AXI4 to AHB-Lite Bridge v3.0

LogiCORE IP Product Guide

Vivado Design Suite

PG177 November 18, 2015

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
- 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+™ Families, UltraScale™ Architecture, Zynq®-7000 All Programmable SoC, 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
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page.	

Notes:

1. For a complete list of supported devices, see the 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.

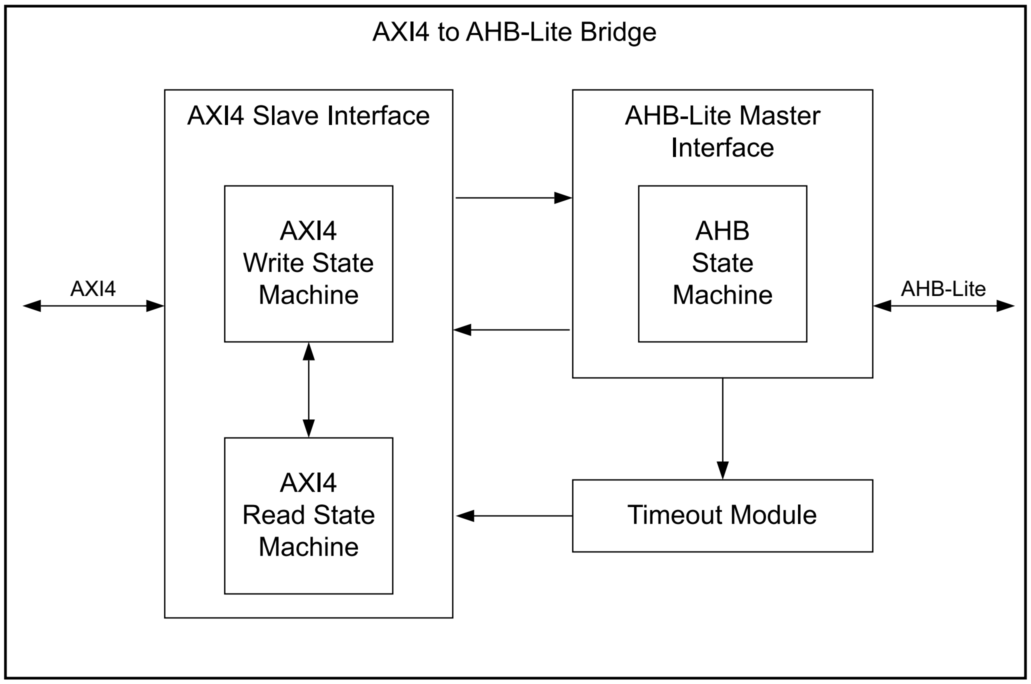


Figure 1-1: **AXI4 to AHB-Lite Bridge Block Diagram**

AXI4 Slave Interface

The AXI4 slave interface module provides a bidirectional slave interface. The AXI4 address width can be configured in the Vivado IDE from values 32 to 64 bits. The AXI4 data bus width can either be 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 can be configured in the Vivado IDE from values 32 to 64 bits and the data bus width can either be 32 or 64 bits.

The AXI4 to AHB-Lite Bridge supports the same data width on both the AXI4 and AHB-Lite interfaces.

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 that parameter value is non-zero. If AHB-Lite slave is not responding, timeout module waits on the number of AXI4 clocks specified through the timeout parameter for the AHB-Lite slave response and then generates the timeout.

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 the *Vivado Design Suite User Guide: Designing With IP* (UG896) [Ref 2].

Note: Maximum frequency numbers for UltraScale™ architecture and 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		
Supports Narrow Burst	**Data Width**	**ID Width**	**Bridge Timeout**	**Slices**	**Slice Flip-Flops**	**LUTs**
0	32	4	0	182	427	287
1	32	4	0	179	428	295
0	64	4	0	216	584	338
1	64	16	0	239	662	356
0	64	16	0	247	656	346
0	64	16	128	226	667	336
1	64	4	256	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 Channel Signals				
s_axi*	AXI4	I	–	See Appendix A of the *Vivado AXI Reference Guide* (UG1037) [Ref 3] for a description of AXI4 signals.
AHB-Lite Signals				
m_ahb_haddr	AHB-Lite	O	0	This is the AHB address bus and the width of the port varies from 32 to 64 bits based on the Vivado IDE selection.
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.
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.

Notes:

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 implemented in AXI4 to AHB-Lite Bridge.

Chapter 3

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 and uses the `s_axi_aclk` at both the AXI4 and AHB-Lite interfaces.

Resets

`s_axi_aresetn` is a synchronous reset input that resets the AXI4 to AHB-Lite Bridge upon assertion. `s_axi_aresetn` is also used to reset the AHB-Lite interface.

Memory Mapping

The AXI4 memory map and the AHB-Lite memory map might vary from a 32-bit to 64-bit memory space. The AXI4 to AHB-Lite Bridge does not modify the address for AHB-Lite. Hence, the address that is presented on the AHB-Lite is exactly as received on the AXI4.

Data Width

The AXI4 to AHB-Lite Bridge supports the same data width on either sides of the bridge interface (AHB and AXI4). The data width selection can be set to 32 or 64.

Narrow Transfers

Transactions where the transfer size is narrower than the data width are treated as narrow transfers. Narrow transfer support is parameterized in the AXI4 to AHB-Lite Bridge. The core must be generated with this parameter set to TRUE to support narrow transfers.

The 8/16-bit or 8/16/32-bit narrow transfers are allowed through the bridge when data width is 32 or 64 respectively.

- **Single Transfer** – In the case of AXI single transaction (for example, awlen == 0), the IP monitors `s_axi_wstrb` and drives `m_ahb_hsize`
- **Burst Transfer** – In the case of AXI burst transfer request, the AXI to AHB-Lite passes the `s_axi_awsize/s_axi_arsize` to `m_ahb_hsize`

Sparse/unaligned transfers are not supported in narrow transfers.

Endianness

Both AXI4 and AHB-Lite interfaces are little endian.

Address/Data Translation

The AXI4 to AHB-Lite Bridge aligns the address on the AHB-Lite master interface for any unaligned read request on the AXI4 interface. As per AHB-Lite protocol there is no concept of unaligned addressing.

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.

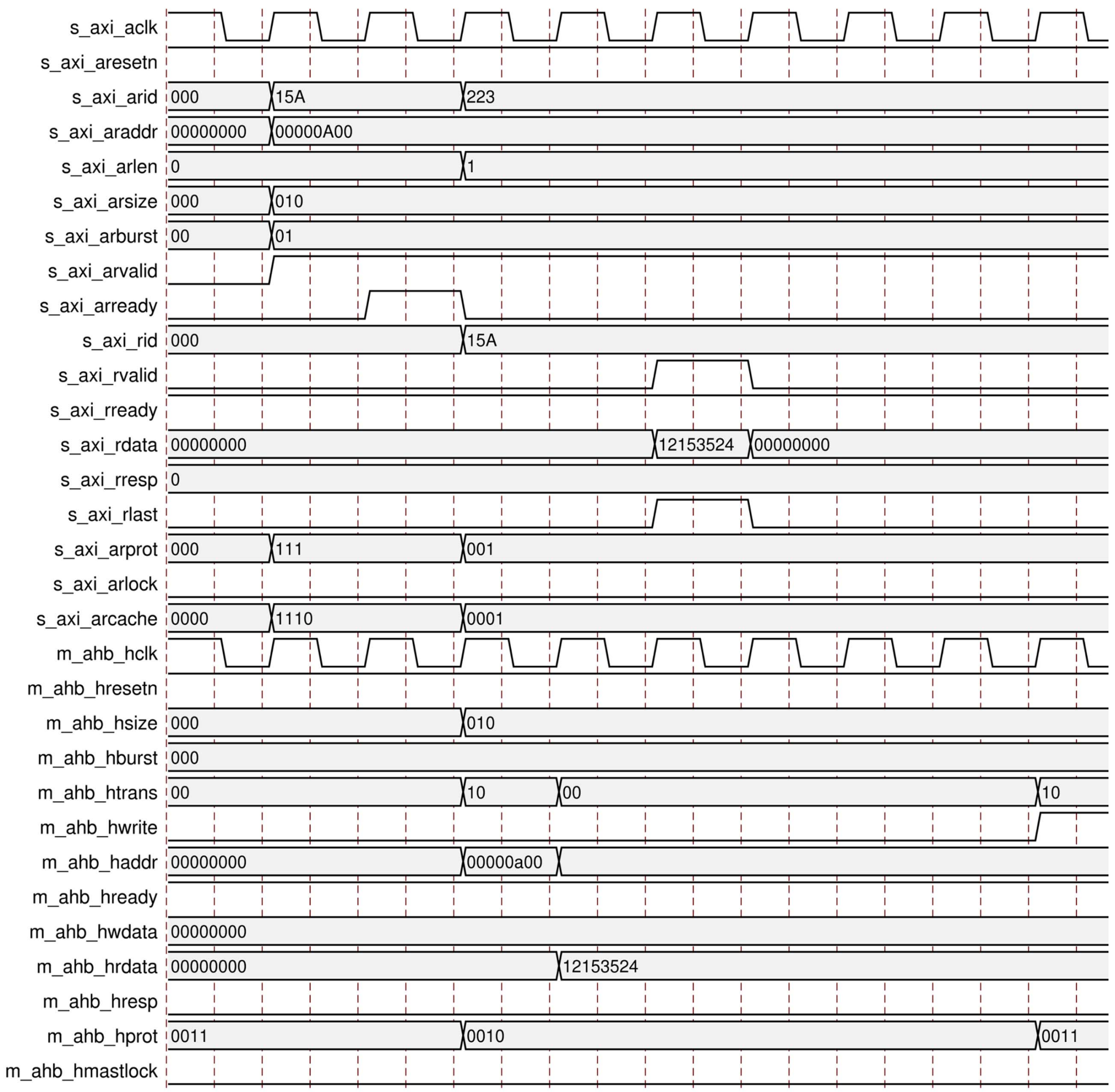


Figure 3-1: **AXI4 Incrementing Burst Read Transfer of Length 1**

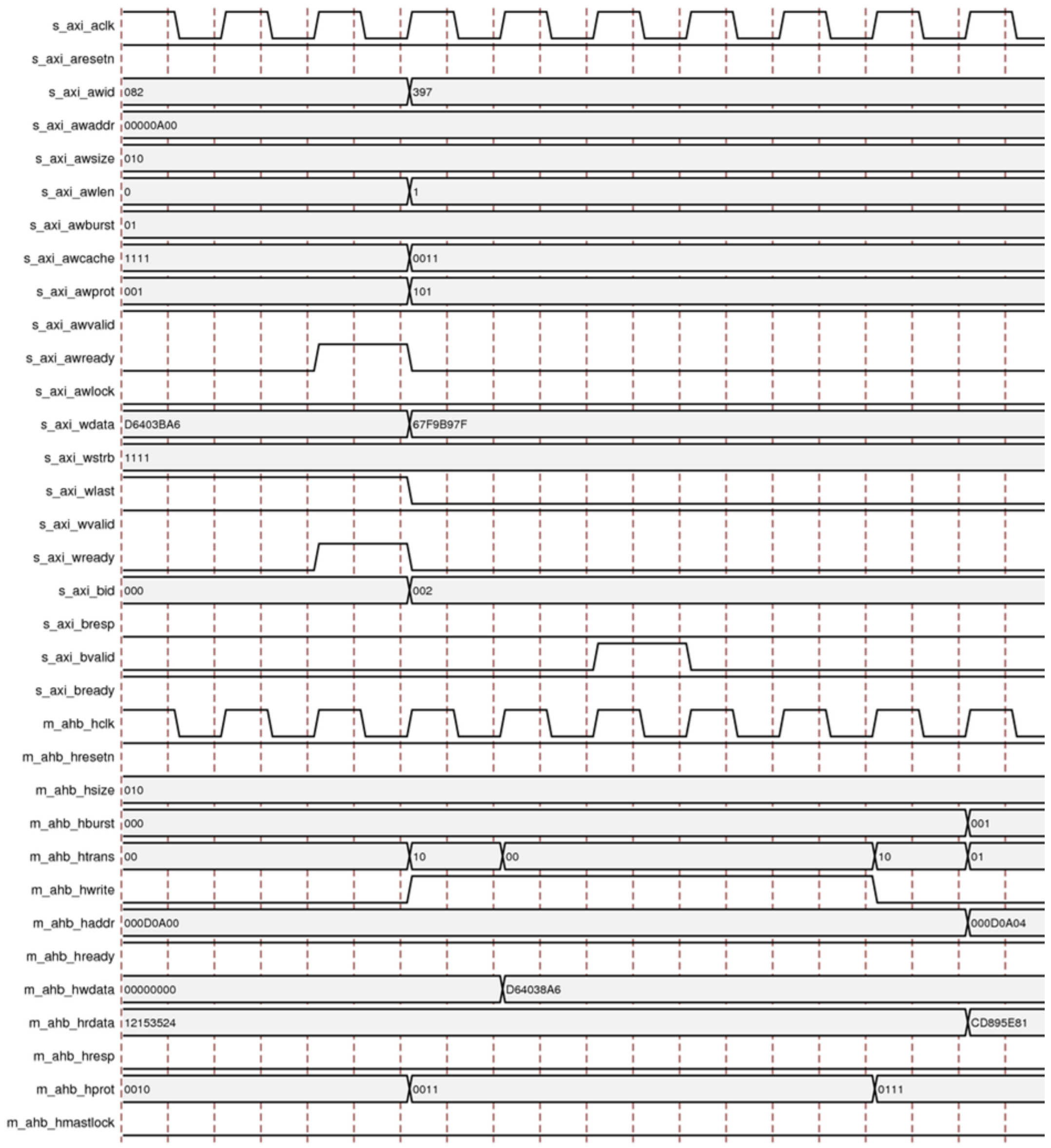


Figure 3-2: **AXI4 Incrementing Burst Write Transfer of Length 1**

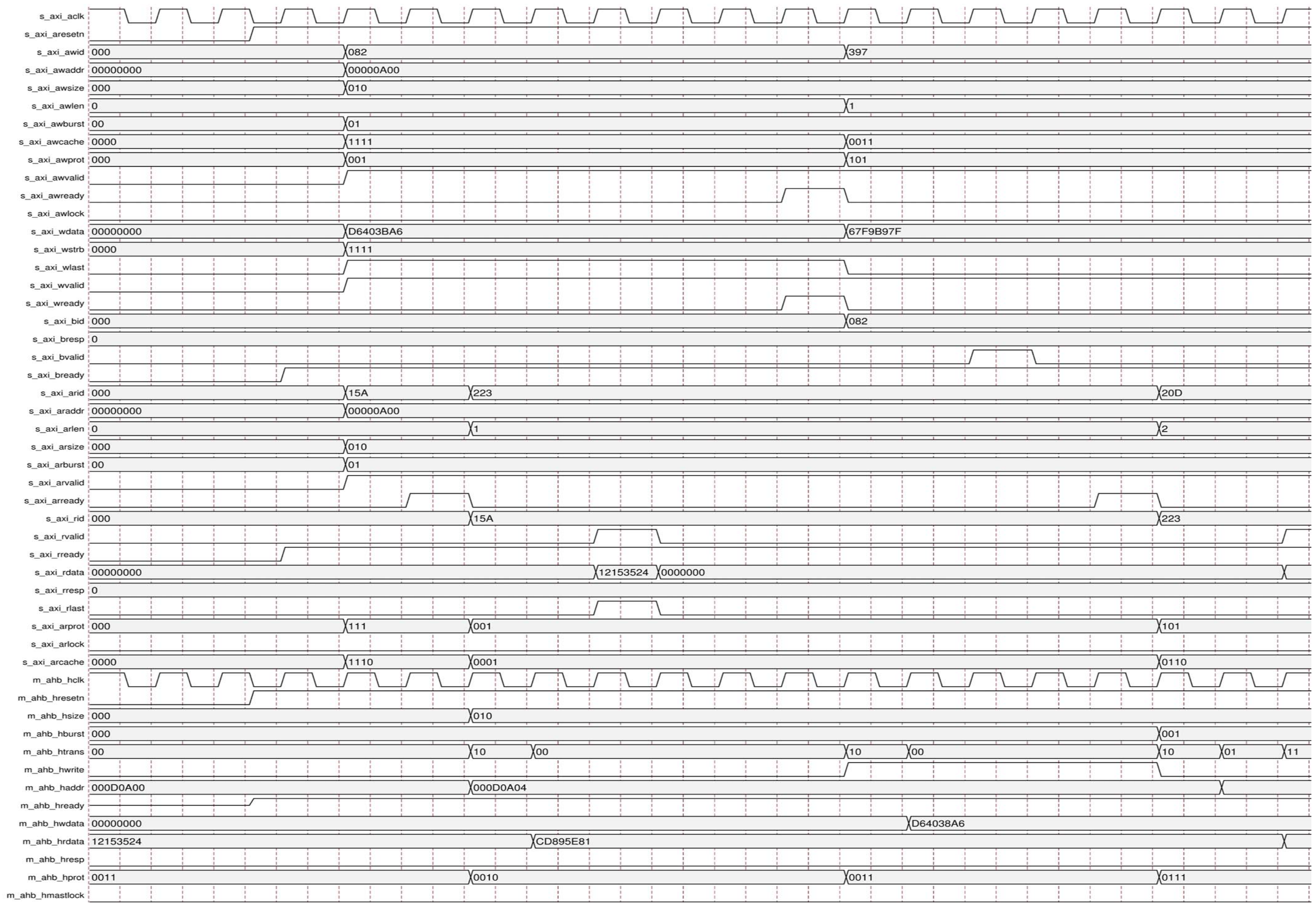


Figure 3-3: **AXI4 Read and Write Transfers**

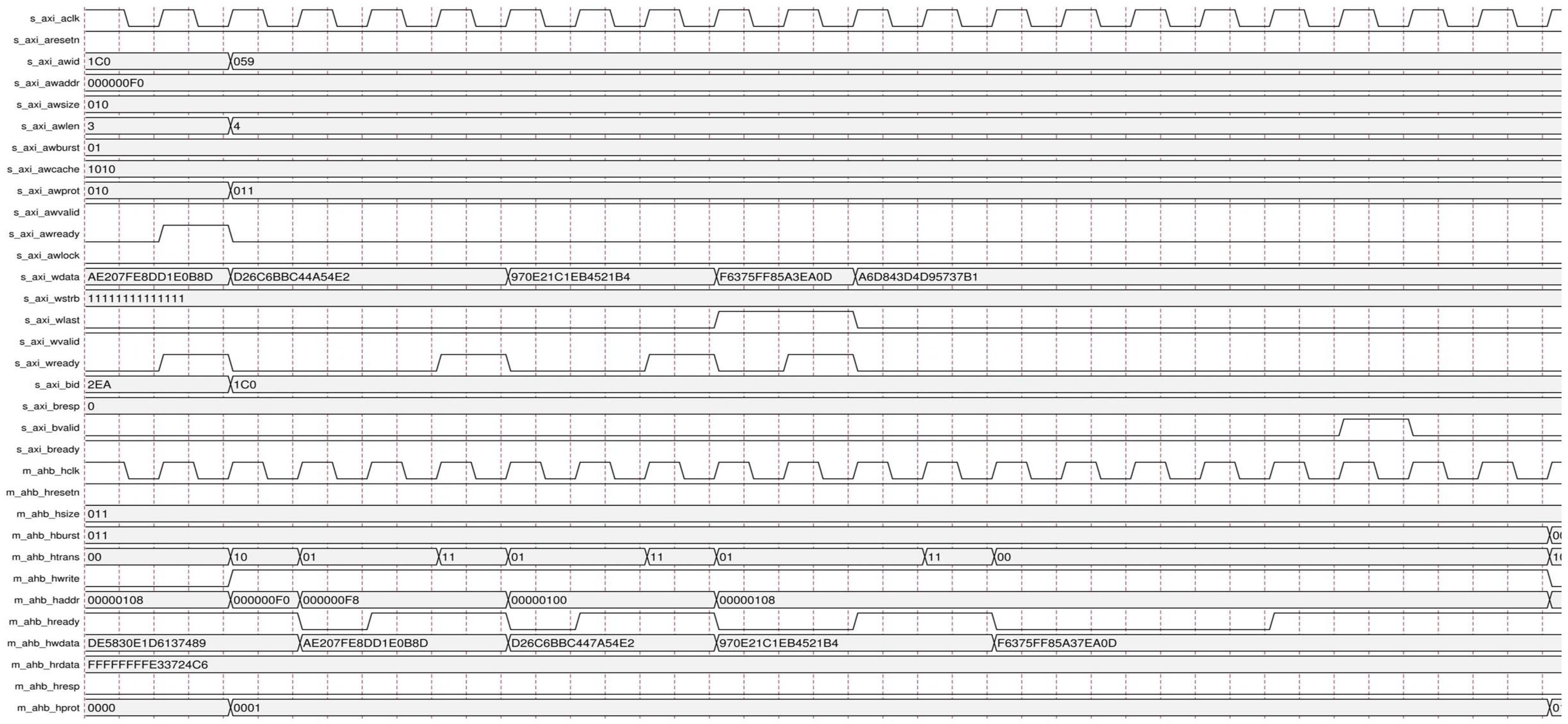


Figure 3-4: **AXI4 Incrementing Burst Write Transfer of Length 4**

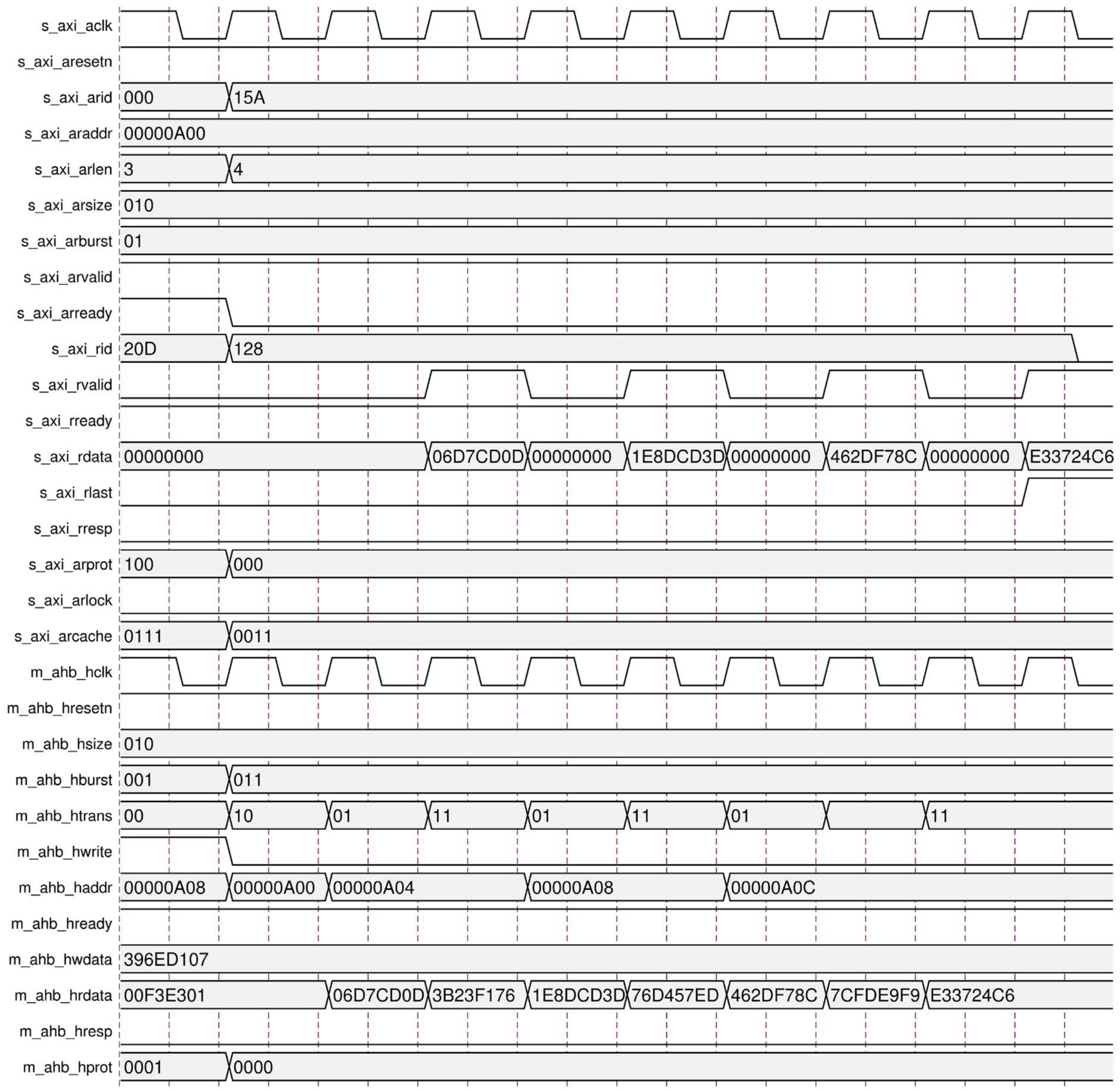


Figure 3-5: **AXI4 Incrementing Burst Read Transfer of Length 4**

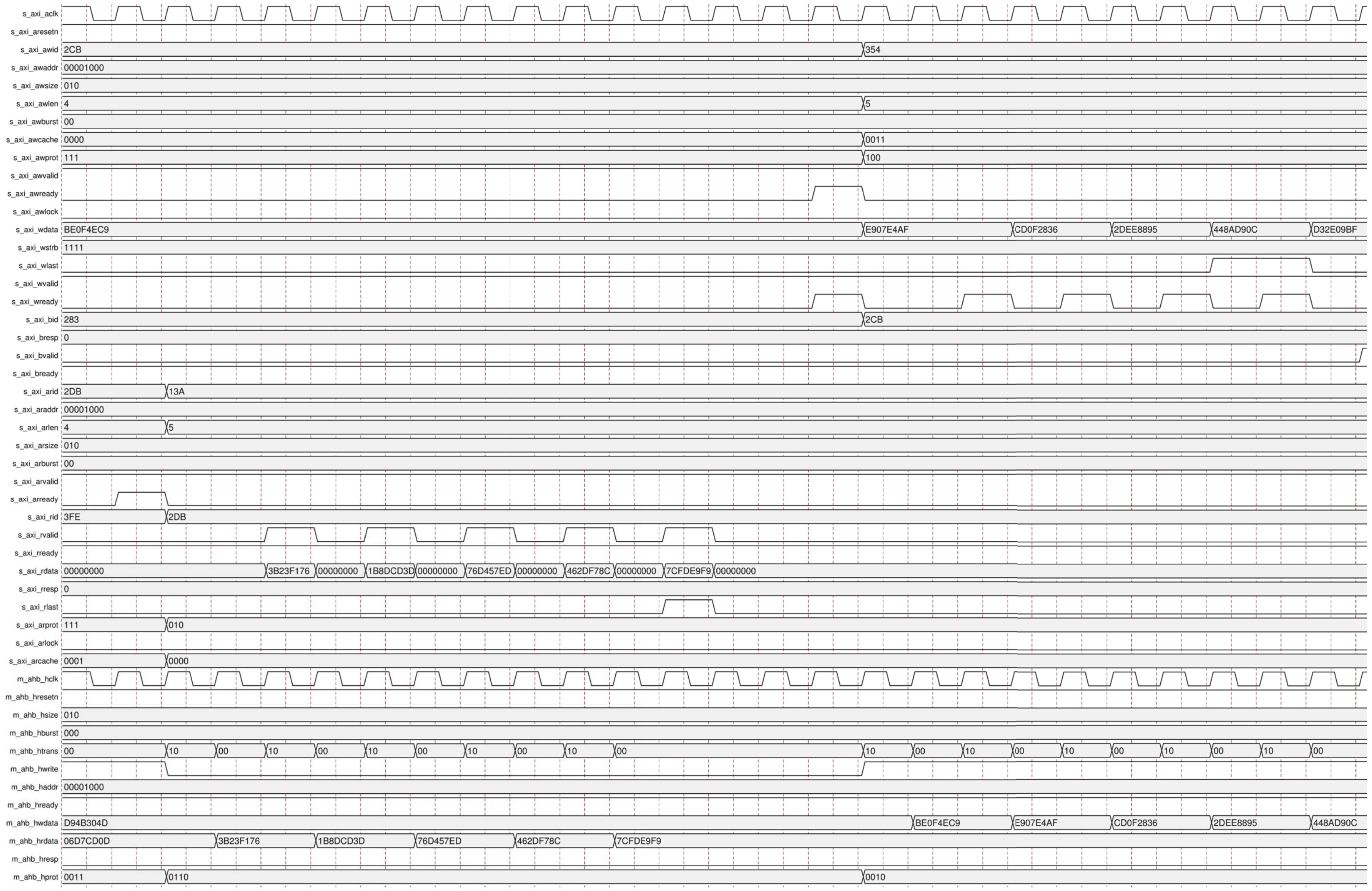


Figure 3-6: **AXI4 Fixed Burst Read and Write Transfers**

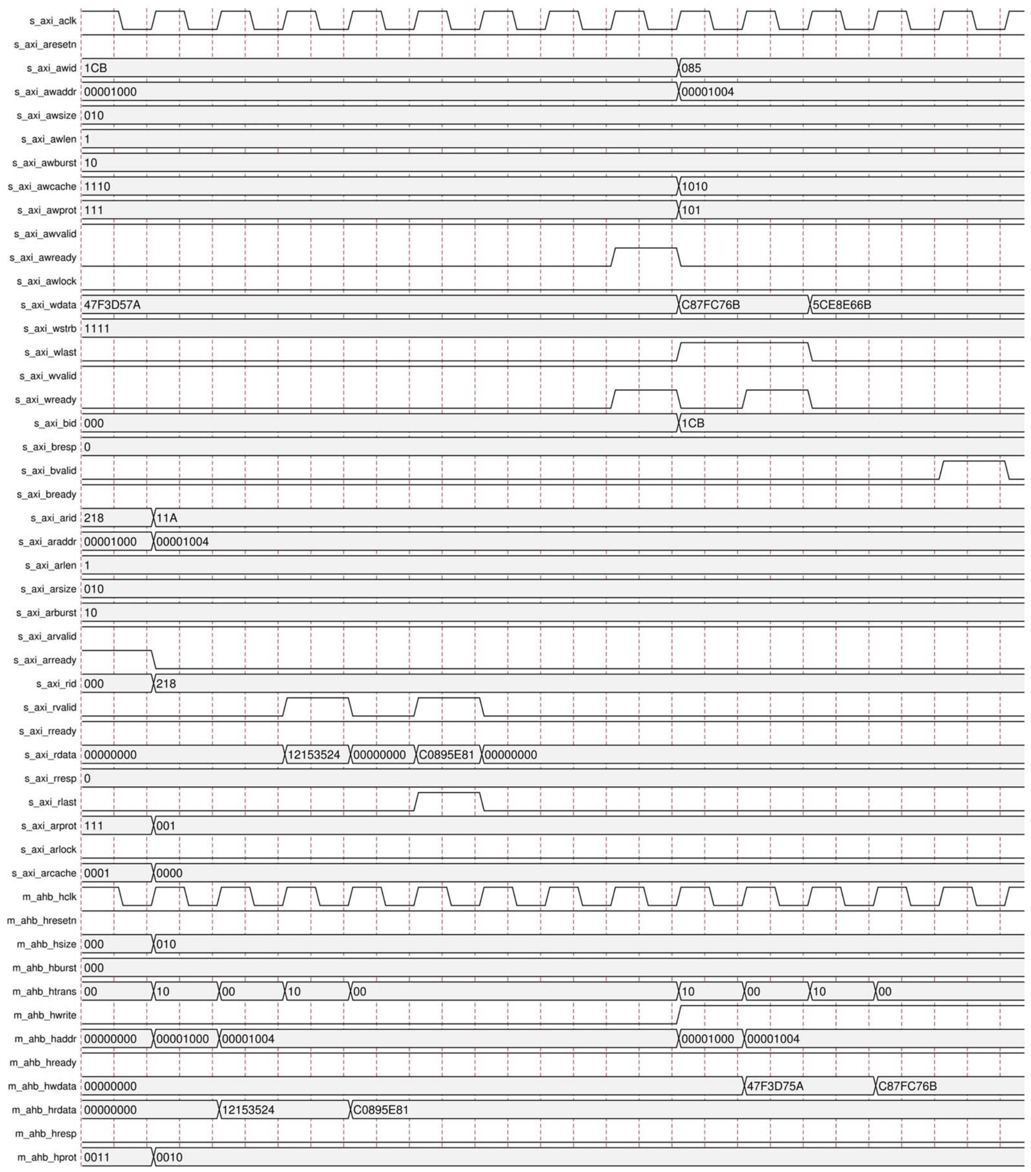


Figure 3-7: **AXI4 Wrapping Burst Read and Write Transfer of Length 2**

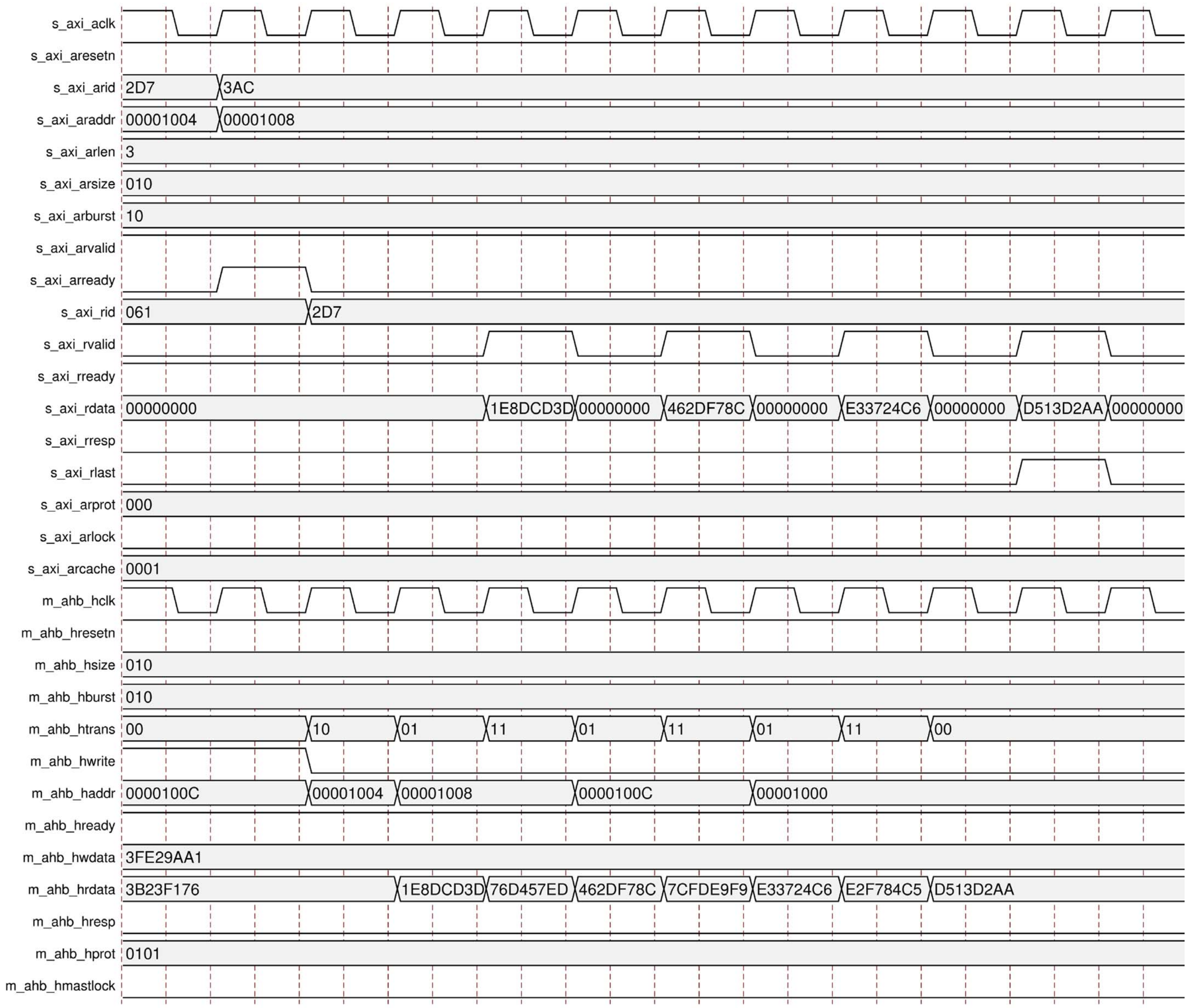


Figure 3-8: **AXI4 Wrapping Burst Read Transfer of Length 4**

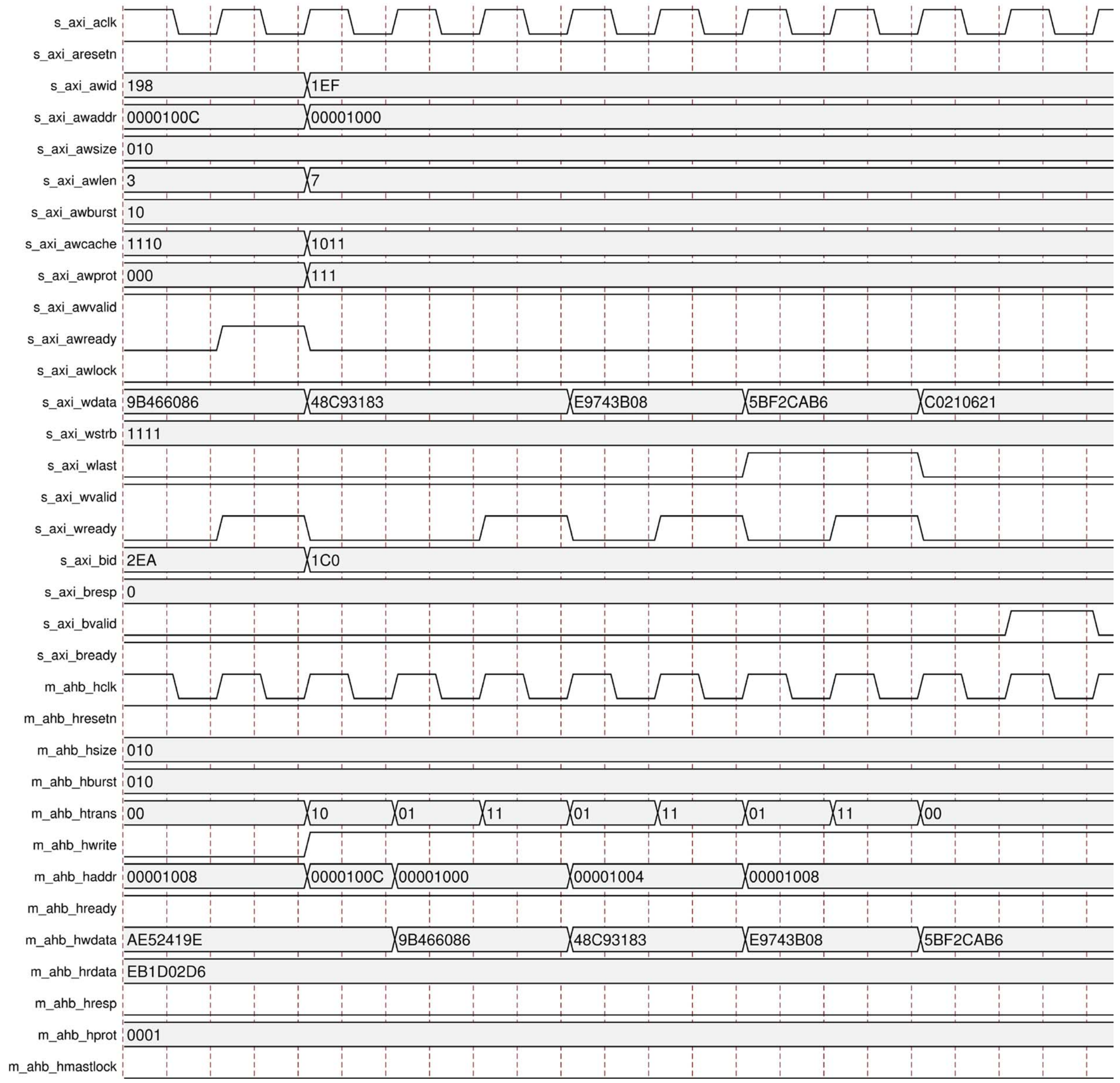


Figure 3-9: **AXI4 Wrapping Burst Write Transfer of Length 4**

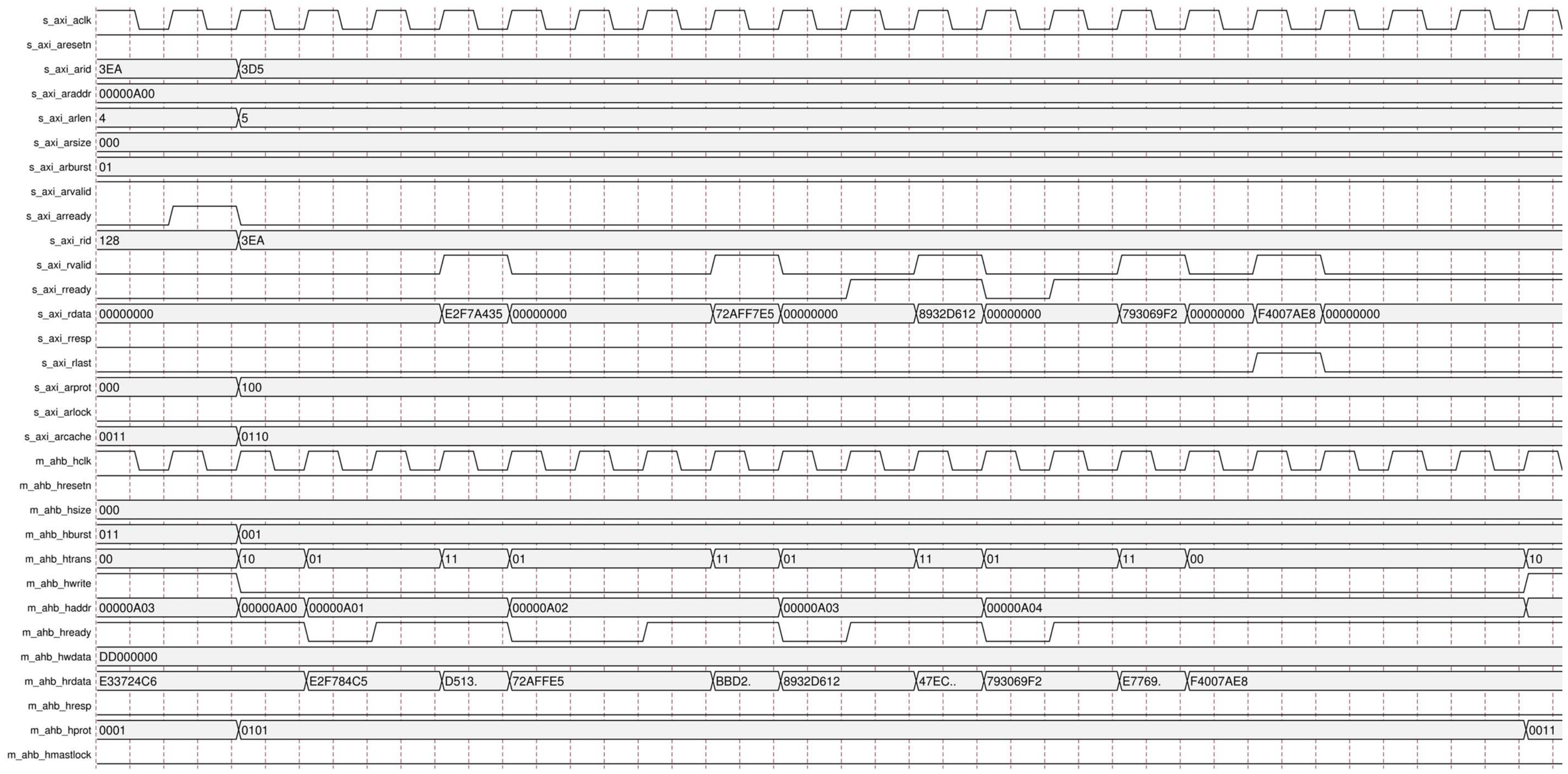


Figure 3-10: **AXI4 8-bit Narrow Read Transfer on 32-bit Data Bus**

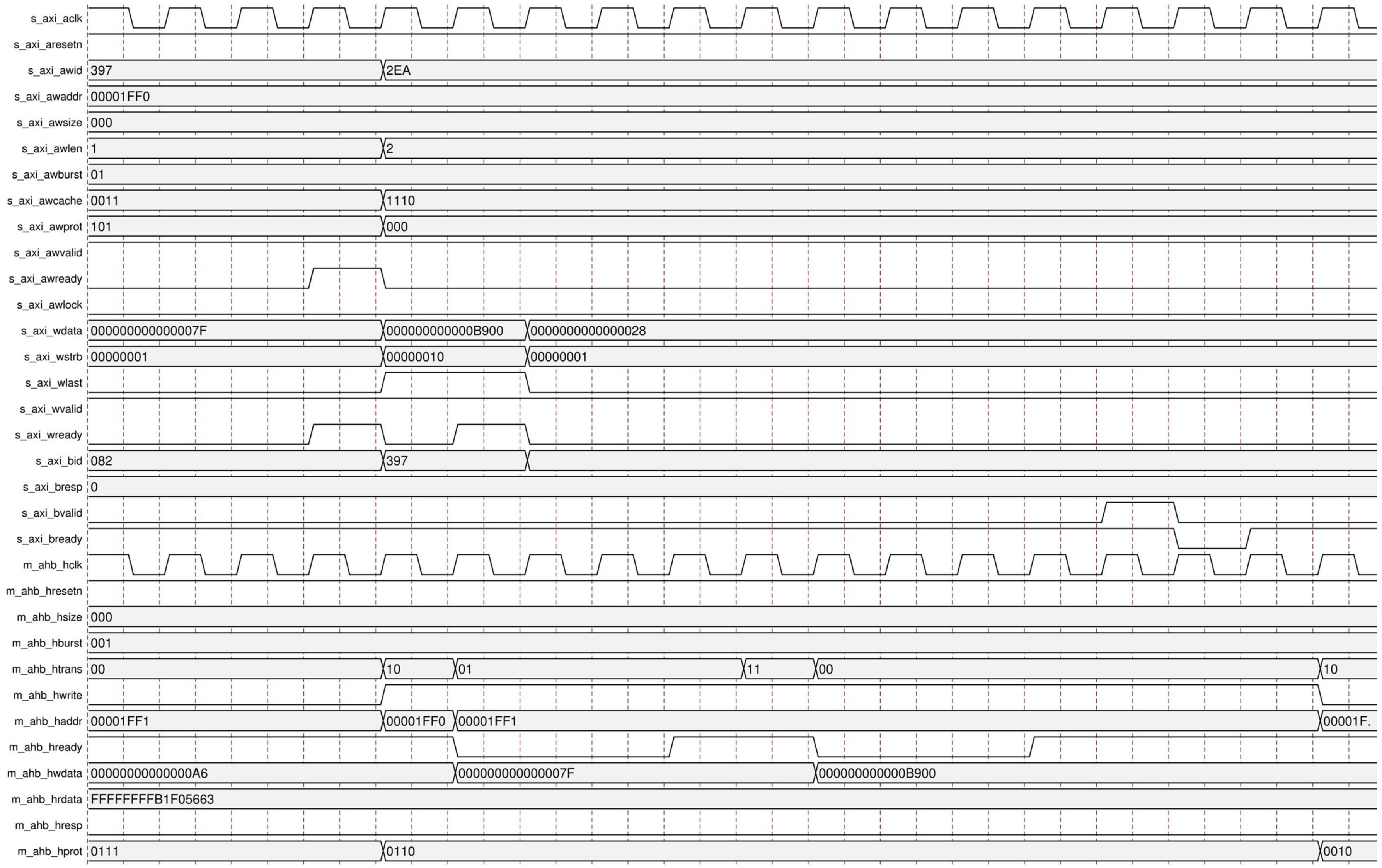


Figure 3-11: **AXI4 8-bit Narrow Write Transfer on 64-bit Data Bus**

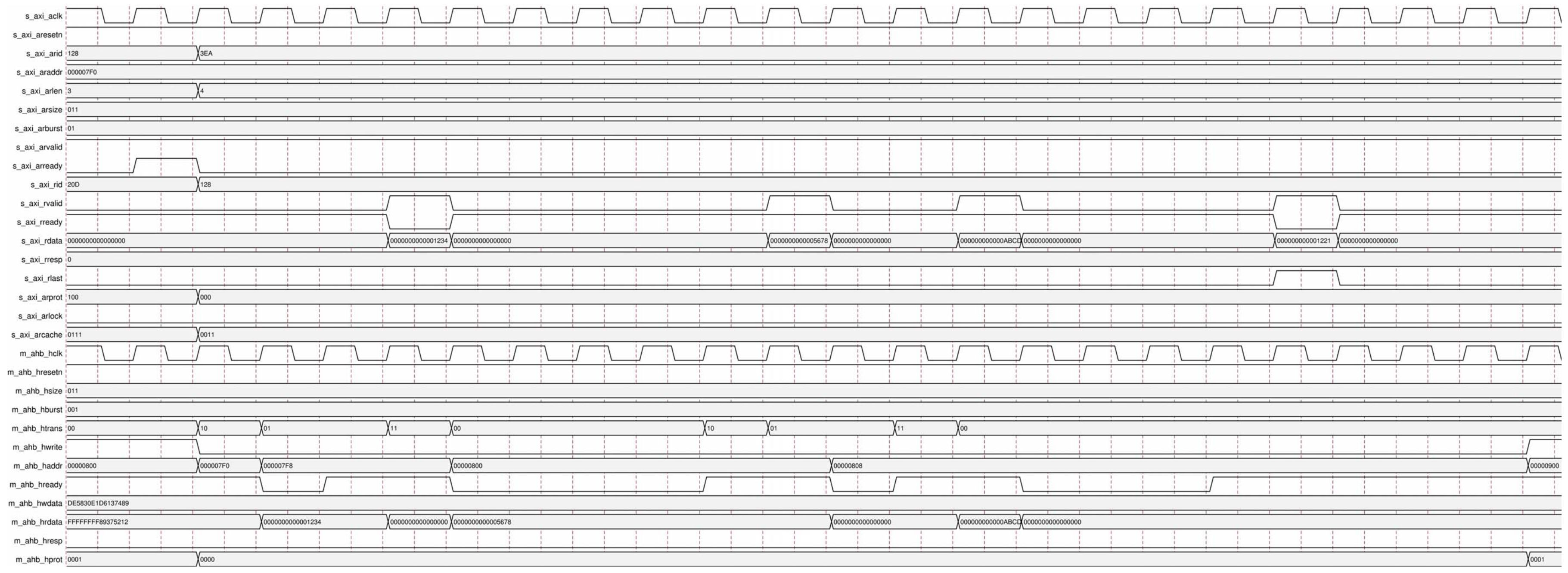


Figure 3-12: **AXI4 Burst Read Transfer of Length 4 That Crosses 1 KB Address Boundary on AHB-Lite**

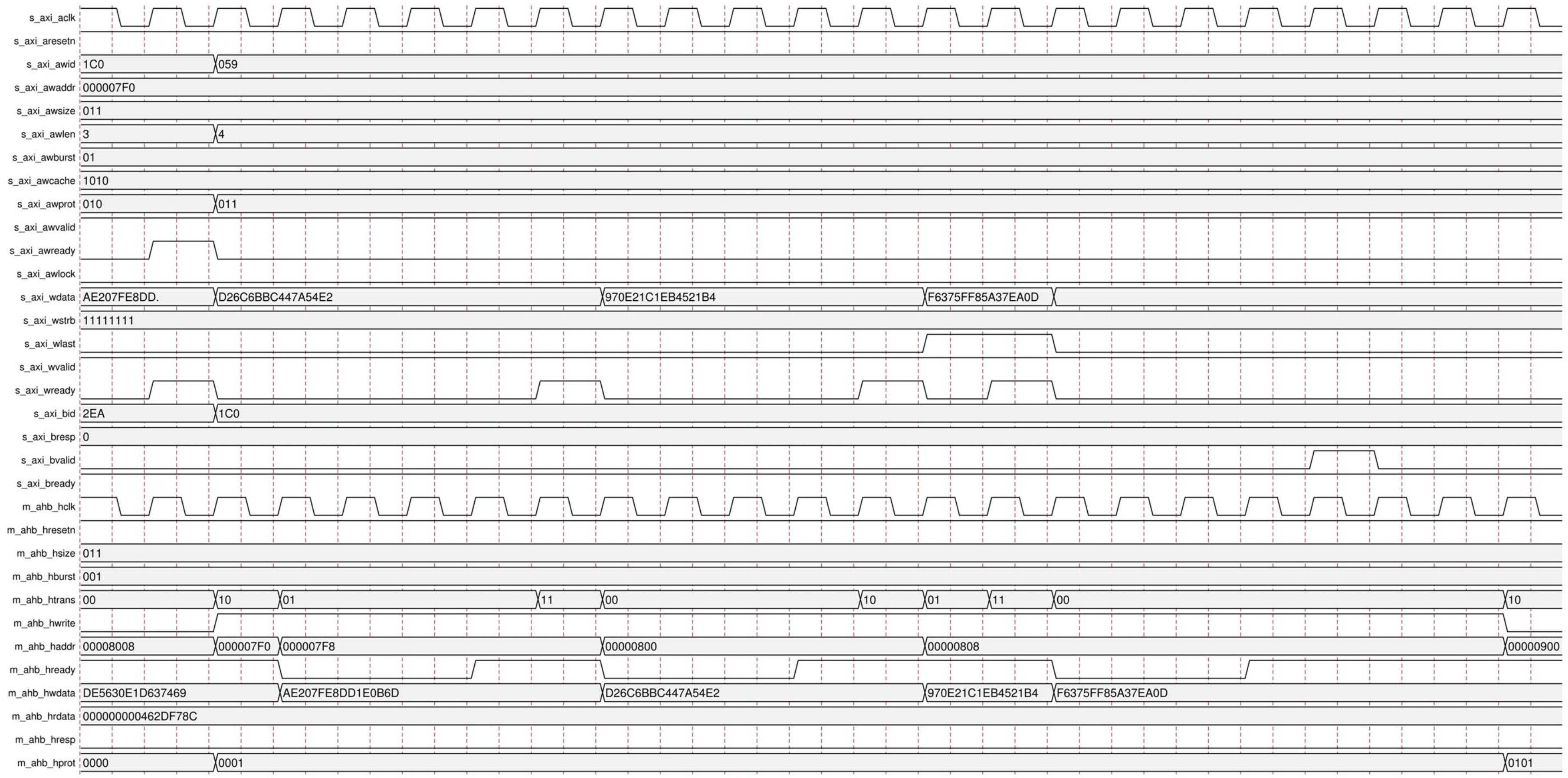


Figure 3-13: **AXI4 Burst Write Transfer of Length 4 That Crosses 1 KB Address Boundary on AHB-Lite**

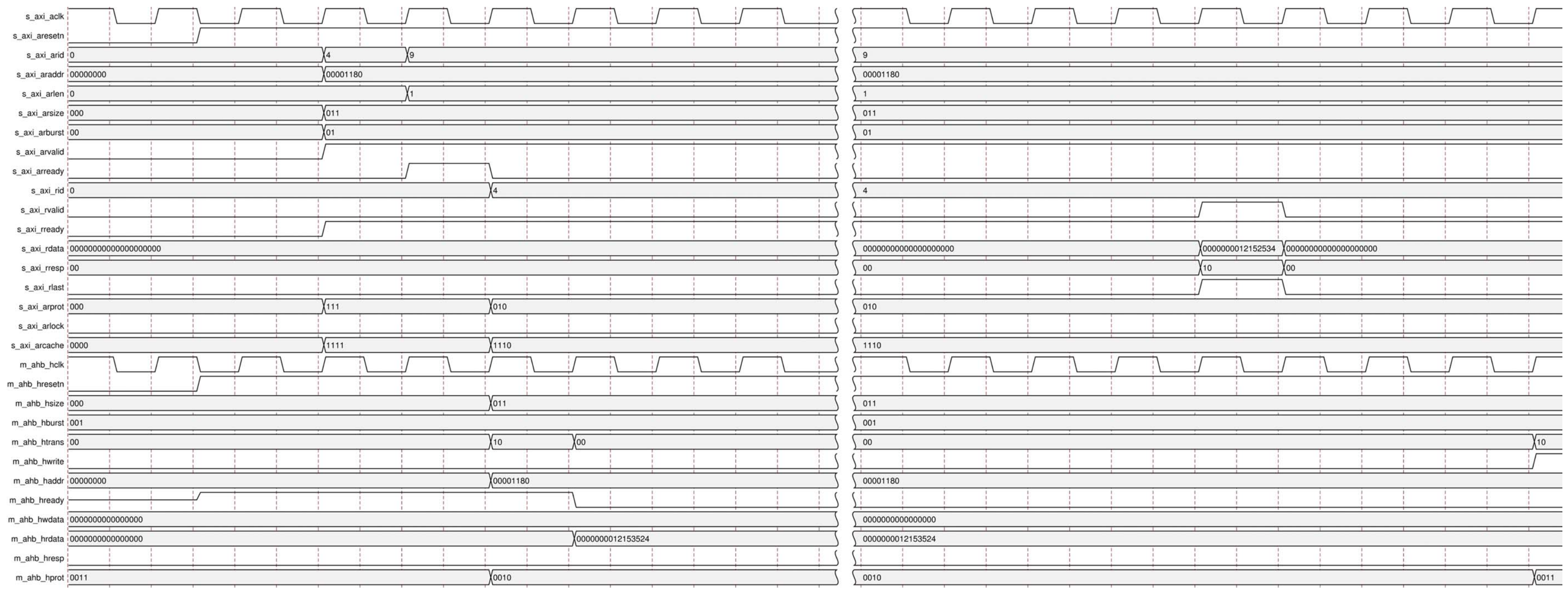


Figure 3-14: **AXI4 Read Timeout of 16**

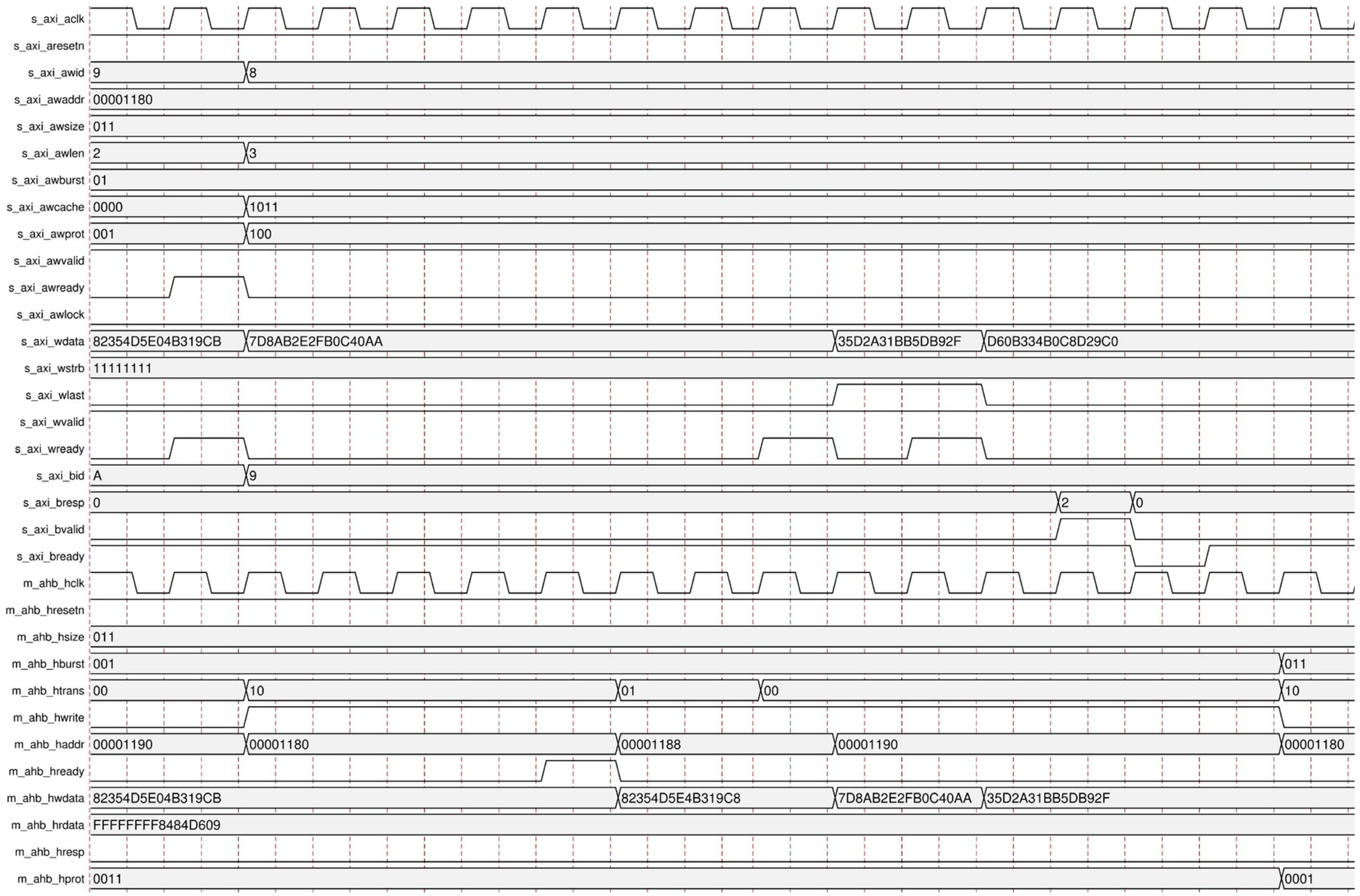


Figure 3-15: **AXI4 Write Timeout of 16**

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the Vivado IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 4]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 6]

Customizing and Generating the Core

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

If you are customizing and generating the core in the IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 4] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

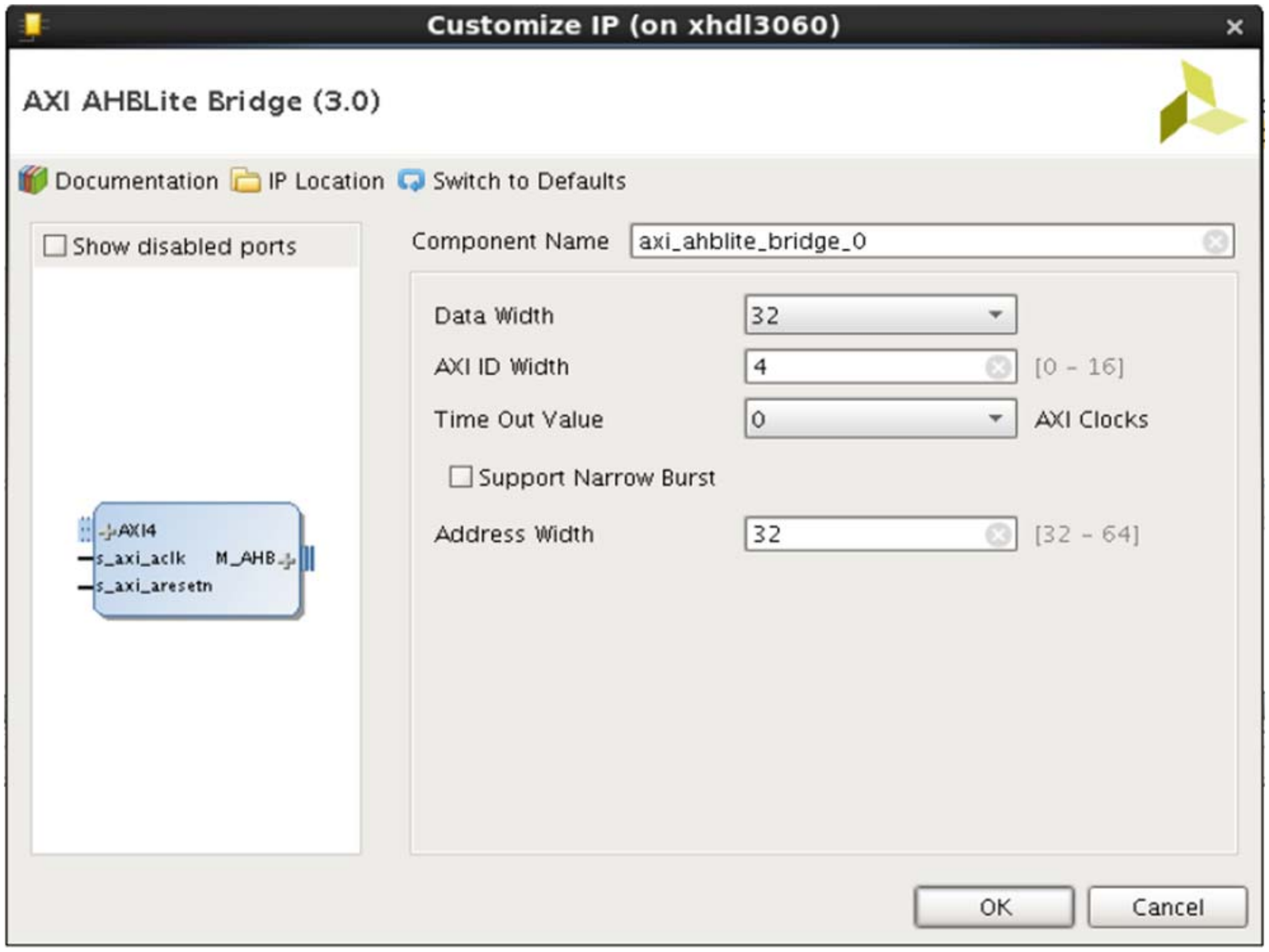
You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the Vivado IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click 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].

Note: Figure in this chapter is an illustration of the Vivado IDE. This layout might vary from the current version.

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



Customize IP (on xhdl3060)

AXI AHBLite Bridge (3.0)

Component Name: axi_ahblite_bridge_0

Data Width: 32

AXI ID Width: 4 [0 - 16]

Time Out Value: 0 AXI Clocks

☐ Support Narrow Burst

Address Width: 32 [32 - 64]

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

- **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 and auto-computed.
- **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.
- **Address Width** – Select width for AHB and AXI4 Address ports, valid values range from 32 to 64.

Output Generation

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

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

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 6].

IMPORTANT: *For cores targeting 7 series or Zynq-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

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 example design provided in the Vivado® Design Suite.

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 5-1. This includes clock generator Clocking wizard IP instantiation and example design module with logic for AXI4 transaction generator and AHB-Lite transaction checker.

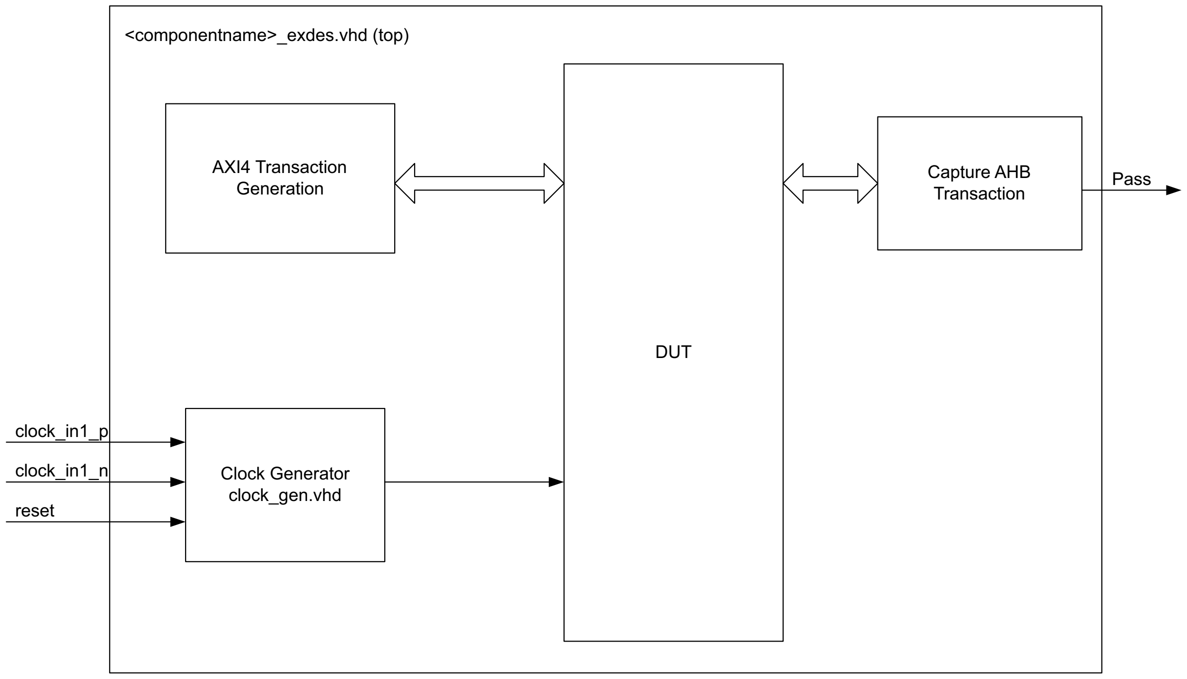

<componentname>_exdes.vhd (top)
AXI4 Transaction Generation
DUT
Capture AHB Transaction
Pass
clock_in1_p
clock_in1_n
reset
Clock Generator clock_gen.vhd

Figure 5-1: **AXI4 to AHB-Lite Bridge Example Design Block Diagram**

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

- **Clock Generator** – Clocking wizard is used to generate the clock for the example design. It generates 100 MHz clock for `s_axi_hclk` of the DUT. The example design DUT is kept under reset until MMCME2 is locked.
- **AXI4 Transaction Generation** – Supports the write and read transactions on the AXI4 interface of the bridge.
- **Capture AHB Transaction** – Serves as the AHB-Lite slave to 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 Customizing and Generating the Core, page 34 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 5-3.
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 5-1 shows the files delivered in the `example_design` directory.

Table 5-1: **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 5-2 shows the files delivered in `simulation` directory.

Table 5-2: **Simulation Directory**

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

Table 5-3 shows the XDC file delivered in `example_design` directory.

Table 5-3: **Constraints Directory**

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

Test Bench

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

Figure 6-1 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.

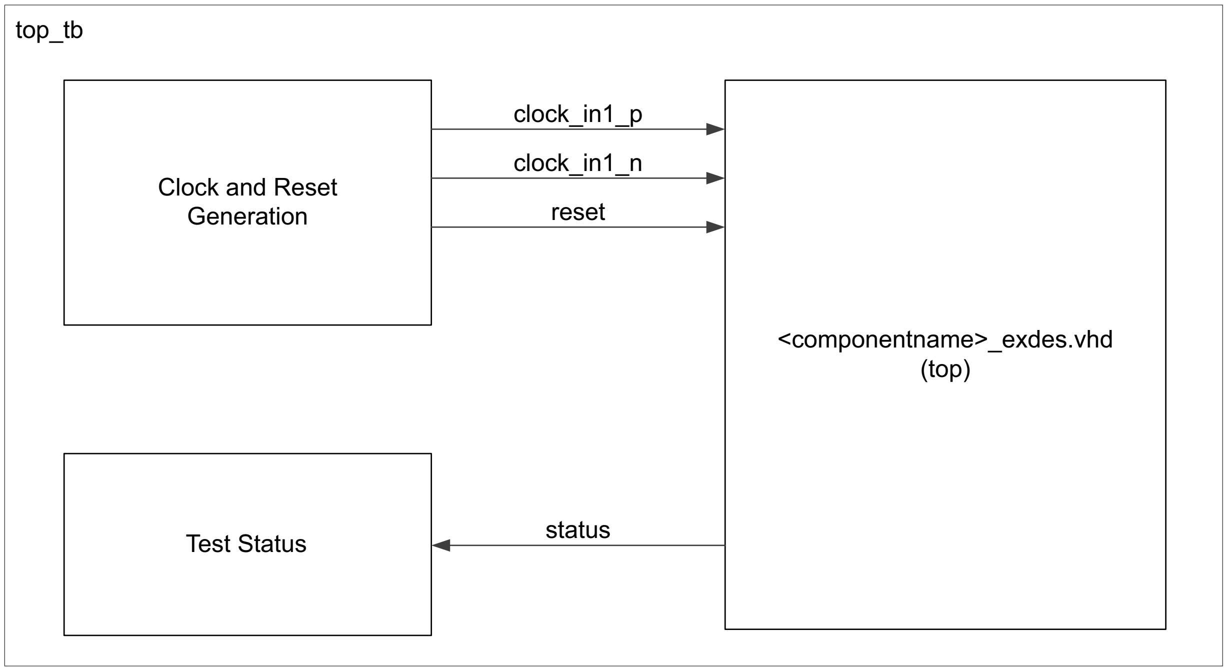


Figure 6-1: **AXI4 to AHB-Lite Bridge Example Design Test Bench**

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 6] 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 AXI4 to AHB-Lite 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.
```

Appendix A

Migrating and Upgrading

This appendix contains information about upgrading to a more recent version of the IP core.

Migrating to the Vivado Design Suite

For information on migrating to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 7].

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations between core versions.

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 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 or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. 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 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

Technical Support

Xilinx provides technical support at Xilinx support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

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 Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The Vivado Design Suite debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used with the logic debug IP cores, including:

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

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 8].

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

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 AXI Reference Guide* (UG1037)
4. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994)
5. *Vivado Design Suite User Guide: Getting Started* (UG910)
6. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
7. *ISE to Vivado Design Suite Migration Guide* (UG911)
8. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
9. *Vivado Design Suite User Guide: Implementation* (UG904)
10. *7 Series FPGAs Overview* (DS180)
11. *LogiCORE IP AXI Interconnect Product Guide* (PG059)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/18/2015	3.0	Added support for UltraScale+ families.
09/30/2015	3.0	• Updated description in AXI4 Slave Interface and AHB-Lite Master Interface sections. • Updated m_ahb_haddr description in I/O Signal Description. • Updated Memory Mapping description. • Updated Fig. 4-1 Customize IP Dialog. • Added Address Width description in Customizing and Generating the Core.
04/02/2014	3.0	• Added description in AHB-Lite Master Interface section. • Updated Table 2-3: I/O Signal Description. • Updated Clocking and Resets sections. • Updated Narrow Transfer section. • Added description to Address/Data Translation section. • Updated Fig. 4-1 Customize IP Dialog.
12/18/2013	2.1	Added UltraScale support.
10/02/2013	2.1	Initial Xilinx public release of document in product guide format. Replaces DS827.

Please Read: Important Legal Notices