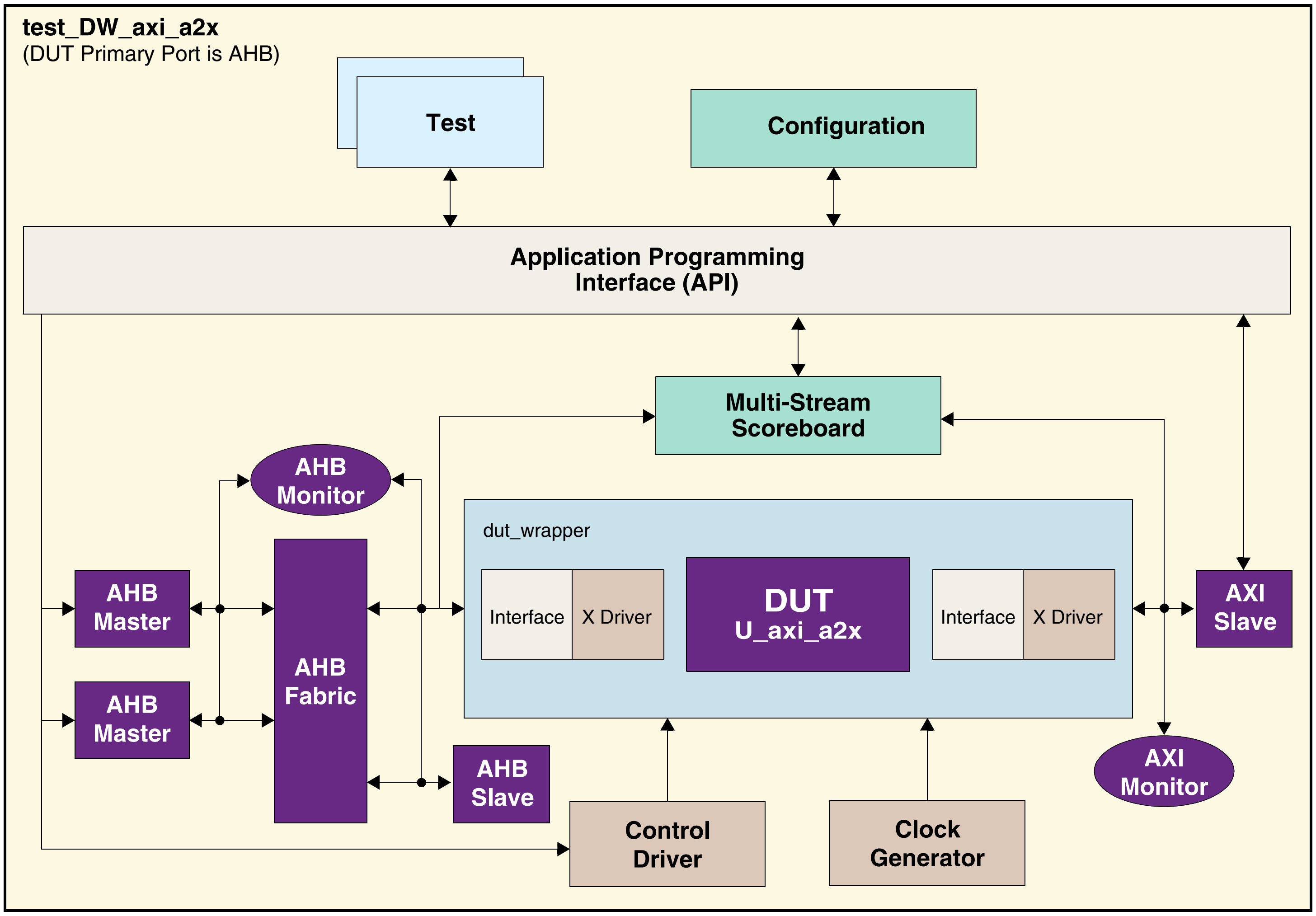
Figure 7-1 Verification Environment Architecture for AXI Primary Port Configurations



Note This is a single environment with `ifdef to control instantiation of primary port drivers.

Figure 7-2 illustrates the DW_axi_a2x architecture of the verification environment for the AHB primary port configuration.

Figure 7-2 Verification Environment Architecture for AHB Primary Port Configurations



Synopsys DesignWare AMBA VIP is used within the environment, illustrated by purple blocks in Figure 7-1 and Figure 7-2.

The AHB and AXI Master VIP components drive transactions—generated in the testcase into the primary port of the bridge. These models also consume the bridge transaction responses and perform some protocol checks.

The AHB and AXI Monitor VIP components ensure that compliance with the respective AMBA protocol is adhered to.

The Control Driver applies logic values—provided by the testcase—to the control interface on the DUT. The Control Driver is used for the following

- Controls resize options (arresize_pp and awresize_pp)
- Controls buffer mode (store and forward or cut-through)
- Controls AHB INCR burst length for read pre-fetch and write preload (hincr_rbcnt_mX, hincr_wbcnt_mX)

The Clock Generator creates synchronous or asynchronous clocks for the simulation—clock period controlled by compile time macros. Resets are driven into the DUT through testbench API tasks.

The configuration class contains DUT parameter values and values on the control interface.

The API supplies tasks to generate AMBA transactions and configure the VIP components.

The Multi-Stream Scoreboard checks DUT compliance against the databook specification. The scoreboard contains a single stream per transaction ID. As an example:

- If the bridge is configured to support eight AHB Masters, the AHB scoreboard contains eight streams.
- If the bridge is configured for a AXI primary port with A2X_PP_BLW = 4, the AXI scoreboard contains sixteen streams.

Each scoreboard stream directly monitors the primary and secondary port interfaces on the bridge. The scoreboard emulates the functionality of the DUT—as specified by the databook—to generate expected transaction information. The expected transactions are compared against the observed transaction to produce a pass/fail status.

The scoreboard also checks that the response of any given transaction matches the specification in the databook. Examples of some, but not all, response checking performed by the scoreboard are as follows:

- Ensure no SPLIT response is observed when A2X_AHB_SPLIT_MODE = 0.
- When A2X_AHB_SPLIT_MODE = 1, ensure all new read commands receive a SPLIT response.
- Ensure an unlocking transaction is generated on the secondary port when an AHB lock sequence completes.

Note If DW_axi_a2x is configured for AXI4/ACE-Lite mode, the testbench is automatically capable of supporting AXI4/ACE-Lite traffic using AMBA VMT VIPs.

A Basic Core Module (BCM) Library

The Basic Core Module (BCM) Library is a library of commonly used blocks for the Synopsys DesignWare IP development. These BCMs are configurable on an instance-by-instance basis and, for the majority of BCM designs, there is an equivalent (or nearly equivalent) DesignWare Building Block (DWBB) component.

This Appendix provides more information about the BCMs used in DW_axi_a2x.

- "BCM Library Components" on page 161
- "Synchronizer Methods" on page 162

A.1 BCM Library Components

Table A-1 describes the list of BCM library components used in DW_axi_a2x.

Table A-1 List of BCM Library Components used in the Design

BCM Module Name	BCM Description	DWBB Equivalent
DW_axi_a2x_bcm05.v	FIFO controller interface for one of two clock domains instantiated in DW_axi_a2x_bcm07.v	DW_fifoctl_if Submodule of DW_fifoctl_s2_sf
DW_axi_a2x_bcm06.v	Synchronous (Single Clock) FIFO Controller with Dynamic Flags	DW_fifoctl_s1_df
DW_axi_a2x_bcm07.v	Synchronous (Dual-Clock) FIFO Controller with Static Flags	DW_fifoctl_s2_sf
DW_axi_a2x_bcm21.v	Single Clock Data Bus Synchronizer	DW_sync
DW_axi_a2x_bcm57.v	Sync. Write-Port, Asynchronous. Read-Port RAM (Flip-Flop-Based)	DW_ram_r_w_s_dff
DW_axi_a2x_bcm58.v	Dual clock two port RAM with re-timing registers (Flip-Flop Based)	
DW_axi_a2x_bcm65.v	Synchronous (Single Clock) FIFO with Static Flags	DW_fifo_s1_sf
DW_axi_a2x_bcm66.v	Synchronous (Dual-Clock) FIFO with Static Flags	DW_fifo_s2_sf

A.2 Synchronizer Methods

This section also describes the synchronizer methods (blocks of synchronizer functionality) that are used in the DW_axi_a2x to synchronize cross clock clocking signals.

This section contains the following sections:

Note The DesignWare Building Blocks (DWBB) contains several synchronizer components with functionality similar to methods documented in this appendix. For more information about the DWBB synchronizer components go to:
https://www.synopsys.com/dw/buildingblock.php

A.2.1 Synchronizers used in DW_axi_a2x

Each of the synchronizers and synchronizer sub-modules are comprised of verified DesignWare Basic Core (BCM) RTL designs. The BCM synchronizer designs are identified by the synchronizer type. The corresponding RTL files comprising the BCM synchronizers used in the DW_axi_a2x are listed and cross referenced to the synchronizer type in Table A-2. Note that certain BCM modules are contained in other BCM modules, as they are used in a building block fashion.

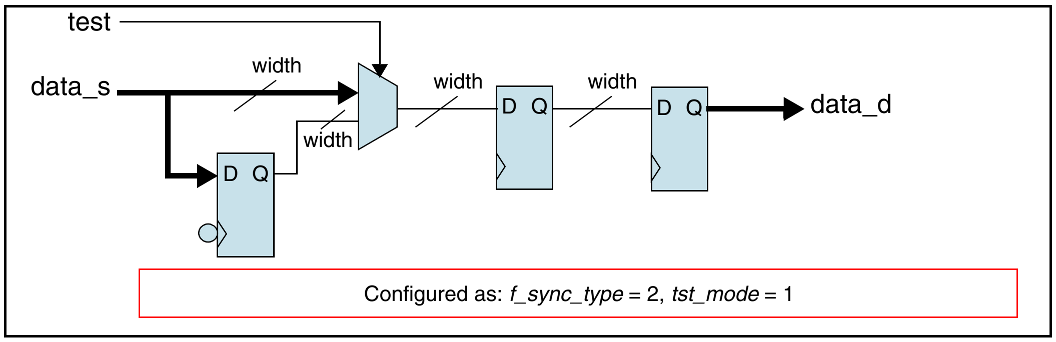
Table A-2 Synchronizers Used in DW_axi_a2x

Synchronizer module file	Sub module file	Synchronizer Type and Number
DW_axi_a2x_bcm21.v		Synchronizer 1: Simple Multiple register synchronizer
DW_axi_a2x_bcm66.v	DW_axi_a2x_bcm05.v DW_axi_a2x_bcm07.v DW_axi_a2x_bcm21.v DW_axi_a2x_bcm58.v	Synchronizer 2: Synchronous dual clock FIFO with Static Flags

A.2.2 Synchronizer 1: Simple Double Register Synchronizer (DW_axi_a2x)

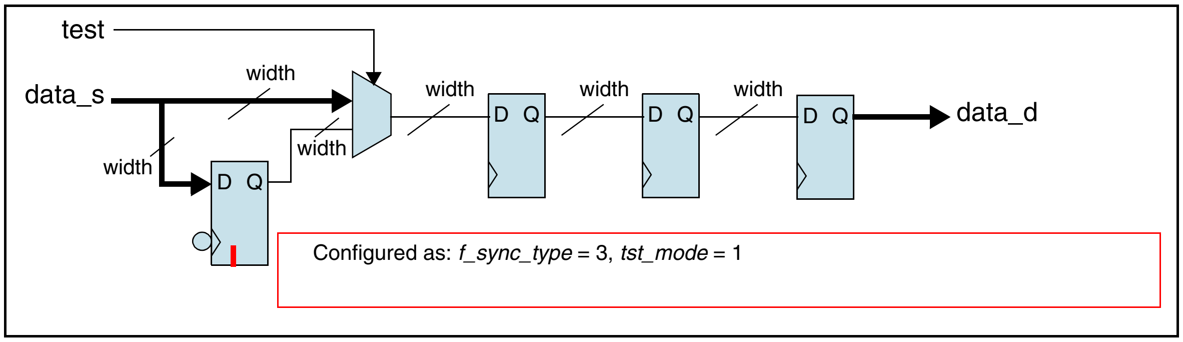
This is a single clock data bus synchronizer for synchronizing data that crosses asynchronous clock boundaries. The synchronization scheme depends on core configuration. If clk_pp/hclk and clk_sp are asynchronous (A2X_CLK_MODE =2) then DW_axi_a2x_bcm21 is instantiated inside the core for synchronization. The number of stages of synchronization is configurable through the parameters A2X_PP_SYNC_DEPTH and A2X_SP_SYNC_DEPTH. The following example shows the two stage synchronization process (Figure A-1) both using positive edge of clock.

Figure A-1 Block Diagram of Synchronizer 1 with two Stage Synchronization (both positive edge)



The following example shows the three stage synchronization process (Figure A-2) both using positive edge of clock.

Figure A-2 Block Diagram of Synchronizer 1 with Three Stage Synchronization (both positive edge)

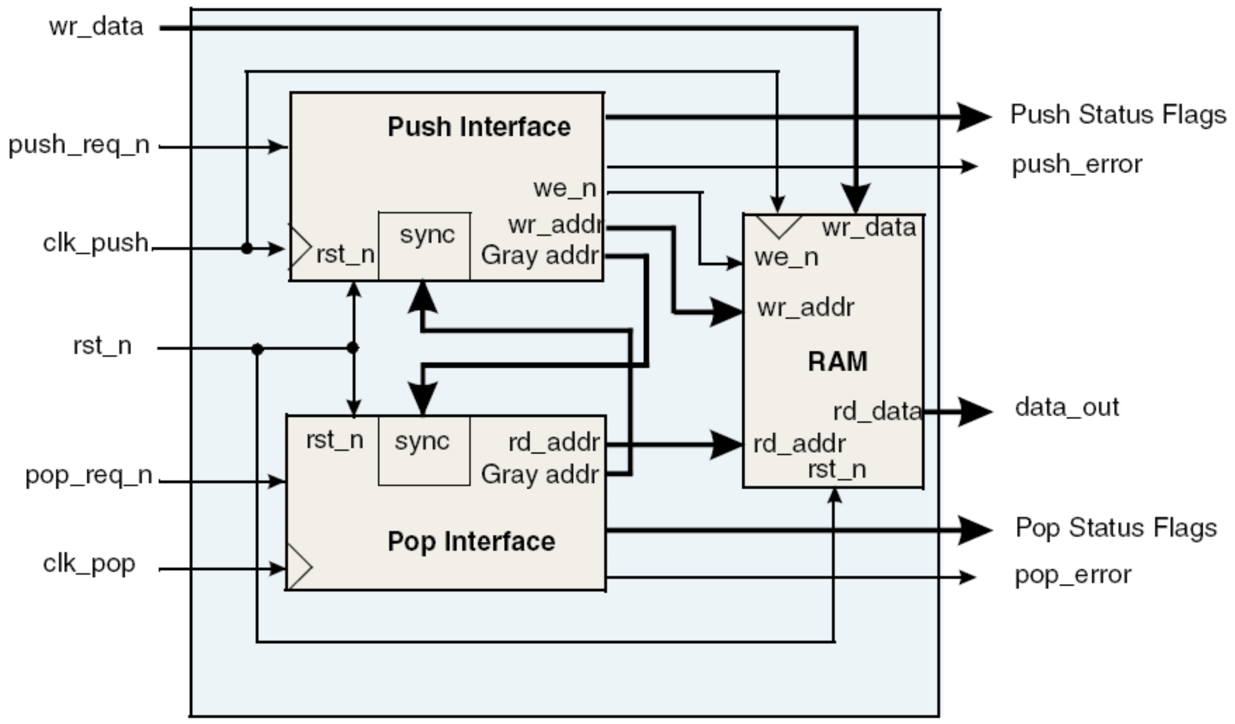


A.2.3 Synchronizer 2: Synchronous (Dual-clock) FIFO with static flags (DW_axi_a2x)

DW_axi_a2x_bcm66 is a dual independent clock FIFO. It combines the DW_axi_a2x_bcm07 FIFO controller and the DW_axi_a2x_bcm58 flip-flop based RAM DesignWare components.

The FIFO provides parameterized WIDTH and DEPTH, and a full complement of flags (full, almost full, half full, almost empty, empty, and error) for both of the clock domains. Figure A-3 shows the block diagram of Synchronizer 2.

Figure A-3 Synchronizer 2 Block diagram



B Glossary

application design	Overall chip-level design into which a subsystem or subsystems are integrated.
BFM	Bus-Functional Model — A simulation model used for early hardware debug. A BFM simulates the bus cycles of a device and models device pins, as well as certain on-chip functions. See also Full-Functional Model.
big-endian	Data format in which most significant byte comes first; normal order of bytes in a word.
blocked command stream	A command stream that is blocked due to a blocking command issued to that stream; see also command stream, blocking command, and non-blocking command.
blocking command	A command that prevents a testbench from advancing to next testbench statement until this command executes in model. Blocking commands typically return data to the testbench from the model.
command channel	Manages command streams. Models with multiple command channels execute command streams independently of each other to provide full-duplex mode function.
command stream	The communication channel between the testbench and the model.
component	A generic term that can refer to any synthesizable IP or verification IP in the DesignWare Library. In the context of synthesizable IP, this is a configurable block that can be instantiated as a single entity (VHDL) or module (Verilog) in a design.
configuration	The act of specifying parameters for a core prior to synthesis; can also be used in the context of VIP.
configuration intent	Range of values allowed for each parameter associated with a reusable core.
cycle command	A command that executes and causes HDL simulation time to advance.
decoder	Software or hardware subsystem that translates from and “encoded” format back to standard format.

design context	Aspects of a component or subsystem target environment that affect the synthesis of the component or subsystem.
design creation	The process of capturing a design as parameterized RTL.
DesignWare Library	A collection of synthesizable IP and verification IP components that is authorized by a single DesignWare license. Products include SmartModels, VMT model suites, DesignWare Memory Models, Building Block IP, and the DesignWareSynthesizable Components for AMBA.
dual role device	Device having the capabilities of function and host (limited).
endian	Ordering of bytes in a multi-byte word; see also little-endian and big-endian.
Full-Functional Mode	A simulation model that describes the complete range of device behavior, including code execution. See also BFM.
GPIO	General Purpose Input Output.
GTECH	A generic technology view used for RTL simulation of encrypted source code by non-Synopsys simulators.
hard IP	Non-synthesizable implementation IP.
HDL	Hardware Description Language – examples include Verilog and VHDL.
IIP	Implementation Intellectual Property — A generic term for synthesizable HDL and non-synthesizable "hard" IP in all of its forms (coreKit, component, core, MacroCell, and so on).
implementation view	The RTL for a core. You can simulate, synthesize, and implement this view of a core in a real chip.
instantiate	The act of placing a core or model into a design.
interface	Set of ports and parameters that defines a connection point to a component.
IP	Intellectual property — A term that encompasses simulation models and synthesizable blocks of HDL code.
little-endian	Data format in which the least-significant byte comes first.
master	Device or model that initiates and controls another device or peripheral.
model	A Verification IP component or a Design View of a core.
monitor	A device or model that gathers performance statistics of a system.
non-blocking command	A testbench command that advances to the next testbench statement without waiting for the command to complete.
peripheral	Generally refers to a small core that has a bus connection, specifically an APB interface.

RTL	Register Transfer Level. A higher level of abstraction that implies a certain gate-level structure. Synthesis of RTL code yields a gate-level design.
SDRAM	Synchronous Dynamic Random Access Memory; high-speed DRAM adds a separate clock signal to control signals.
SDRAM controller	A memory controller with specific connections for SDRAMs.
slave	Device or model that is controlled by and responds to a master.
SoC	System on a chip.
soft IP	Any implementation IP that is configurable. Generally referred to as synthesizable IP.
sparse data	Data bus data which is individually byte-enabled. The AXI bus allows sparse data transfers using the WSTRB signal, each byte lane individually enabled. The AHB bus does not allow sparse data transfers.
static controller	Memory controller with specific connections for Static memories such as asynchronous SRAMs, Flash memory, and ROMs.
synthesis intent	Attributes that a core developer applies to a top-level design, ports, and core.
synthesizable IP	A type of Implementation IP that can be mapped to a target technology through synthesis. Sometimes referred to as Soft IP.
technology-independent	Design that allows the technology (that is, the library that implements the gate and via widths for gates) to be specified later during synthesis.
Testsuite Regression Environment (TRE)	A collection of files for stand-alone verification of the configured component. The files, tests, and functionality vary from component to component.
VIP	Verification Intellectual Property — A generic term for a simulation model in any form, including a Design View.
wrap, wrapper	Code, usually VHDL or Verilog, that surrounds a design or model, allowing easier interfacing. Usually requires an extra, sometimes automated, step to create the wrapper.
zero-cycle command	A command that executes without HDL simulation time advancing.

Index