LogiCORE IP AXI to AHB-Lite Bridge v2.1

Product Guide for Vivado Design Suite

PG177 December 18, 2013

XILINX®

XILINX

Table of Contents

Introduction

The ARM® AMBA® AXI4 to Advanced High Performance Bus (AHB-Lite) Bridge translates AXI4 transactions into AHB-Lite transactions. It functions as a slave on the AXI4 interface and as a master on the AHB-Lite interface. The AXI4 to AHB-Lite Bridge main use model is to connect the AHB-Lite slaves with AXI4 masters.

Features

AXI4 Slave Interface:

- AXI4 interface is based on the AXI4 specification
- Connects as a 32/64-bit slave on 32/64-bit AXI4
- Supports 1:1 (AXI4:AHB) synchronous clock ratio
- Incrementing burst transfers (length 1 to 256)
- Wrapping burst transfers (length 2, 4, 8, and 16)
- Fixed burst transfers (of length 1 to 16)
- Narrow transfers
- Limited cache encoding and limited protection unit support
- Address/data phase timeout

AHB-Lite Master Interface:

- Connects as a 32/64-bit master on 32/64-bit AHB-Lite interface
- Supports single burst transfers
- Wrapping burst transfers (length 4, 8, and 16)
- Incrementing burst transfers (length 4, 8, 16, and undefined burst length
- Limited protection control
- Narrow transfers
- AHB-Lite master does not issue incrementing burst transfers that cross 1 KB address boundaries

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale™ Architecture, Zynq®-7000, 7 Series
Supported User Interfaces	AXI4, AHB-Lite
Resources	See Table 2-2.
Provided with Core	
Design Files	VHDL
Example Design	VHDL
Test Bench	VHDL
Constraints File	N/A
Simulation Model	Not Provided
Supported S/W Driver	N/A
Tested Design Flows(2)	
Design Entry	Vivado® Design Suite IP Integrator
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For a complete list of supported devices, see Vivado IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The AXI4 to AHB-Lite Bridge translates AXI4 transactions into AHB-Lite transactions. The bridge functions as a slave on the AXI4 interface and as a master on the AHB-Lite interface. AXI4 to AHB-Lite Bridge block diagram is shown in Figure 1-1 and described in following sections.

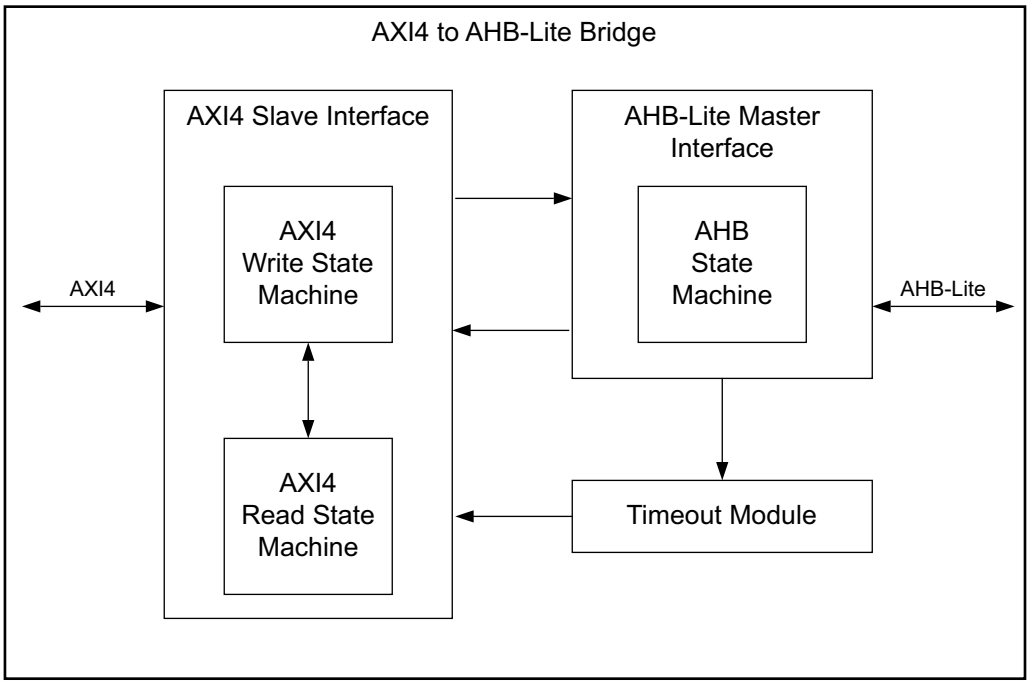

Figure 1-1: **AXI4 to AHB-Lite Bridge Block Diagram**

AXI4 Slave Interface

The AXI4 slave interface module provides a bidirectional slave interface. The AXI4 address width is fixed at 32 bits. The AXI4 data bus width can be either 32 or 64. The AXI4 to AHB-Lite Bridge supports the same data width on both the AXI4 and AHB-Lite interfaces.

When both write and read transfers are simultaneously requested on the AXI4 interface, the read request is given a higher priority than the write request.

AXI4 Write State Machine

The AXI4 write state machine is part of the AXI4 slave interface module and functions on AXI4 write channels. This module controls AXI4 write accesses and generates the write response. If a bridge timeout occurs, this module completes the AXI4 write transaction with a SLVERR response.

AXI4 Read State Machine

The AXI4 read state machine is part of the AXI4 slave interface module and functions on AXI4 read channels. This module controls the AXI4 read accesses and generates the read response. If a bridge timeout occurs, this module completes the AXI4 read transaction with a SLVERR response.

AHB-Lite Master Interface

The AHB-Lite master interface module provides the AHB-Lite master interface. The AHB-Lite address width is fixed at 32 bits and the data bus width can be either 32 or 64 bits.

AHB State Machine

The AHB state machine is part of the AHB-Lite master interface module.

When the AXI4 interface initiates a write access, the AHB state machine module receives the control signals and data from AXI4 slave interface, then transfers the same to the equivalent AHB-Lite write access. This module also transfers the AHB-Lite write response to the AXI4 slave interface.

When the AXI4 interface initiates a read access, the AHB state machine module receives the control signals from AXI4 slave interface, then transfers the same to the equivalent AHB-Lite read access. This module also transfers AHB-Lite read data and read response to the AXI4 slave interface.

Timeout Module

The timeout module generates the timeout when the AHB-Lite slave is not responding to the AHB transaction. This is parameterized and generates the timeout only when its value is non-zero.

The timeout module waits for number of AXI4 clocks indicated by parameterized timeout value for the AHB-Lite slave response and generates the timeout if the AHB slave does not respond.

Unsupported Features

AXI4 Slave Interface

- Data bus widths greater than 64
- No registers are implemented because posted writes
- Locked, Barrier, trust zone, and exclusive operations
- Out-of-order read transaction completion
- Out-of-order write transaction completion
- Unaligned/Sparse burst transfers (holes in strobes)
- EXOKAY and DECERR responses to AXI4
- Low-power state
- Secure accesses

AHB-Lite Master interface

- Data bus widths greater than 64
- Cacheable access

Licensing and Ordering Information

This Xilinx LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License.

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Standards

AXI4 interface is based on the AXI4 specification and AHB-Lite interface based on the AHB specification.

Performance

Performance characterization of this core has been done using margin system methodology. The details of the margin system characterization methodology is described in "Vivado IP Optimization (F_{Max} Characterization) appendix," in the *Vivado Design Suite User Guide: Designing With IP* (UG896) [Ref 2].

Note: Maximum frequency numbers for Zynq®-7000 devices are expected to be similar to 7 series device numbers.

Table 2-1: **Maximum Frequencies**

Family	Speed Grade	F_{Max} (MHz)
		AXI4
Virtex-7	–1	200
Kintex-7		200
Artix-7		150
Virtex-7	–2	240
Kintex-7		240
Artix-7		180
Virtex-7	–3	280
Kintex-7		280
Artix-7		200

Latency

The best possible core configuration has been selected for calculating the read and write latency for the increment, wrapping, and fixed type of burst transfers. The read latency from read address valid (`s_axi_arvalid`) to the data beat (`s_axi_rvalid`) of AXI4 to AHB-Lite Bridge is four clock cycles.

The write latency from write address valid (`s_axi_awvalid`) to the availability of valid data on AHB data bus is three clock cycles.

Resource Utilization

Note: Resource utilization numbers for UltraScale™ architecture and Zynq-7000 devices are expected to be similar to 7 series device numbers.

7 Series FPGAs

Table 2-2 provides approximate resource counts for the various core options using 7 series FPGAs.

Table 2-2: **Device Utilization – 7 Series FPGAs**

Parameter Values				Device Resources		
Data Width	**ID Width**	**Timeout**	**Supports Narrow Burst Transfers**	**Slices**	**Slice Flip-Flops**	**LUTs**
32	4	0	0	182	427	287
32	4	0	1	179	428	295
64	4	0	0	216	584	338
64	16	0	1	239	662	356
64	16	0	0	247	656	346
64	16	128	0	226	667	336
64	4	256	1	219	598	344

Port Descriptions

Table 2-3 shows the I/O signals of the AXI4 to AHB-Lite Bridge.

Table 2-3: **I/O Signal Description**

Signal Name	Interface	I/O	Initial State	Description
AXI4 Interface System Signals				
s_axi_aclk	System	I	–	AXI4 clock.
s_axi_aresetn	System	I	–	AXI4 reset, active-Low.
AXI4 Write Address Channel Signals				
s_axi_awid	AXI4	I	–	Write address ID. This signal is the identification tag for the write address group of signals. Width of the port is your ID width. This port does not generate if ID width is zero.
s_axi_awlen[7:0]	AXI4	I	–	Burst length. The burst length gives the exact number of transfers in a burst.
s_axi_awsize[2:0]	AXI4	I	–	Burst size. Indicates the size of each transfer in the burst.
s_axi_awburst[1:0]	AXI4	I	–	Burst type. The burst type coupled with the size information details how the address for each transfer within the burst is calculated.
s_axi_awcache[3:0]	AXI4	I	–	Cache type. Indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction.
s_axi_awlock	AXI4	I	–	Lock type. This signal provides additional information about the atomic characteristics of the transfer.
s_axi_awaddr[31:0]	AXI4	I	–	AXI4 Write address. The write address bus gives the address of the first transfer in a write burst transaction.
s_axi_awprot[2:0]	AXI4	I	–	Protection type. Indicates the normal, privileged, or secure protection level of the write transaction and whether the transaction is a data access or an instruction access. The default value is a normal non-secure data access.
s_axi_awvalid	AXI4	I	–	Write address valid. Indicates that valid write address and control information are available.
s_axi_awready	AXI4	O	0	Write address ready. Indicates that the slave is ready to accept an address and associated control signals.
AXI4 Write Data Channel Signals				
s_axi_wdata	AXI4	I	–	Write data bus. 32 or 64 data width is supported by the AHB-Lite Bridge.

Table 2-3: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	I/O	Initial State	Description
s_axi_wstrb	AXI4	I	–	Write strobes. Indicates which byte lanes to update in memory. The 4/8-bit width port is created when 32/64-bit data width is selected respectively.
s_axi_wvalid	AXI4	I	–	Write valid. Indicates that valid write data and strobes are available.
s_axi_wlast	AXI4	I	–	Write last. Indicates the last transfer in a write burst.
s_axi_wready	AXI4	O	0	Write ready. Indicates that the slave can accept the write data.
AXI4 Write Response Channel Signals				
s_axi_bid	AXI4	O	0	Response ID. The identification tag of the write response. Width of the port is your ID width. This port does not generate if ID width is zero.
s_axi_bresp[1:0]	AXI4	O	0	Write response. Indicates the status of the write transaction.
s_axi_bvalid	AXI4	O	0	Write response valid. Indicates that a valid write response is available.
s_axi_bready	AXI4	I	–	Response ready. Indicates that the master can accept the response information.
AXI4 Read Address Channel Signals				
s_axi_arid	AXI4	I	–	Read address ID. This signal is the identification tag for the read address group of signals. Width of the port is your ID width. This port does not generate if ID width is zero.
s_axi_araddr[31:0]	AXI4	I	–	Read address. The read address bus gives the initial address of a read burst transaction.
s_axi_arcache[3:0]	AXI4	I	–	Cache type. This signal provides additional information about the cacheable characteristics of the transfer.
s_axi_arprot[2:0]	AXI4	I	–	Protection type. This signal provides protection unit information for the read transaction. The default value is normal non secure data access.
s_axi_arvalid	AXI4	I	–	Read address valid. Indicates when High that the read address and control information is valid and remains stable until the address acknowledgement signal `s_axi_arready` is High.
s_axi_arlen[7:0]	AXI4	I	–	Burst length. The burst length gives the exact number of transfers in a burst.
s_axi_arsize[2:0]	AXI4	I	–	Burst size. Indicates the size of each transfer in the burst.
s_axi_arburst[1:0]	AXI4	I	–	Burst type. The burst type coupled with the size information details how the address for each transfer within the burst is calculated.

Table 2-3: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	I/O	Initial State	Description
s_axi_arlock	AXI4	I	–	Lock type. This signal provides additional information about the atomic characteristics of the transfer.
s_axi_arready	AXI4	O	0	Read address ready. Indicates that the slave is ready to accept an address and associated control signals.
AXI4 Read Data Channel Signals				
s_axi_rid	AXI4	O	0	Read ID tag. This signal is the identification tag for the read data group of signals. Width of the port is your ID width. This port does not generate if ID width is zero.
s_axi_rdata	AXI4	O	0	Read data bus. Width can be 32/64-bit.
s_axi_rlast	AXI4	O	0	Read last. Indicates the last transfer in a read burst.
s_axi_rresp[1:0]	AXI4	O	0	Read response. Indicates the status of the read transfer.
s_axi_rvalid	AXI4	O	0	Read valid. Indicates that the required read data is available and the read transfer can complete.
s_axi_rready	AXI4	I	–	Read ready. Indicates that the master can accept the read data and response information.
AHB-Lite Signals				
m_ahb_hclk	AHB-Lite	O	–	AHB Clock. `s_axi_aclk` is tied to `m_ahb_hclk`.
m_ahb_hresetn	AHB-Lite	O	–	AHB Reset, active-Low. s_axi_aresetn is tied to `m_ahb_hresetn`.
m_ahb_haddr[31:0]	AHB-Lite	O	0	This is the AHB address bus and is fixed to 32-bit.
m_ahb_hprot[3:0]	AHB-Lite	O	0011(1)	Protection type. Indicates the normal, privileged transaction and whether the transaction is a data access or an instruction access.
m_ahb_htrans[1:0]	AHB-Lite	O	0	AHB Transfer Type, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE, or BUSY.
m_ahb_hsize[2:0]	AHB-Lite	O	0	AHB Size of Transfer.
m_ahb_hwrite	AHB-Lite	O	0	Indicates the transfer direction, this signal indicates an AHB write access when High and an AHB read access when Low.
m_ahb_hwdata	AHB-Lite	O	0	Write data. The write data bus transfers data from the master to the slaves during write operations. Width of the port can be 32/64-bit.
m_ahb_hburst[2:0]	AHB-Lite	O	0	AHB Burst type. The burst type indicates if the transfer is a single transfer or forms part of a burst.The burst can be incrementing or wrapping.
m_ahb_hmastlock	AHB-Lite	O	0	Indicates that the current master is performing a locked sequence of transfers
m_ahb_hready	AHB-Lite	I	–	Ready. The AHB slave uses this signal to extend an AHB transfer.

Table 2-3: **I/O Signal Description *(Cont'd)***

Signal Name	Interface	I/O	Initial State	Description
m_ahb_hrdata	AHB-Lite	I	–	AHB read data driven by slave. Width of the port can be 32/64-bit.
m_ahb_hresp	AHB-Lite	I	–	Transfer Response. When Low, indicates that the transfer status is OKAY. When High, indicates that the transfer status is ERROR.

1. The default value 0011 is driven on m_ahb_hprot[3:0] as per the ARM® recommendation, this corresponds to non-cacheable, non-bufferable, privileged, data access.

Register Space

There are no registers in the AXI4 to AHB-Lite Bridge.

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

Clocking

The AXI4 to AHB-Lite Bridge is a synchronous design which uses the `s_axi_aclk` at both the AXI4 and AHB-Lite interfaces. `m_ahb_hclk` is driven by the AXI4 to AHB-Lite Bridge (tied to `s_axi_aclk`).

Resets

`s_axi_aresetn` is a synchronous reset input that resets the AXI4 to AHB-Lite Bridge when asserted. `s_axi_aresetn` is also used to reset the AHB-Lite interface. `m_ahb_hresetn` is driven by the AXI4 to AHB-Lite Bridge (tied to `s_axi_aresetn`).

Memory Mapping

The AXI4 memory map and the AHB-Lite memory map are a single, complete 32-bit (4 GB) memory space. The AXI4 to AHB-Lite Bridge does not modify the address for AHB-Lite. Therefore, the address that is presented on the AHB-Lite is exactly as received on the AXI4 interface.

Data Width

The AXI4 to AHB-Lite Bridge supports the same data widths on both AXI4 and AHB-Lite interfaces. The 32 and 64-bit data widths are supported. Data width is the same on both AXI4 and AHB-Lite interfaces.

Narrow Transfers

Transactions where the transfer size is narrower than the data bus width are treated as narrow transfers. Narrow transfer support is parameterized in the AXI4 to AHB-Lite Bridge. Narrow transfers on the AXI4 interface are supported and can be transferred to and from AHB-Lite interface when the core is generated with parameter to support narrow burst transfers is set toTRUE.

The 8/16-bit or 8/16/32-bit narrow burst transfers are allowed through the bridge when data width is 32 or 64 respectively. The AXI4 to AHB-Lite Bridge monitors the AXI4 strobe, `s_axi_wstrb`, and drives `m_ahb_hsize` when there are single transfers initiated by the AXI4 master.

The AXI4 strobe, `s_axi_wstrb`, is ignored in the AXI4 to AHB-Lite Bridge for burst transfers. Also, for burst transfers, `m_ahb_hsize` is determined based on `s_axi_awaddr`/`s_axi_araddr` and `s_axi_awsize`/`s_axi_arsize`. Sparse/unaligned transfers are not supported in burst transfers.

Endianness

Both the AXI4 and AHB-Lite are little endian.

Address/Data Translation

No address/data translation/conversion from AXI4 to AHB-Lite takes place inside the AXI4 to AHB-Lite Bridge. The write/read address from AXI4 is passed to AHB-Lite address. AXI4 write data is passed on to AHB-Lite and AHB-Lite read data is passed on to AXI4 read data.

Read and Write Ordering

When a read and a write request are issued simultaneously (`s_axi_awvalid`/`s_axi_wvalid` and `s_axi_arvalid` are asserted High) from the AXI4 interface, the AXI4 to AHB-Lite Bridge gives a higher priority to the read request. When both write and read requests are valid, the write request is initiated on AHB-Lite after the read is requested on AHB-Lite interface.

1 KB Burst Crossing

The AXI4 to AHB-Lite Bridge implements the logic that checks if the AXI4 transaction (both read and write) crosses the 1 KB boundary, then splits the transaction to prevent it from crossing boundaries. Any burst initiated on the AXI4 interface that crosses a 1 KB address boundary is converted to two INCR bursts of undefined length with the restarted burst on AHB-Lite.

Bridge Timeout Condition

Timeout logic is parameterized in the AXI4 to AHB-Lite Bridge. A timeout is generated when the timeout parameter value is 16, 32, 64, 128, or 256. When a request is issued from the AXI4 interface, the bridge translates this request into the corresponding AHB-Lite transfer and requests it on the AHB-Lite interface. If this request is not responded to by the AHB-Lite interface, the timeout logic waits for a timeout period and responds with SLVERR on the AXI4 interface. The bridge sends an IDLE transfer to AHB-Lite interface.

If the parameter value is "0," it is assumed that the AHB slave always responds when a transfer is requested and that no timeout logic exists that automatically responds on the AXI4 interface. When the AHB slave does not respond, the AXI4 to AHB-Lite Bridge waits indefinitely for the AHB-Lite slave response.

Note: Select the value for the timeout parameter to avoid the AXI4 to AHB-Lite Bridge waiting indefinitely for the AHB-Lite slave response.

AXI4 Response Signaling

Only OKAY and SLVERR responses are generated on the AXI4 side. EXOKAY and DECERR are never used. An ERROR on the AHB-Lite interface or a timeout error because of a bridge timeout results in SLVERR.

No registers are implemented in AXI4 to AHB-Lite Bridge to differentiate the AHB-Lite ERROR and bridge timeout error. It is assumed and recommended that the AXI4 master resets the bridge with SLVERR response.

Protection and Cache Support

The protection and cache support is limited in the AXI4 to AHB-Lite Bridge. The protection and cache support in AXI4 is mapped to the available equivalent AHB-Lite protection as listed in the Table 3-1. In AXI4, no cacheable transaction exists in the AHB-Lite. There are no modifiable, write allocate, read allocate, secure and non secure accesses in the AHB-Lite interface that exist in AXI4 interface.

The protection support privileged, instruction, normal, and data accesses in AXI4 are mapped to the equivalent AHB-Lite protection. The `s_axi_awprot[2:0]` and `s_axi_arprot[2:0]` signals carry the protection unit support on the AXI4 interface.

The AXI4 bufferable/non-bufferable transaction attributes are mapped to the equivalent AHB-Lite protection signals. The `s_axi_awcache[3:0]` and `s_axi_arcache[3:0]` carry the cache support attributes of AXI4.

Table 3-1: **AXI4 Protection And Cache Support Translation to AHB-Lite Protection**

AXI4 Protection and Cache Support		AHB-Lite Protection Support	Description
s_axi_awcache [3:0]/ s_axi_ arcache[3:0]	**s_axi_awprot[2:0]/ s_axi_arprot[2:0]**	**m_ahb_hprot[3:0]**	
xxxx	xx1	0x1x	Privileged on AXI4 is translated to the equivalent privileged in AHB-Lite
xxxx	xx0	0x0x	Normal access on AXI4 is mapped to the equivalent user access on AHB
xxxx	0xx	0xx1	Data access on AXI4 is translated to the equivalent Data access in AHB-Lite
xxxx	1xx	0xx0	Instruction access on AXI4 is mapped to the equivalent Opcode fetch on AHB
00x1	xxx	01xx	Bufferable on AXI4 is translated to the equivalent Bufferable in AHB-Lite
00x0	xxx	00xx	Non-bufferable on AXI4 is translated to the equivalent Non-bufferable in AHB-Lite
xxxx	xxx	0xxx	There is no cacheable in AXI4. Always transferring Non-cacheable on AHB-Lite

As per the ARM recommendation [Ref 1] the default value on AHB-Lite protection control signal `m_ahb_hprot[3:0]` is 0011 to correspond to a non-cacheable, non-bufferable, privileged data access.

Bridge Transaction Translation

Table 3-2 shows translation of AXI4 transactions to AHB-Lite transactions. The supported AXI4 transactions are translated to AHB-Lite transactions.

- Incrementing burst of length 1 on AXI4 is converted to the equivalent AHB single transaction.
- Incrementing burst transfers of length 4, 8, and 16 on AXI4 are converted to the equivalent AHB-Lite incrementing bursts when there is no 1 KB cross in the access.
- Any other incrementing burst of length up to 256 (2 to 256, other than 4, 8, and 16) is converted to the AHB-Lite incrementing burst transfer of undefined length.
- Wrapping burst transfers of length 4, 8, and 16 are converted to the equivalent AHB-Lite wrapping burst.

For AXI4 transactions given below, multiple AHB-Lite transactions must be performed.

- For one AXI4 transaction, two AHB-Lite transactions must be requested when a 1 KB cross is detected in an AXI4 transfer. If there is 1 KB cross access with incrementing burst transfers of length 4, 8, and 16, then the corresponding transaction on AXI4 is converted to the two AHB-Lite incrementing burst transfers of undefined length.
- AXI4 allows WRAP type burst transactions of length 2, 4, 8, and 16; however, the AHB-Lite interface only supports WRAP 4, 8, and 16 transactions. WRAP transfer of length 2 on the AXI4 interface is converted to two AHB-Lite single burst transactions.
- Fixed burst is supported in AXI4 whereas there is no fixed burst transaction in AHB-Lite. Fixed burst on AXI4 is converted to AHB-Lite single transactions with fixed address.

Table 3-2: **AXI4 Transaction to AHB-Lite Transaction**

AXI4 Transaction	AHB-Lite Transaction	Description
Incrementing burst transaction of length 1	Single transaction	AXI4 incrementing burst transaction with length 1 is single transaction on AHB
Incrementing burst transfers of length 4, 8, and 16 without 1 KB crossing	Incrementing bursts 4, 8, and 16	If the access initiated by AXI4 does not cross the 1 KB, INCR4, INCR8, and INCR16 are same in both AXI4 and AHB-Lite

Table 3-2: **AXI4 Transaction to AHB-Lite Transaction *(Cont'd)***

AXI4 Transaction	AHB-Lite Transaction	Description
Incrementing burst transfers of length 4, 8, and 16 with 1 KB crossing	Incrementing burst transfers of undefined length	If the access initiated by AXI4 crosses the 1 KB, the transfer is split so the transfers on AHB-Lite are two increment bursts of undefined length transfers.
Any other incrementing burst up to 256 (Incrementing bursts of length from 2 to 256 other than 4, 8, and 16)	Incrementing burst transfers of undefined length	If the access is incrementing burst of length 2 to 256 other than 4, 8, and 16, the transfer on AHB-Lite is incrementing burst of undefined length transfer.
WRAP type burst transactions of length 4, 8, and 16	WRAP type burst transactions of 4, 8, and 16	WRAP4, WRAP8, and WRAP16 are the same in both AXI4 and AHB
WRAP type burst transaction of length 2	Two single transactions	There is no WRAP2 burst transfer in AHB-Lite. The transaction is split as two, single AHB-Lite transactions
Fixed burst transactions of length 1 to 16	Single transactions with the same address	There is no fixed burst transaction in AHB-Lite. Fixed transaction on AXI4 is single transactions on AHB-Lite to the same address location. The number of single transactions (for example, 1 to 16) depends on the AXI4 burst length.

Timing Diagrams

The AXI4 to AHB-Lite Bridge operation for various read and write transfers is shown in the subsequent timing diagrams.

1. AXI4 incrementing burst read transfer of length 1 is shown in Figure 3-1.
2. AXI4 incrementing burst write transfer of length 1 is shown in Figure 3-2.
3. AXI4 read and write transfers is shown in Figure 3-3. As shown, when read and write transfers are initiated on the AXI4 interface at the same time, the read transfer is given a higher priority than the write transfer.
4. AXI4 incrementing burst write transfer of length 4 is shown in Figure 3-4.
5. AXI4 incrementing burst read transfer of length 4 is shown in Figure 3-5.
6. AXI4 fixed burst read and write transfers are shown in Figure 3-6. AXI4 fixed burst read and write transfers of length 5 are transferred to five AHB-Lite single read and write transfers.
7. AXI4 wrapping burst read and write transfers of length 2 are shown in Figure 3-7. One AXI4 wrapping burst read/write transfer of length 2 is split into two AHB-Lite single read/write transfers.
8. AXI4 wrapping burst read transfer of length 4 is shown in Figure 3-8.
9. AXI4 wrapping burst write transfer of length 4 is shown in Figure 3-9.
10. AXI4 8-bit narrow read transfer on 32-bit data bus is shown in Figure 3-10.
11. AXI4 8-bit narrow write transfer on 64-bit data bus is shown in Figure 3-11.
12. AXI4 burst read transfer of length 4 that crosses the 1 KB address on the AHB-Lite is shown in Figure 3-12. One AXI4 read transfer is split into two AHB-Lite read transfers as the 1 KB boundary is crossed.
13. AXI4 burst write transfer of length 4 that crosses the 1 KB Address on the AHB-Lite is shown in Figure 3-13. One AXI4 write transfer is split into two AHB-Lite write transfers as the 1 KB boundary is crossed.
14. AXI4 read timeout of 16 is shown in Figure 3-14. AXI4 read transfer is completed with a SLVERR response.
15. AXI4 write timeout of 16 is shown in Figure 3-15. AXI4 write transfer is completed with a SLVERR response.

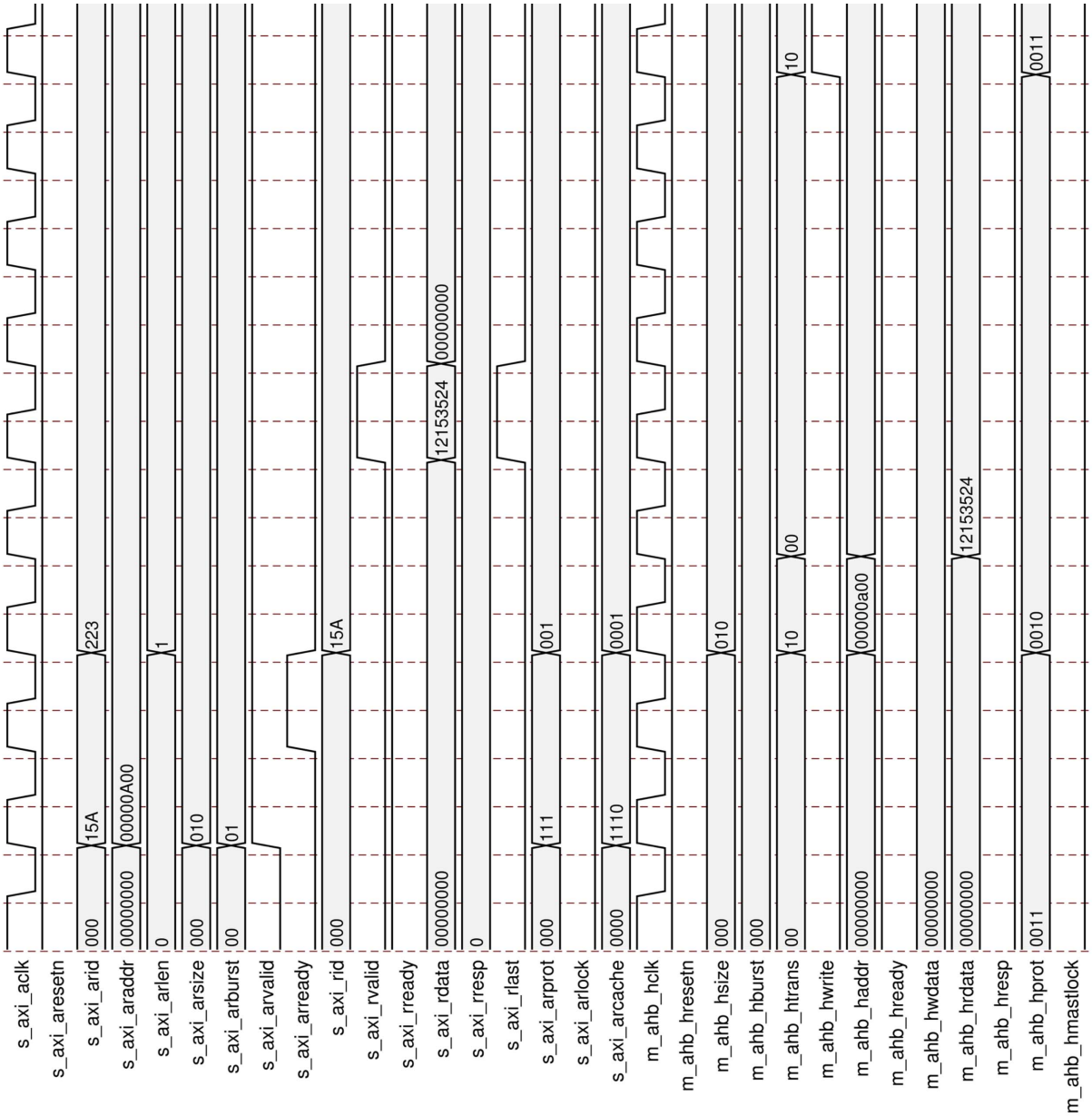

Figure 3-1: **AXI4 Incrementing Burst Read Transfer of Length 1**

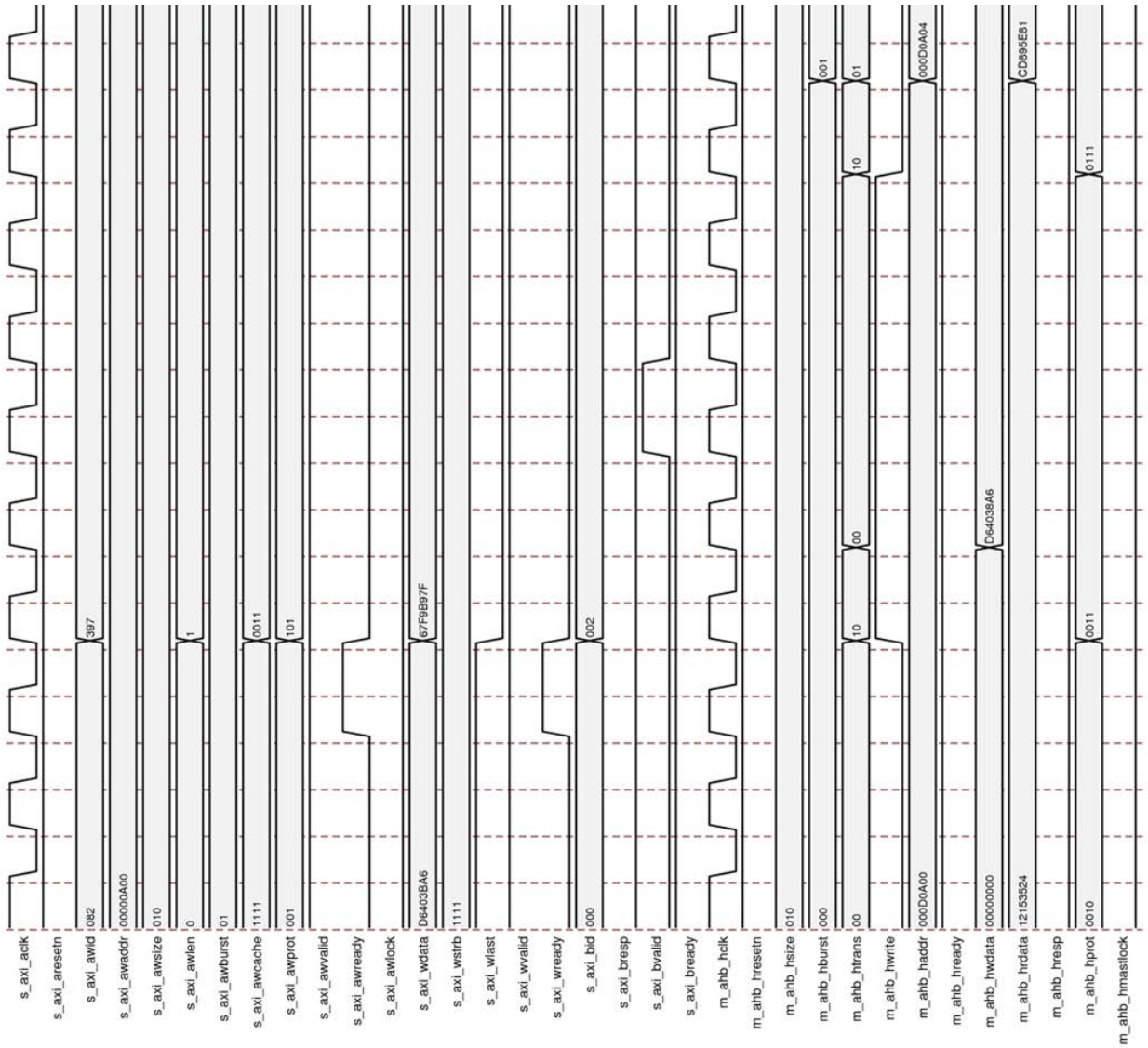

Figure 3-2: **AXI4 Incrementing Burst Write Transfer of Length 1**

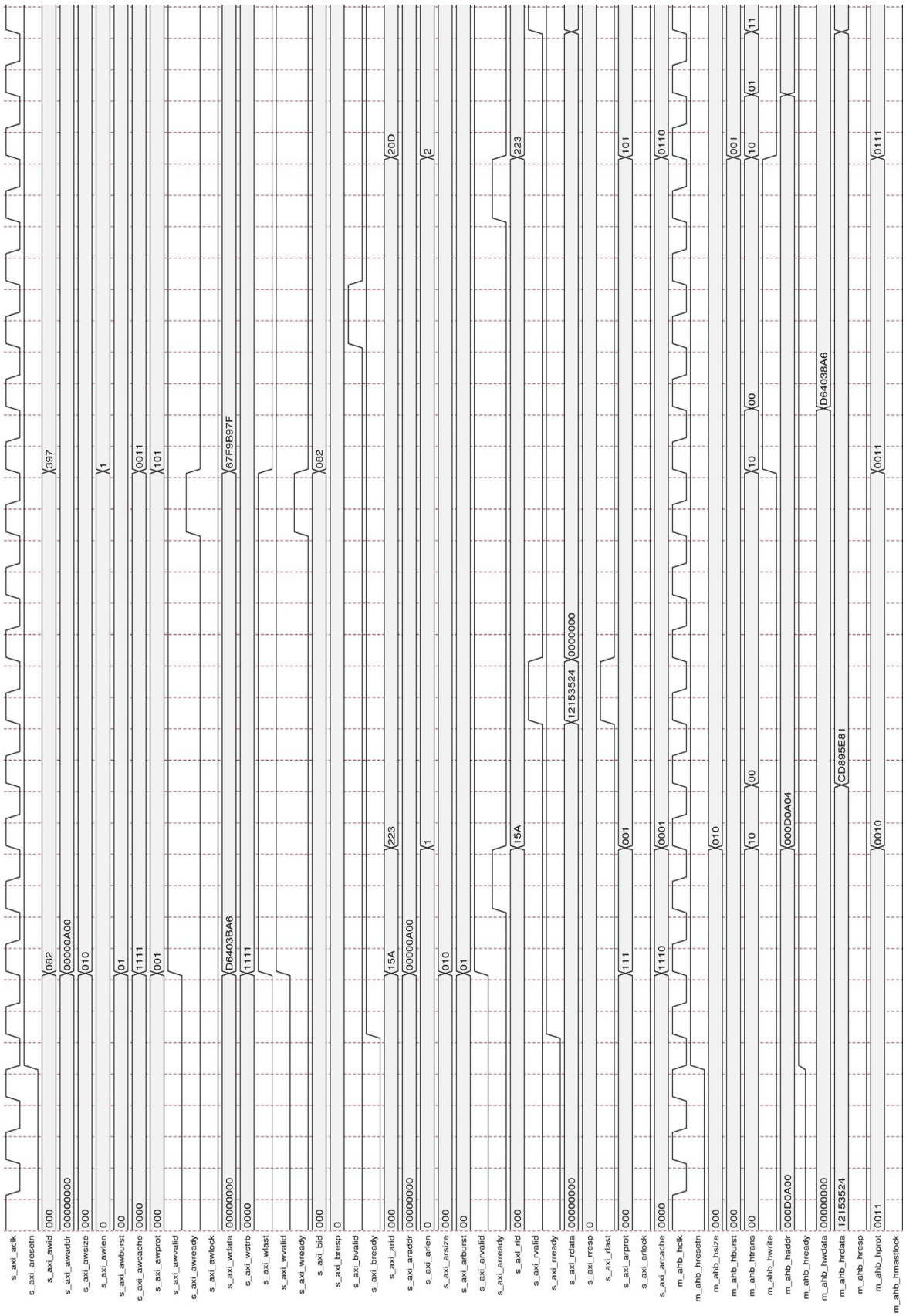

Figure 3-3: **AXI4 Read and Write Transfers**

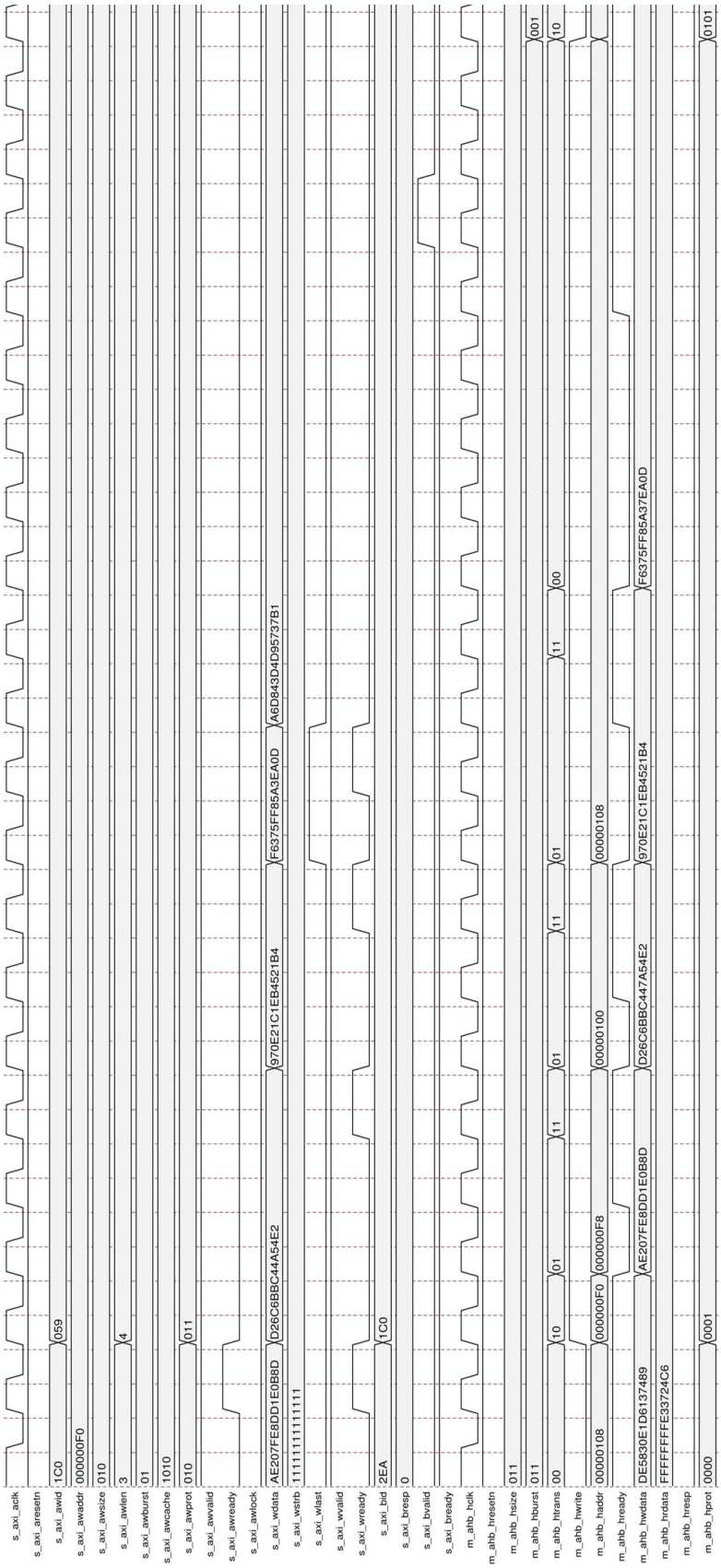

Figure 3-4: **AXI4 Incrementing Burst Write Transfer of Length 4**

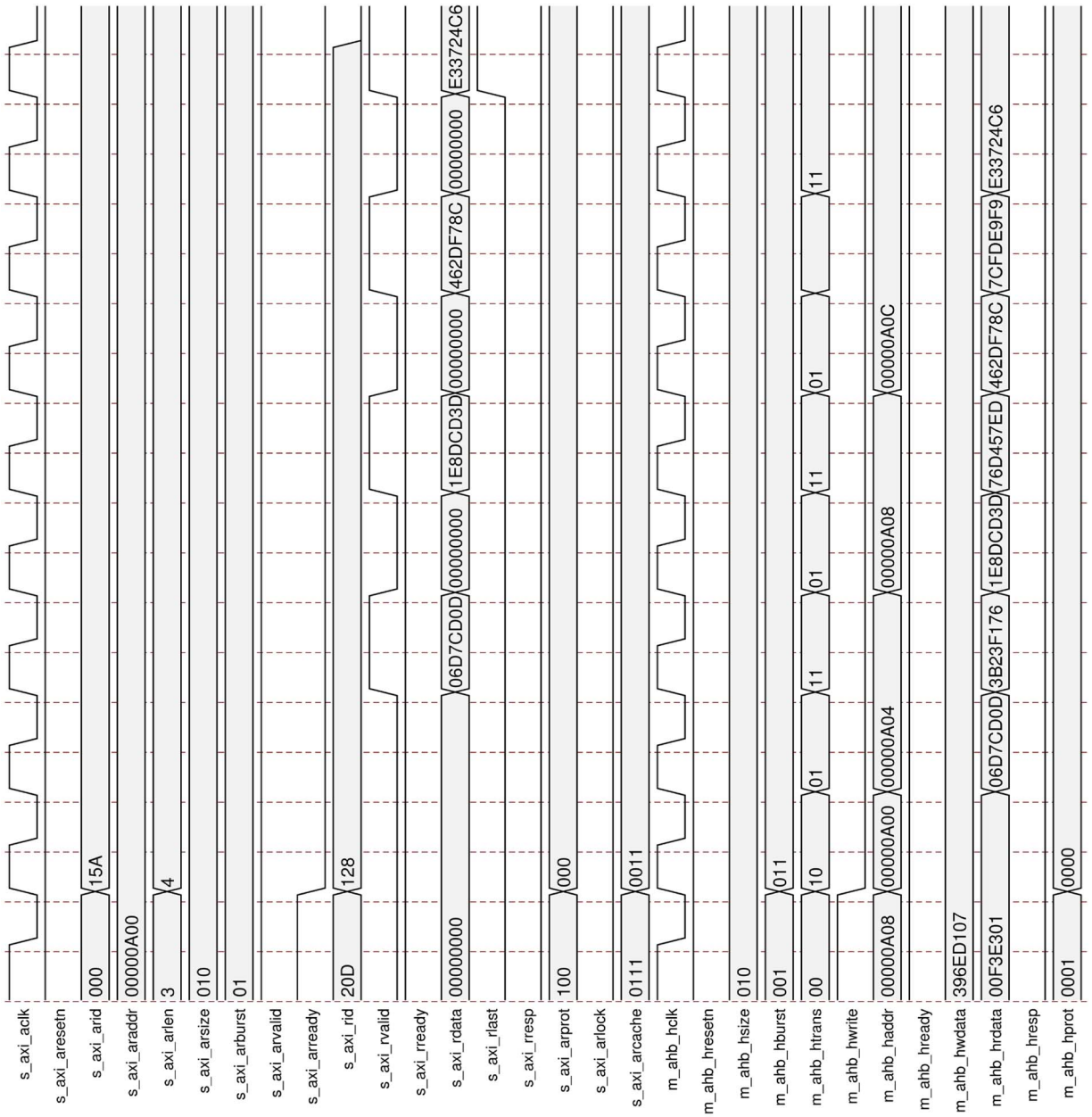

Figure 3-5: **AXI4 Incrementing Burst Read Transfer of Length 4**

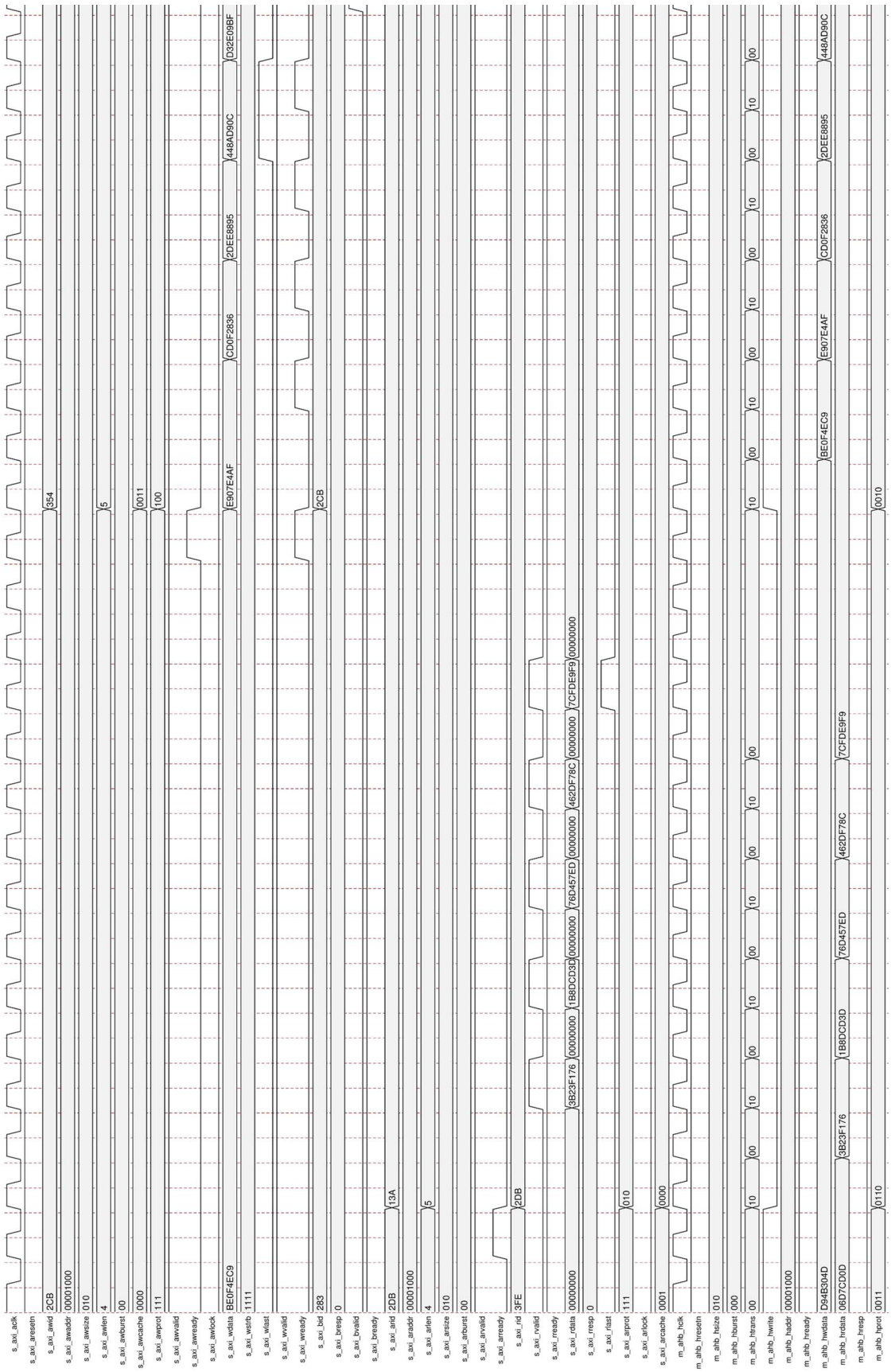

Figure 3-6: **AXI4 Fixed Burst Read and Write Transfers**

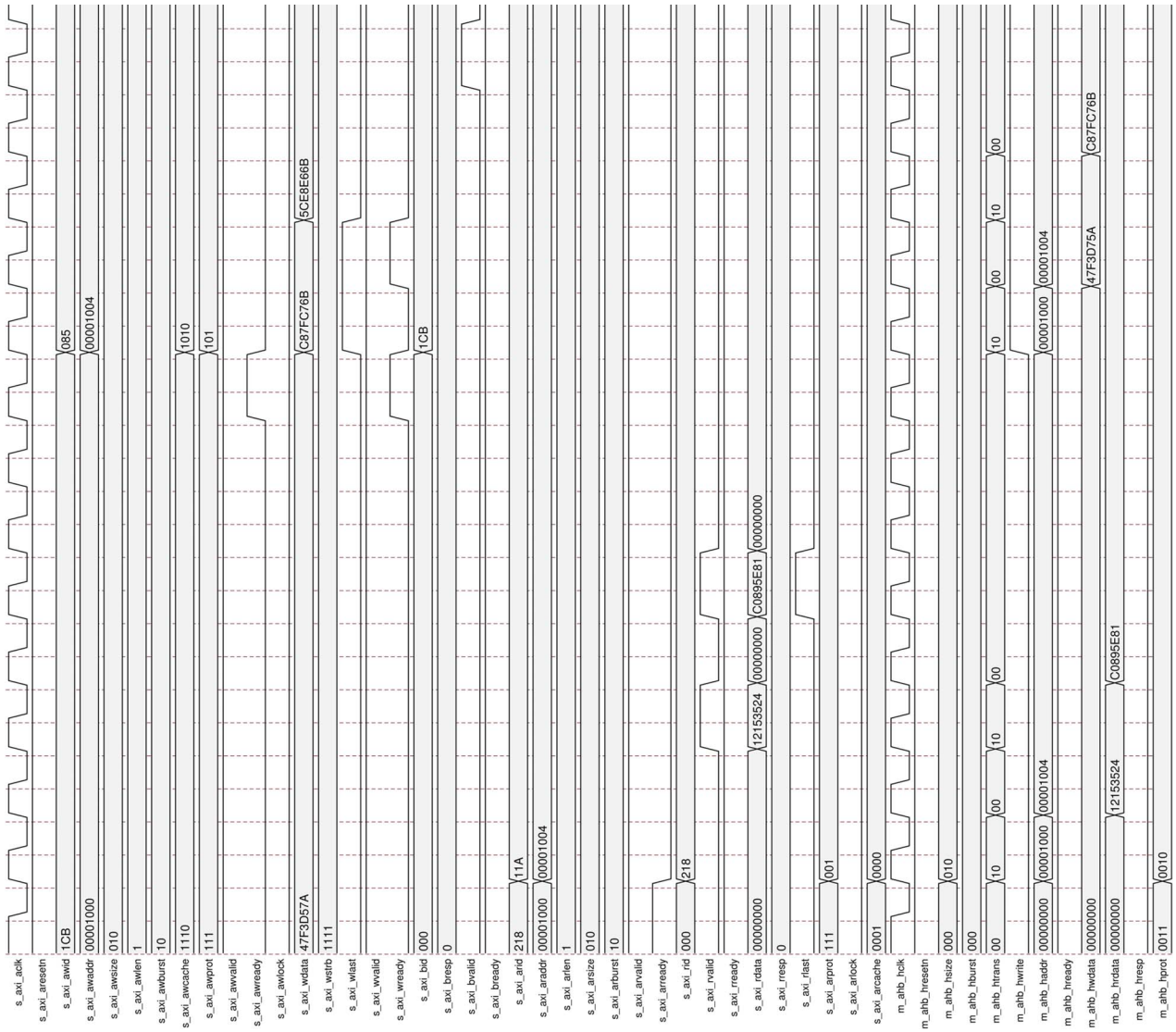

Figure 3-7: **AXI4 Wrapping Burst Read and Write Transfer of Length 2**

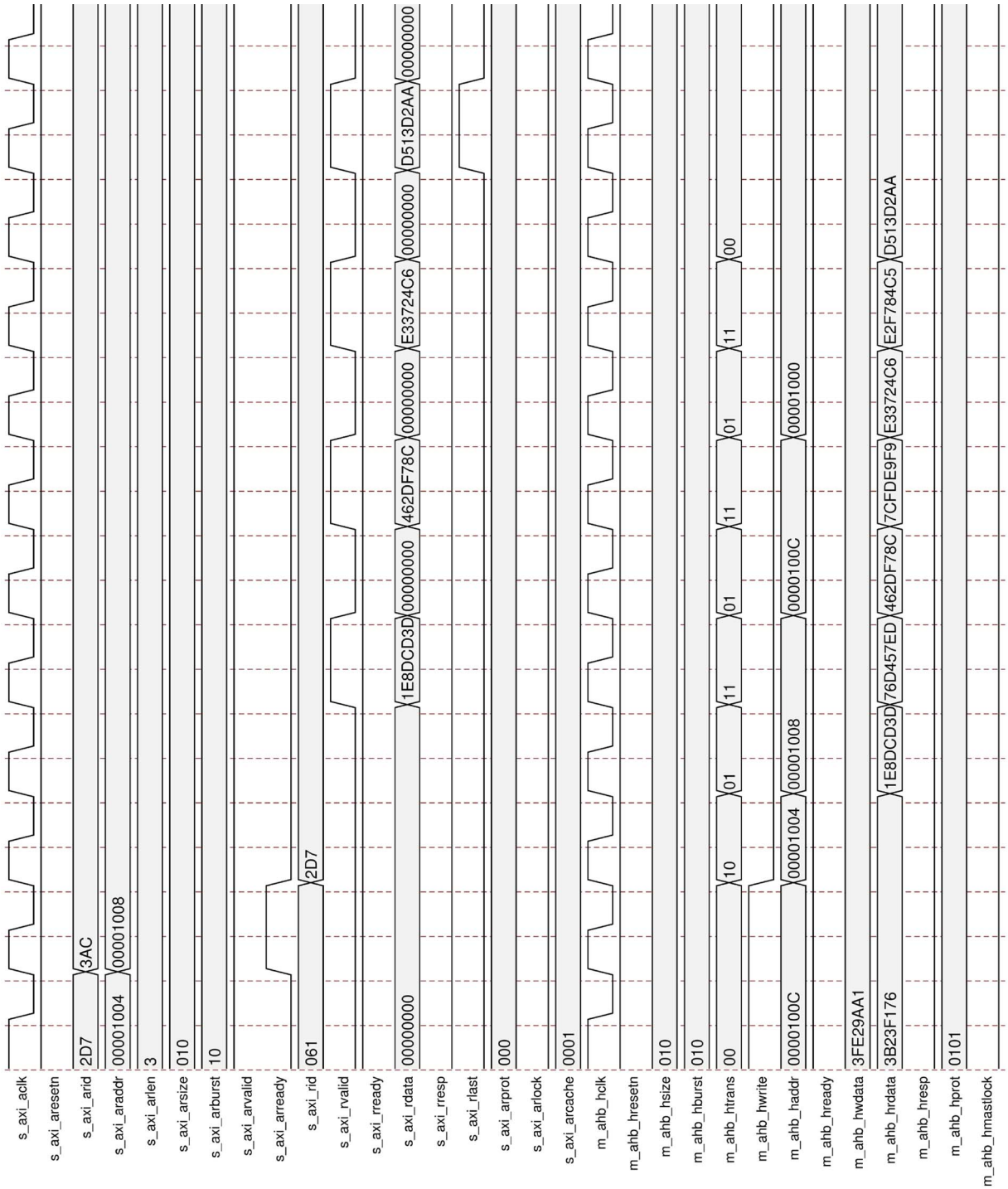

Figure 3-8: **AXI4 Wrapping Burst Read Transfer of Length 4**

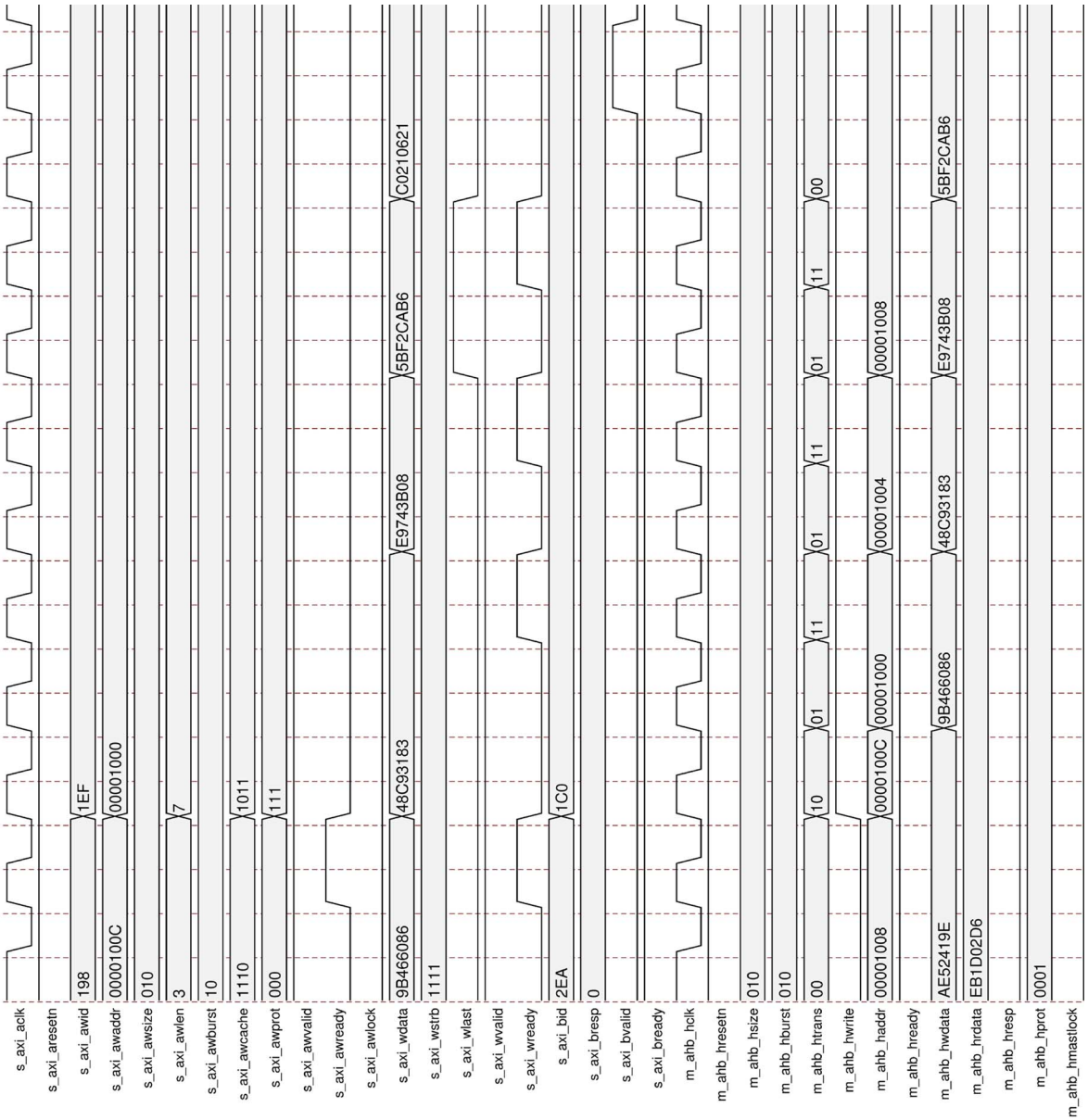

Figure 3-9: **AXI4 Wrapping Burst Write Transfer of Length 4**

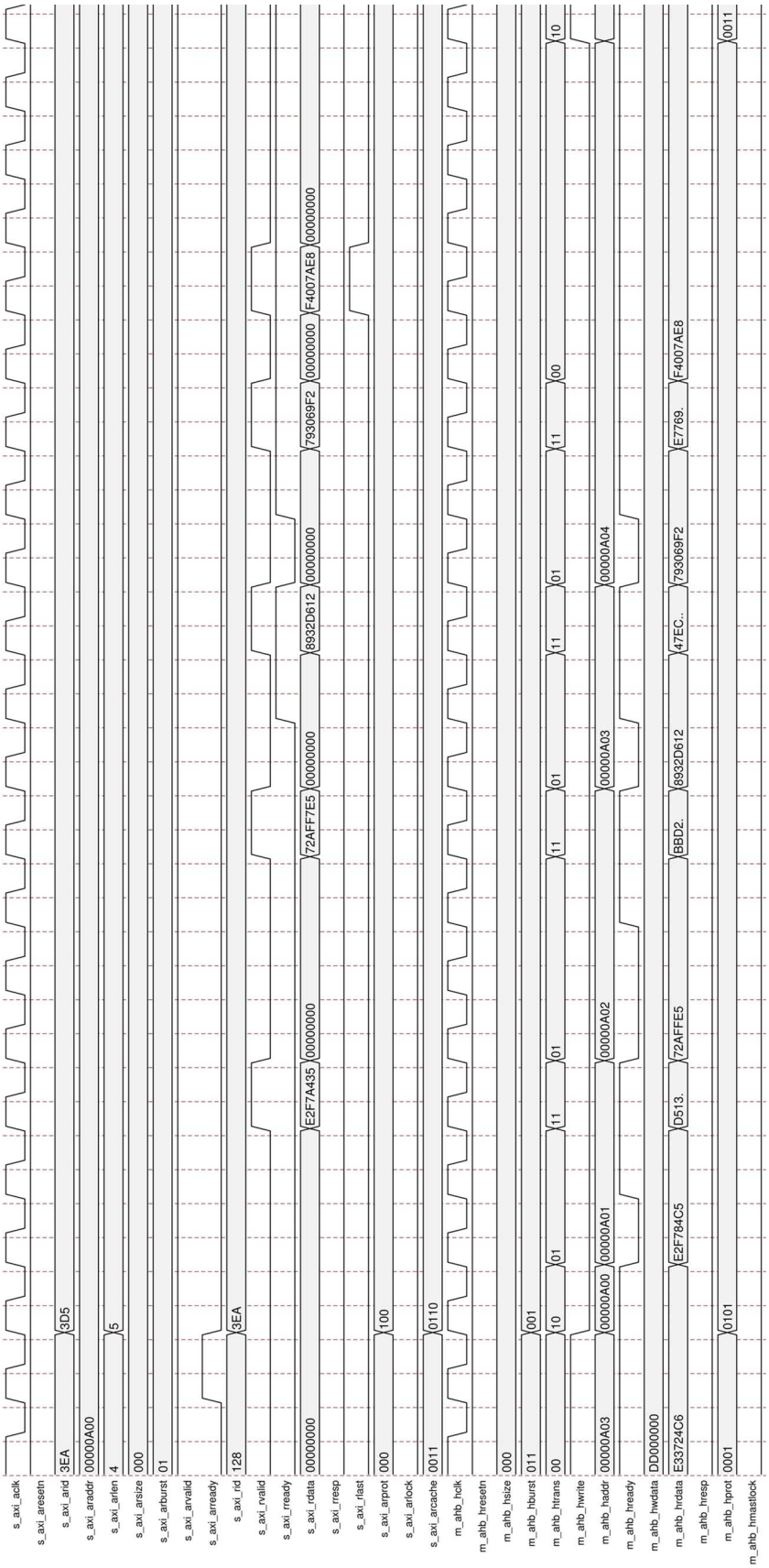

Figure 3-10: **AXI4 8-bit Narrow Read Transfer on 32-bit Data Bus**

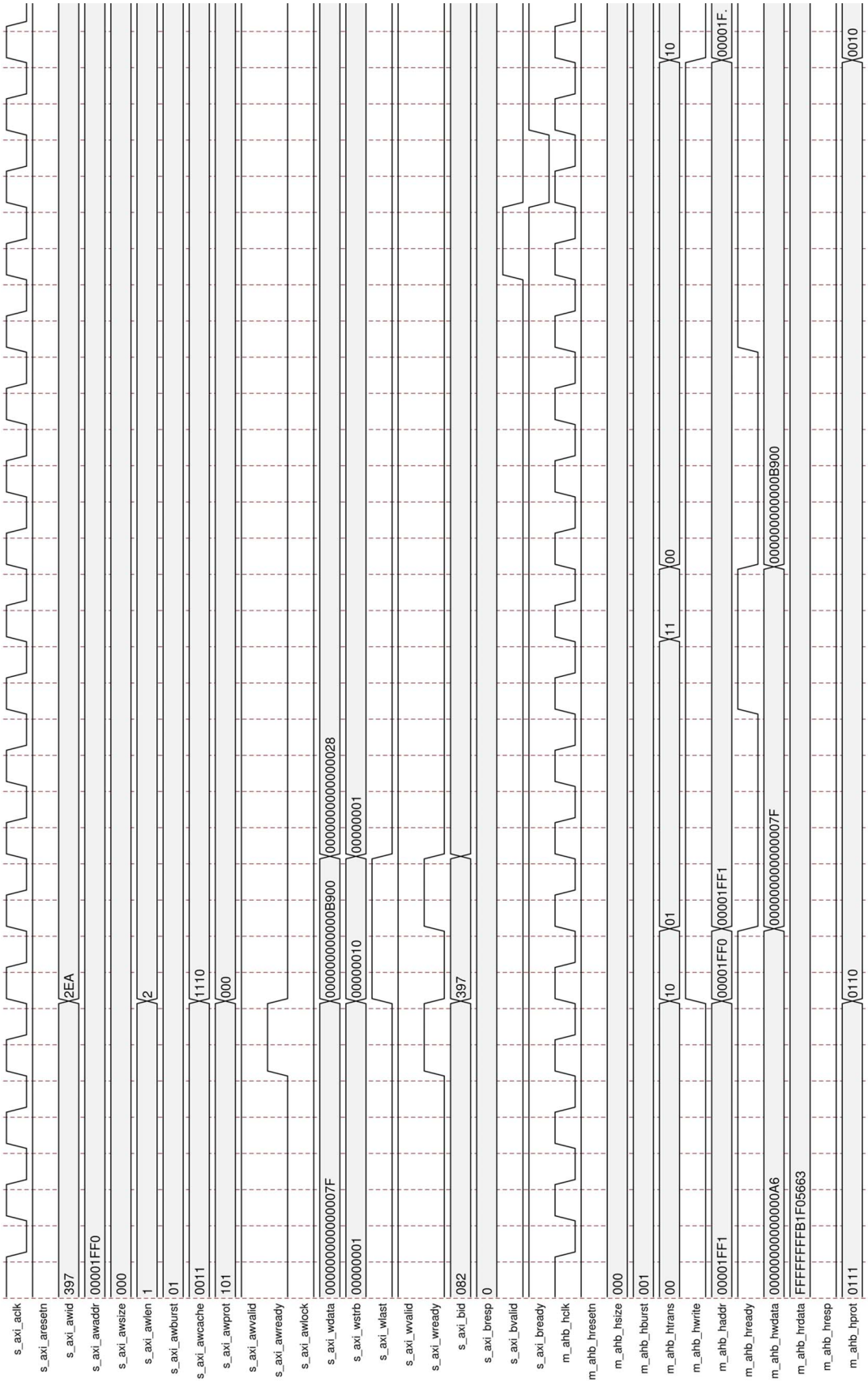

Figure 3-11: **AXI4 8-bit Narrow Write Transfer on 64-bit Data Bus**

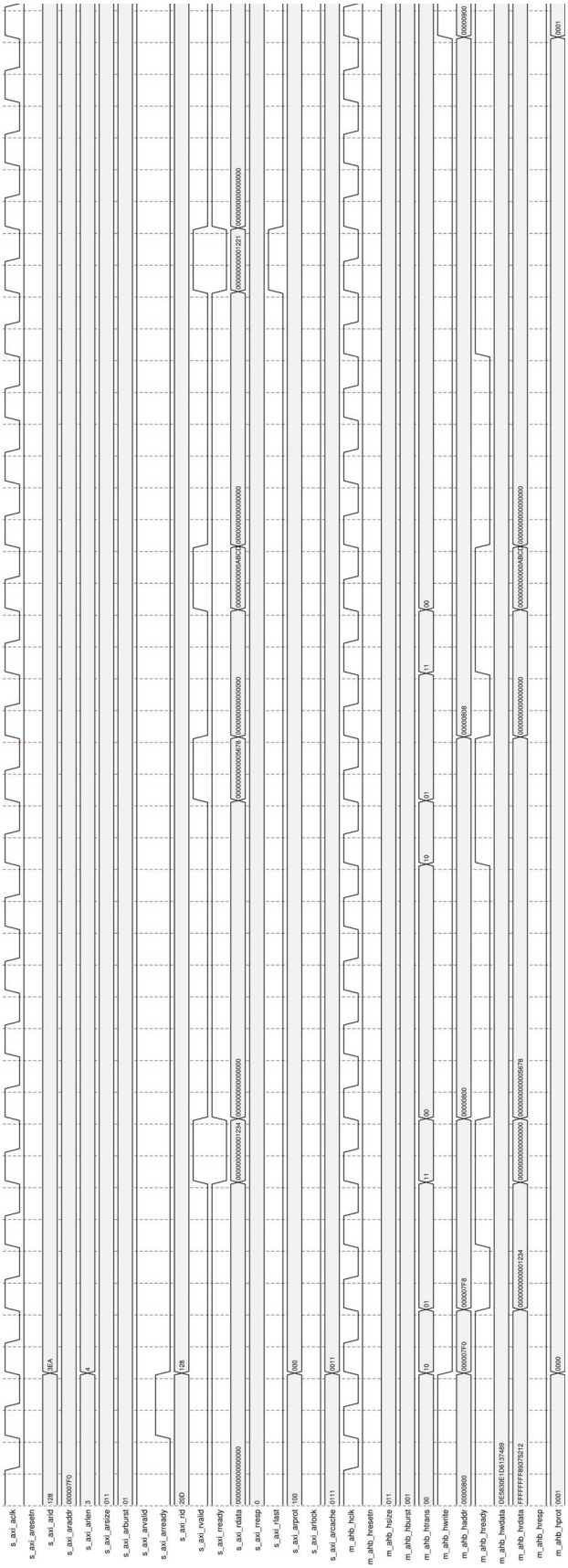

Figure 3-12: **AXI4 Burst Read Transfer of Length 4 That Crosses 1 KB Address Boundary on AHB-Lite**

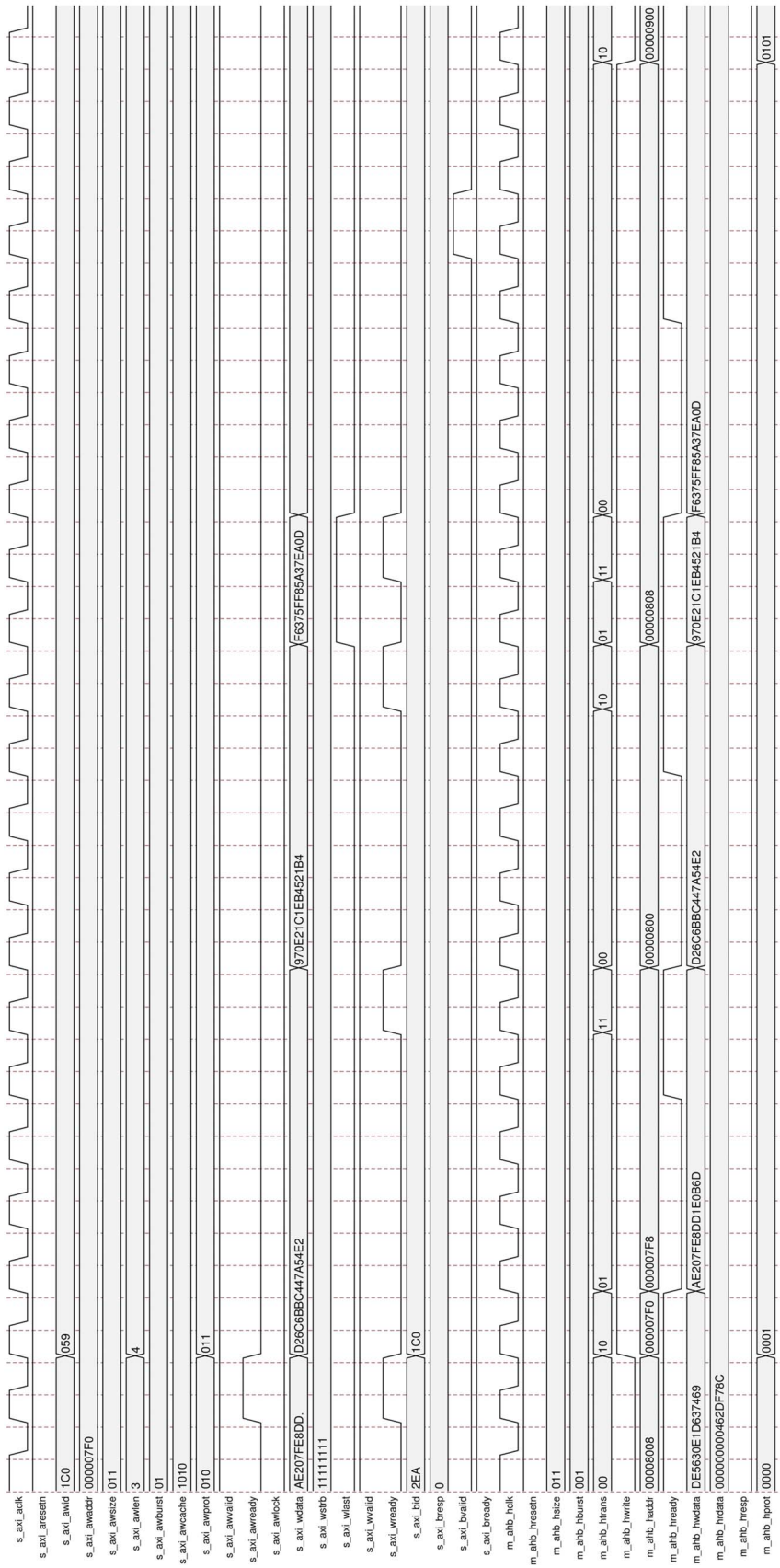

Figure 3-13: **AXI4 Burst Write Transfer of Length 4 That Crosses 1 KB Address Boundary on AHB-Lite**

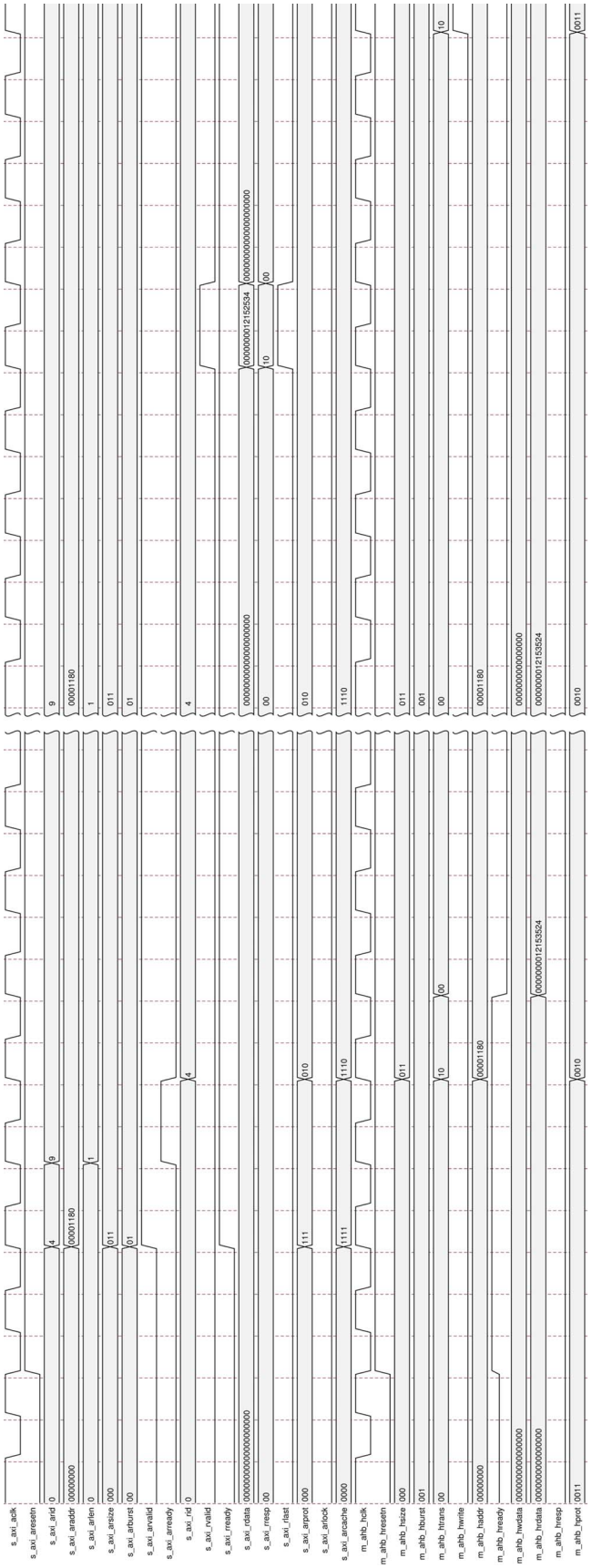

Figure 3-14: **AXI4 Read Timeout of 16**

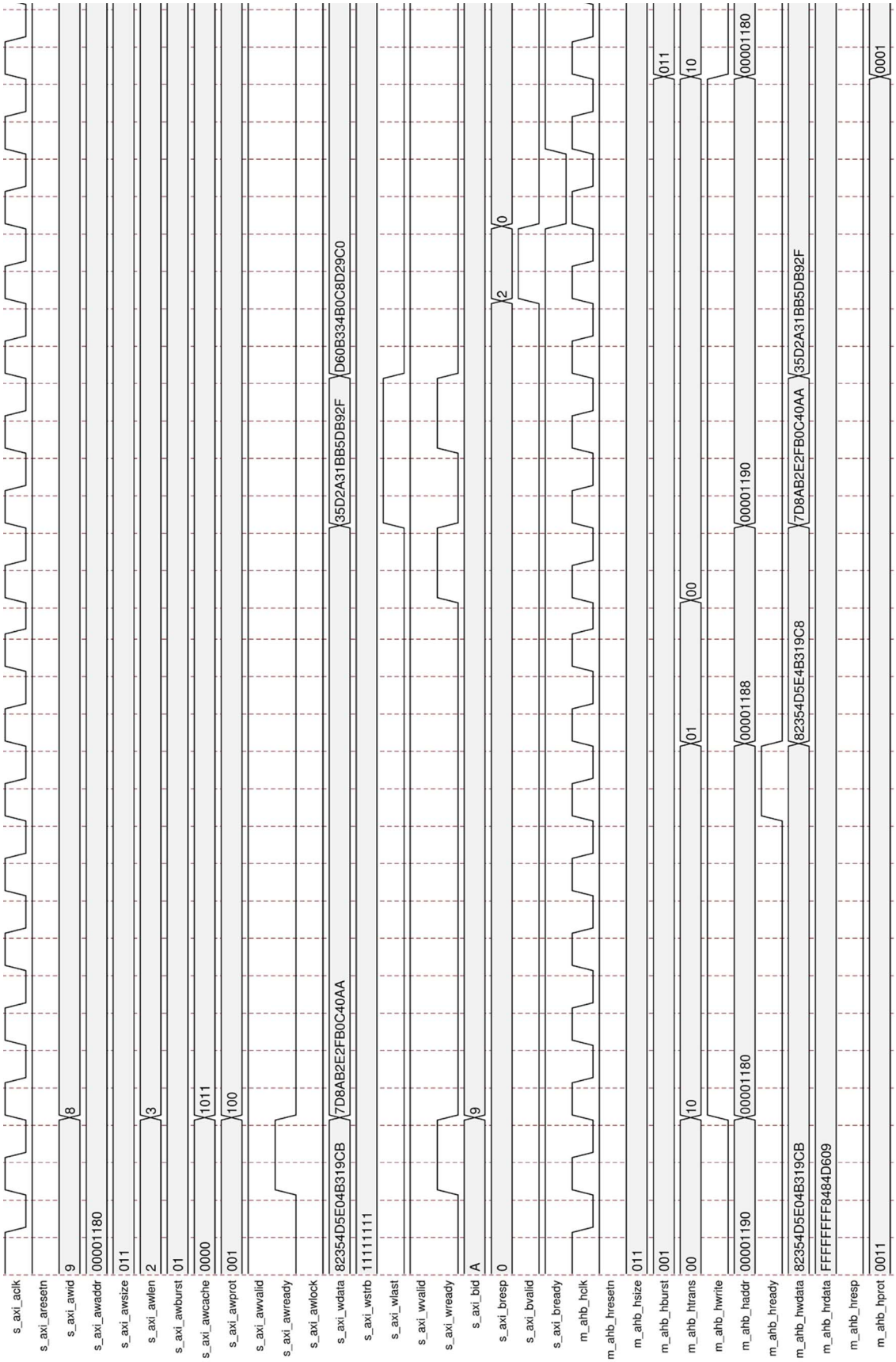

Figure 3-15: **AXI4 Write Timeout of 16**

Customizing and Generating the Core

This chapter includes information about using Xilinx tools to customize and generate the core in the Vivado® Design Suite.

Vivado Integrated Design Environment

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or popup menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3].

If you are customizing and generating the core in the Vivado IP integrator, see *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) [Ref 4] for detailed information. Vivado Integrated Design Environment (IDE) might auto-compute certain configuration values when validating or generating the design, as noted in this section. You can view the parameter value after successful completion of `validate_bd_design` command.

Note: Figures in this chapter are illustrations of the Vivado IDE. This layout might vary from the current version.

Figure 4-1 shows the Customize IP window settings for AXI4 to AHB-Lite Bridge IP core.

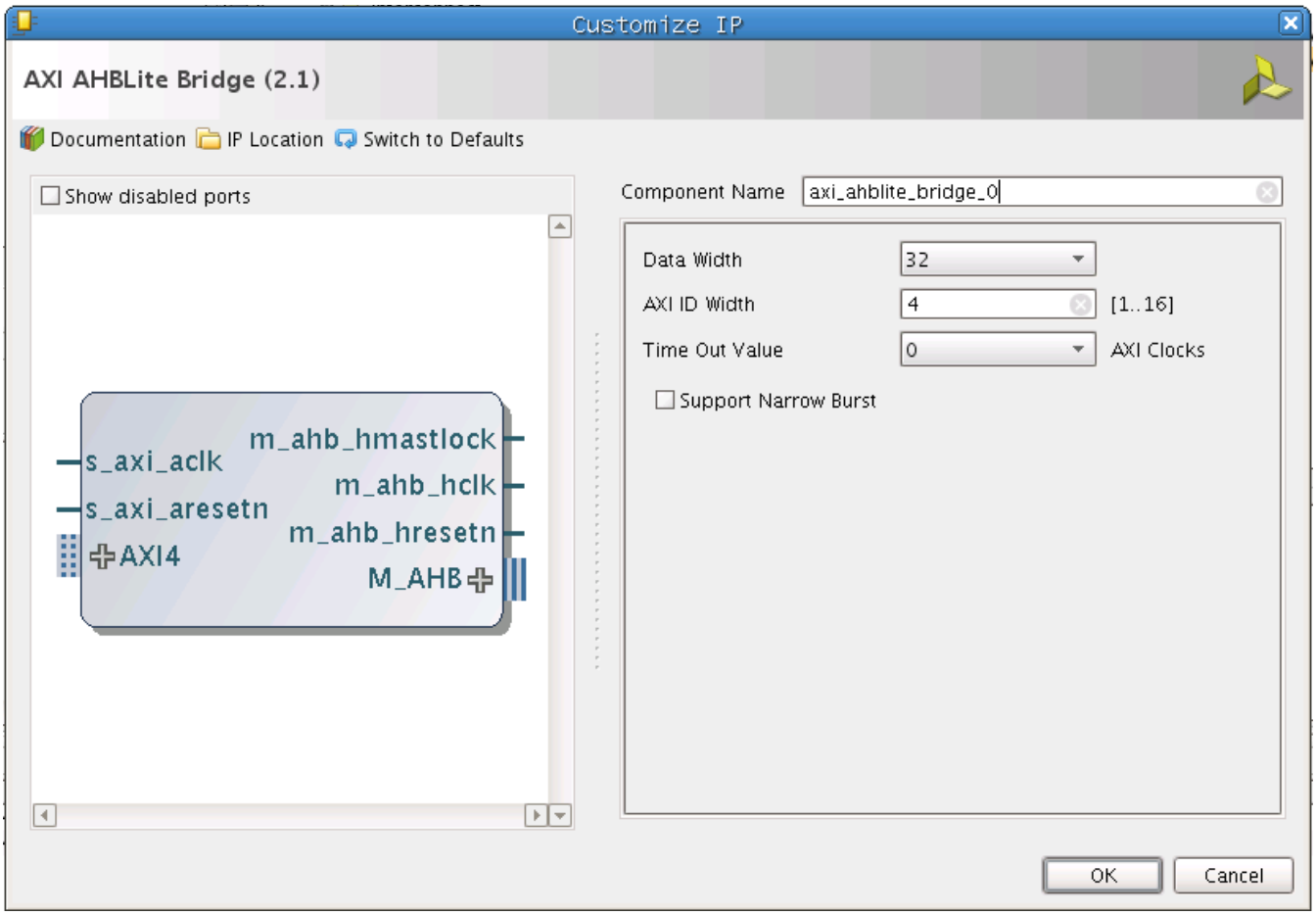

Figure 4-1: **Vivado Customize IP Dialog Box**

Figure 4-2 shows the Customize IP window settings with IP integrator.

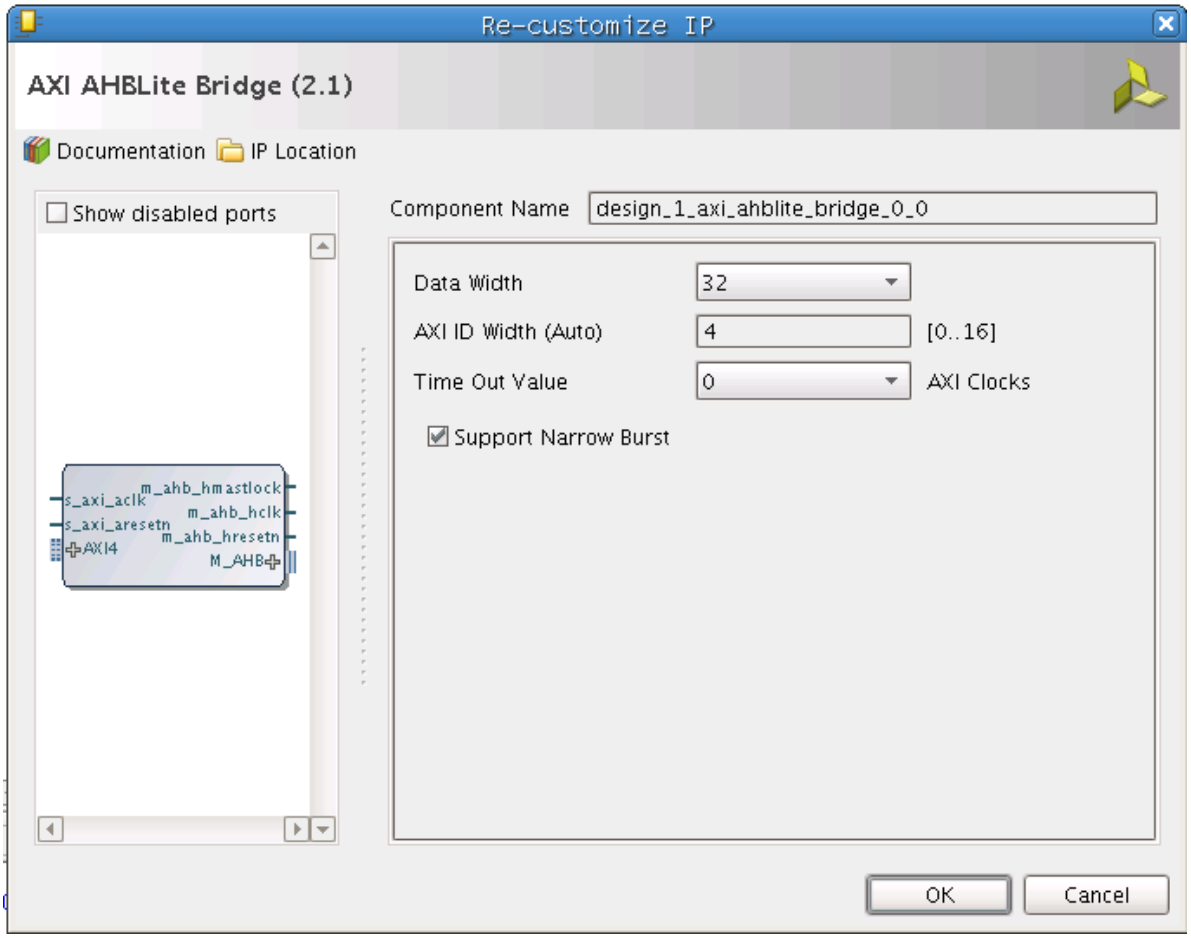

Figure 4-2: **Vivado IP Integrator**

- **Component Name** – The base name of the output files generated for the core. Names must begin with a letter and can be composed of any of the following characters: a to z, 0 to 9, and "_".
- **Data Width** – Select 32 or 64 AHB to AXI4 data width.
- **AXI4 ID Width** – Indicates the widths of ID tags.

  Note: In IP integrator, this cell is grayed-out.
- **Timeout Value** – Indicates the number of AXI4 clocks to wait for AHB-Lite slave to respond.
- **Support Narrow Burst** – Indicates the support for transactions with transfer size narrower than the data bus width.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Constraining the Core

There are no IP specific constraints. For details, see the *Vivado® Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Simulation

For comprehensive information about Vivado® simulation components, as well as information about using supported third party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5].

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado® Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Example Design

This chapter contains information about the provided example design in the Vivado® Design Suite environment.

The top module instantiates all components of the core and example design that are needed to implement the design in hardware, as shown in Figure 8-1. This includes clock generator (MMCME2) and example design module with logic to generate AXI4 transaction and AHB-Lite transaction checker.

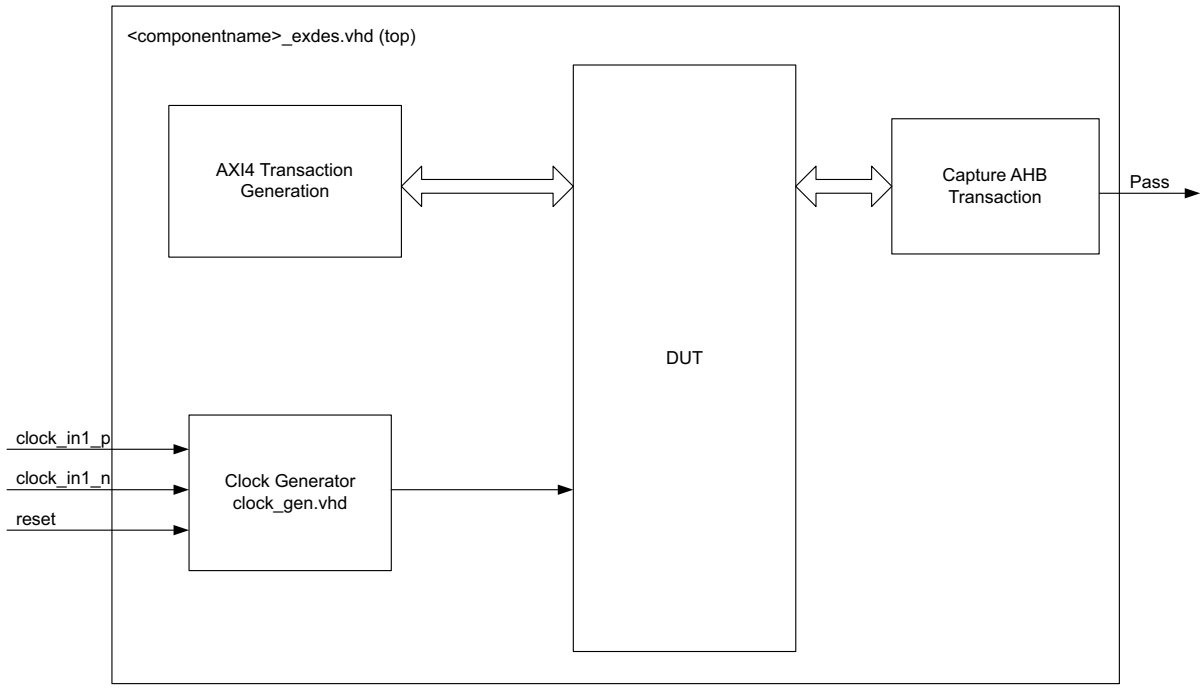

Figure 8-1: **AXI4 to AHB-Lite Bridge Example Design Block Diagram**

This example design demonstrates the transactions on AXI4 interface of the DUT.

- **Clock Generator** – MMCME2 is used to generate the clock for the example design. It generates 100 MHz clock for `s_axi_aclk` of the DUT. The example design DUT is kept under reset until MMCME2 is locked.
- **AXI4 Transaction Generation** – Supports the AXI4 write and read transactions on the AXI4 interface of the bridge.

- **Capture AHB Transaction** – Serves as the AHB-Lite slave for the bridge and manages write and read transactions from the AHB-Lite interface of the bridge.

Implementing the Example Design

After following the steps described in Chapter 4, Customizing and Generating the Core to generate the core, implement the example design as follows:

1. Right-click the core in the Hierarchy window, and select **Open IP Example Design**.
2. A new window pops up, asking you to specify a directory for the example design. Select a new directory or keep the default directory.
3. A new project is automatically created in the selected directory and it is opened in a new Vivado window.
4. You need to uncomment the I/O constraints settings in `<component_name>_exdes.xdc` specified in Table 8-3.
5. In the Flow Navigator (left-side pane), click **Run Implementation** and follow the directions.

Example Design Directory Structure

In the current project directory, a new project with the name `<component_name>_example` is created and the files are generated in the `<component_name>_example/<component_name>_example.srcs/` directory. This directory and its subdirectories contain all the source files that are required to create the AXI4 to AHB-Lite Bridge example design.

Table 8-1 shows the files delivered in the `<component_name>_example/<component_name>_example.srcs/sources_1/imports/example_design/` directory.

Table 8-1: **Example Design Directory**

Name	Description
<component_name>_exdes.vhd	Top-level HDL file for the example design.
clock_gen.vhd	Clock generation module for example design.

Table 8-2 shows the files delivered in `<component_name>_example/<component_name>_example.srcs/sources_1/sim_1/imports/simulation/` directory.

Table 8-2: **Simulation Directory**

Name	Description
<component_name>_exdes_tb.vhd	Test bench for example design.

Table 8-3 shows the files delivered in `<component_name>_example/<component_name>_example.srcs/sources_1/constrs_1/imports/example_design/` directory.

Table 8-3: **Constraints Directory**

Name	Description
<component_name>_exdes.xdc	Top-level constraints file for example design.

Simulating the Example Design

Using the AXI4 to AHB-Lite Bridge example design (delivered as part of the AXI4 to AHB-Lite Bridge), you can quickly simulate and observe the behavior of the core.

Setting Up the Simulation

The Xilinx simulation libraries must be mapped into the simulator. If the libraries are not set for your environment, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5] for assistance compiling Xilinx simulation models and setting up the simulator environment. To switch simulators, click **Simulation Settings** in the Flow Navigator (left pane). In the Simulation options list, change **Target Simulator**.

Simulation Results

The simulation script compiles the AHB-Lite to AXI4 Bridge example design and supporting simulation files. It then runs the simulation and checks to ensure that it completed successfully.

If the test passes, then the following message is displayed:

```
Test Completed Successfully
```

If the test fails or does not complete, then the following message is displayed:

```
Test Failed!! Test Timed Out.
```

Test Bench

This chapter contains information about the provided test bench in the Vivado® Design Suite environment.

Figure 9-1 shows the test bench for the AXI4 to AHB-Lite Bridge example design. The top-level test bench generates 200 MHz clock and drives initial reset to the example design.

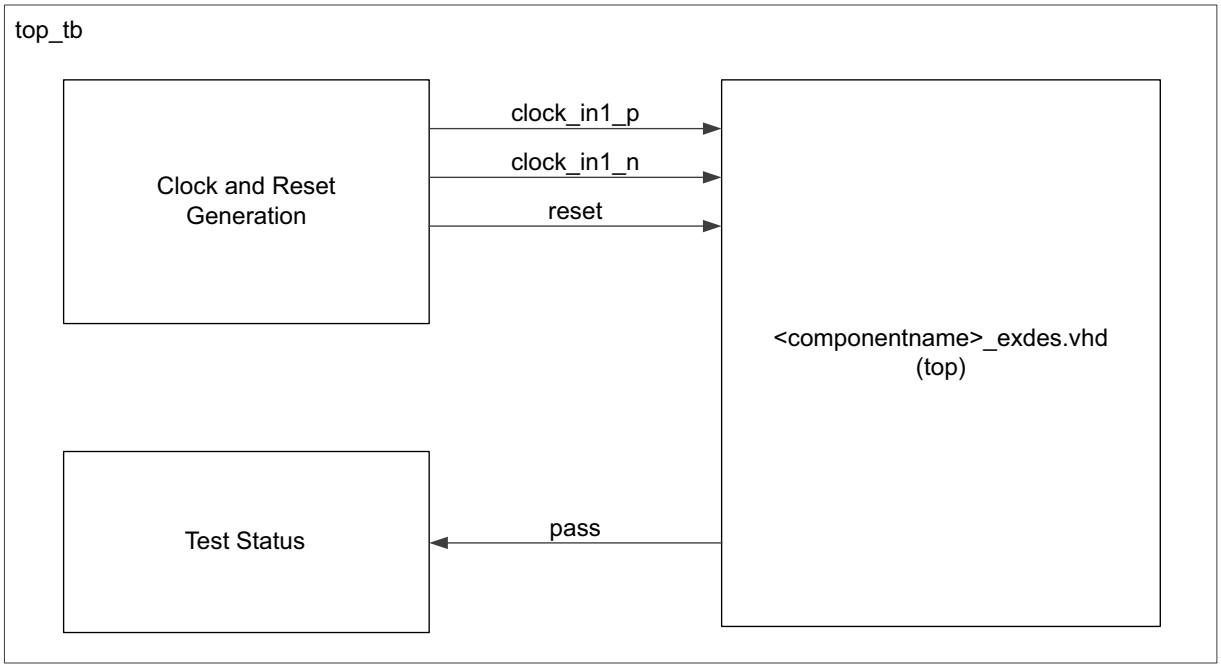

Figure 9-1: **AXI4 to AHB-Lite Bridge Example Design Test Bench**

Appendix A

Migrating and Upgrading

This appendix contains information about migrating a design from ISE® to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information on migrating to the Vivado Design Suite, see *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 6].

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI4 to AHB-Lite Bridge, the Xilinx Support web page (www.xilinx.com/support) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the AXI4 to AHB-Lite Bridge. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page (www.xilinx.com/support) or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page (www.xilinx.com/download). For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can also be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as:

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the AXI4 to AHB-Lite Bridge

AR: 54425

Contacting Technical Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support:

1. Navigate to www.xilinx.com/support.
2. Open a WebCase by selecting the WebCase link located under Additional Resources.

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx Design Tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file(s) to include with the WebCase.

Note: Access to WebCase is not available in all cases. Login to the WebCase tool to see your specific support options.

Debug Tools

There are many tools available to address AXI4 to AHB-Lite Bridge design issues. It is important to know which tools are useful for debugging various situations.

Vivado Lab Tools

Vivado® lab tools insert logic analyzer and virtual I/O cores directly into your design. Vivado lab tools allow you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature represents the functionality in the Vivado IDE that is used for logic debugging and validation of a design running in Xilinx devices in hardware.

The Vivado lab tools logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 7].

Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:

www.xilinx.com/company/terms.htm.

References

These documents provide supplemental material useful with this product guide:

1. *ARM® AMBA® AXI4-Stream Protocol Specification*
2. *Vivado® Design Suite User Guide: Designing with IP* (UG896)
3. *Vivado Design Suite User Guide: Getting Started* (UG910)
4. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994)
5. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
6. *ISE® to Vivado Design Suite Migration Guide* (UG911)
7. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
8. *Vivado Design Suite User Guide: Implementation* (UG904)
9. *7 Series FPGAs Overview* (DS180)
10. *LogiCORE IP AXI Interconnect Product Guide* (PG059)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/02/2013	2.1	Initial Xilinx public release of document in product guide format. Replaces DS827.
12/18/2013	2.1	Added UltraScale support.

Notice of Disclaimer