AHB-Lite to AXI4 Bridge v3.0

LogiCORE IP Product Guide

Vivado Design Suite

PG176 November 18, 2015

Table of Contents

Introduction

The ARM® AMBA® Advanced High Performance Bus (AHB-Lite) to AXI4 Bridge translates AHB-Lite transactions into AXI4 transactions. It functions as an AHB-Lite slave on the AHB bus and as an AXI4 master on the AXI4 bus.

Features

- AXI4 interface is based on the AXI4 specification
- AHB-Lite interface based on the AHB specification
- AHB and AXI4 data widths are the same and either 32/64-bit based on the configuration
- AHB and AXI4 address widths are the same and based on the configuration
- Supports narrow transfers on the AHB interface
- Supports burst termination on the AHB during which dummy transfers are initiated on the AXI4 interface
- Timeout feature to indicate no response from AXI4 slave during write and read transactions

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

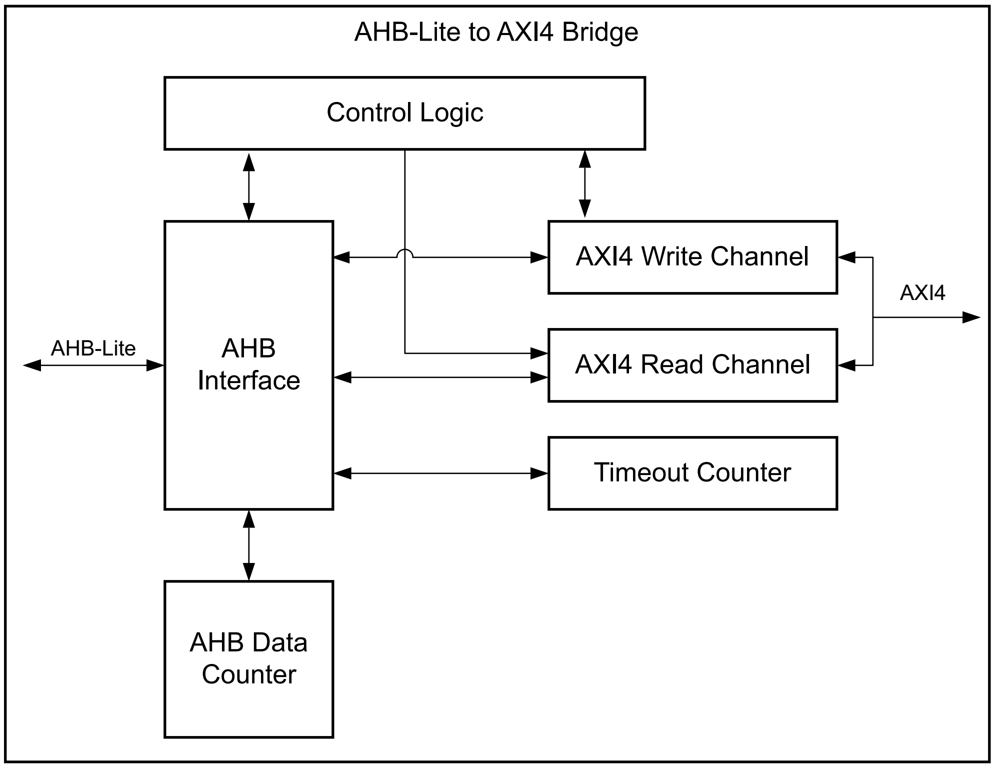


Figure 1-1: **AHB-Lite to AXI4 Bridge Block Diagram**

AHB Interface

The AHB-Lite Slave module provides the AHB-Lite slave interface on the AHB bus side. This module registers the AHB side control signals when a new transfer is initiated. It generates the HREADY based on the current transfer progress on the AXI4 side and burst termination seen on AHB side.

AHB Data Counter

This module counts the valid data received from AHB for the write transfers. The count value is used to limit the HREADY generation after the required number of samples are received for the current transfer.

Control Logic

The control logic state machine is the central controlling unit which instructs the submodules on the progress of the transfer. It detects the characteristics of the transfer initiated on the AHB side (Read/Write, Burst, Single) and instructs the submodules to map the current AHB transfer to AXI4 transfer appropriately.

AXI4 Write Channel

The AXI4 Write channel controls the AXI4 Write transaction channels (Write Address channel, Write Data channel, and Write response channel) based on the instruction from Control logic. The Write strobes are appropriately controlled during narrow transfers and burst termination on the AHB as per the protocol. The AXI4 Write channel completes the transfers with dummy data if required, when burst-termination is seen on the corresponding AHB transfer.

AXI4 Read Channel

The AXI4 Read channel controls the AXI4 Read transaction channels (Read Address channel and Read Data channel) based on the instruction from Control logic. As per the protocol, the AXI4 Read channel completes the transfers by discarding the data if required when burst-termination is seen on the corresponding AHB transfer.

Timeout Module

The timeout module generates the timeout when the AXI4 slave is not responding to the AHB transaction. This is parameterized and generates the timeout only when that parameter value is non-zero. If AXI4 slave is not responding, timeout module waits on the number of AHB clocks specified through the timeout parameter for the AXI4 slave response and then generates the timeout.

Unsupported Features

AHB Slave Interface

- Greater than 64-bit data width.
- SPLIT and RETRY responses.

AXI4 Master Interface

- Greater than 64-bit data width
- Register and posted write implementation
- Fixed address burst (as there is no such transaction in AHB)
- Locked, Barrier, trust zone, exclusive operations
- Out-of-order transaction completion
- Exclusive accesses initiation

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 core is configured for best possible configuration for calculation of read latency. The read latency from read address valid (`s_ahb_htrans` = NONSEQ) to the data beat (`s_ahb_hready`) of AHB-Lite to AXI4 Bridge is four clock cycles.

The core is configured for best possible configuration for calculation of write latency. The write latency from write address valid (`s_ahb_htrans` = NONSEQ) to the write response (`s_ahb_hready`) of AHB-Lite to AXI4 Bridge is five 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**	**Bridge Timeout**	**Slices**	**Slice Flip- Flops**	**LUTs**
0	32	0	77	182	147
0	32	16	78	188	158
0	32	32	75	189	159
0	32	64	77	190	160
0	32	128	102	286	224
0	32	256	78	192	162
0	64	0	109	294	225
0	64	16	109	295	226
0	64	32	100	285	210
0	64	64	78	188	158
0	64	128	75	189	159
0	64	256	77	190	160
1	32	0	75	187	158
1	32	16	74	193	177
1	32	32	81	194	170
1	32	64	78	195	179

Table 2-2: **Device Utilization – 7 Series FPGAs *(Cont'd)***

Parameter Values			Device Resources		
Supports Narrow Burst	**Data Width**	**Bridge Timeout**	**Slices**	**Slice Flip- Flops**	**LUTs**
1	32	128	79	196	180
1	32	256	77	197	181
1	64	0	102	286	224
1	64	16	109	293	232
1	64	32	109	294	225
1	64	64	109	295	226
1	64	128	111	296	227
1	64	256	110	297	228

Port Descriptions

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

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

Signal Name	Interface	I/O	Initial State	Description
AXI4 Channel Signals				
m_axi*	AXI4	I	–	See Appendix A of the *Vivado AXI Reference Guide* (UG1037) [Ref 3] for a description of AXI4 signals.
AHB-Lite Signals				
s_ahb_hclk	AHB	I	–	AHB Clock.
s_ahb_hresetn	AHB	I	–	AHB active-Low reset.
s_ahb_hsel	AHB	I	–	Slave select signal for AHB interface.
s_ahb_haddr	AHB	I	–	AHB Address bus, width depends on address width Vivado IDE configuration.
s_ahb_hprot[3:0]	AHB	I	–	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.
s_ahb_htrans[1:0]	AHB	I	–	AHB Transfer Type (NONSEQUENTIAL, SEQUENTIAL, IDLE, or BUSY.)
s_ahb_hsize[2:0]	AHB	I	–	Indicates the size of the transfer.
s_ahb_hwrite	AHB	I	–	Direction. Indicates an AHB write access when High and an AHB read access when Low.
s_ahb_hburst[2:0]	AHB	I	–	Burst type. Indicates if the transfer forms part of the burst.

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

Signal Name	**Interface**	**I/O**	**Initial State**	**Description**
s_ahb_hwdata	AHB	I	–	Write data. The write data bus is used to transfer data from the master to the bus slaves during write operations.
s_ahb_hready_out	AHB	O	1	Transfer done. The AHB slave uses this signal to extend an AHB transfer.
s_ahb_hready_in	AHB	I	–	HREADY input signals from interconnect.
s_ahb_hrdata	AHB	O	0	Read Data. The slave drives this bus during read cycles when `s_ahb_hwrite` is Low. Width of the bus can be 32 or 64.
s_ahb_hresp	AHB	O	0	Provides information on the status of the transfer.

IMPORTANT: *In the case of single slave system where AHB multiplexer is not used, HREADY_OUT has to be looped back and connected to the HREADY_IN port for correct system operation.*

Register Space

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

Chapter 3

Designing with the Core

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

Clocking

The AHB-Lite to AXI4 Bridge is a synchronous design and uses the `s_ahb_hclk` at both AHB-Lite and AXI4 interfaces.

Resets

`s_ahb_hresetn` is a synchronous active-Low reset input that resets the AHB-Lite to AXI4 Bridge upon assertion. `s_ahb_hresetn` is also used to reset the AXI4 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 AHB-Lite to AXI4 Bridge does not modify the address for AXI4. Hence, the address that is presented on the AXI4 is exactly as received on the AHB-Lite.

Data Width

The AHB-Lite to AXI4 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 AHB-Lite to AXI4 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 and 64 respectively.

AHB-Lite Response Signaling

`OKAY` and `ERROR` responses are supported. `SPLIT` and `RETRY` responses are not supported. `SLVERR`, `DECERR`, and timeout error (if used) from AXI4 interface are mapped to `ERROR` response on the AHB-Lite interface.

AXI4 Response Signaling

The responses expected from AXI4 slave are `OKAY`, `SLVERR`, and `DECERR`. `EXOKAY` response is not expected as exclusive accesses requests are not initiated by the AXI4 master interface.

Endianness

Both AHB-Lite and AXI4 interfaces are little endian.

Address/Data Translation

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

Data Width Selection on AHB and AXI4 Interface

The data width of the AHB and AXI4 interfaces are selected to the same value either 32 or 64 bits.

Bridge Timeout Condition

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

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

- When Timeout is selected and the timeout condition occurs, the transaction on the AXI4 side is closed as follows:
  - For write transactions AWVALID/WVALID are deasserted without waiting for AWREADY/WREADY. If the write transaction is in the response phase BREADY is deasserted.
  - For a read transaction ARREADY/READY are deasserted.
- Because the bridge gives the same ERROR response for slave error cases and timeout cases, it is mandatory to provide a reset to the bridge after the `ERROR` response is detected from the bridge when the timeout logic is activated.
- When the timeout logic is not activated, it is not required to give reset to the bridge when an `ERROR` response is seen for a particular transaction.

Transaction Mapping from AHB-Lite to AXI4 Transaction

The different possible transactions from AHB-Lite interface to AXI4 interface are mapped in Table 3-1.

Table 3-1: **Transaction Mapping from AHB-Lite to AXI4 Transaction**

AHB-Lite Transaction	**AXI4 Transaction**		**Description**
HBURST	**AWBURST/ ARBURST**	**AWLEN/ ARLEN**	
SINGLE	INCR	0	Single transfers on AHB are converted to INCR of length 0.
INCR	INCR	0[1]	Indefinite length increment transfers on AHB are converted to INCR of length 0.
WRAP4	WRAP	3	WRAP4 transfer on AHB is converted to WRAP transfer of length 3 on AXI4 side.
INCR4	INCR	3	INCR4 transfer on AHB is converted to INCR transfer of length 3 on AXI4 side.
WRAP8	WRAP	7	WRAP8 transfer on AHB is converted to WRAP transfer of length 7 on AXI4 side.
INCR8	INCR	7	INCR8 transfer on AHB is converted to INCR transfer of length 7 on AXI4 side.
WRAP16	WRAP	15	WRAP16 transfer on AHB is converted to WRAP transfer of length 15 on AXI4 side.
INCR16	INCR	15	INCR16 transfer on AHB is converted to INCR transfer of length 15 on AXI4 side.

Notes:

1. Infinite length of INCR transfers from the AHB-Lite interface are initiated as INCR transactions of burst length 0. Each transaction on AHB is initiated as separate transfer on the AXI4 side.

Protection Signal Mapping from AHB-Lite to AXI4 Interface

Table 3-2: **Protection Signal Mapping from AHB-Lite Interface to AXI4 Interface**

AHB Signal		**AXI4 Signal[1]**	
Signal	**Value**	**Signal (AR/AW)**	**Value**
HPROT[0]	0	PROT[2]	1
HPROT[0]	1	PROT[2]	0
HPROT[1]	0	PROT[0]	0
HPROT[1]	1	PROT[0]	1
HPROT[2]	0	CACHE[0]	0

Table 3-2: **Protection Signal Mapping from AHB-Lite Interface to AXI4 Interface *(Cont'd)***

AHB Signal		**AXI4 Signal**[1]	
Signal	**Value**	**Signal (AR/AW)**	**Value**
HPROT[2]	1	CACHE[0]	1

Notes:

1. Values which are not mapped takes the default value for outputs. Input values which are not considered for mapping are ignored during mapping and outputs are set to default values for such inputs.

Timing Diagrams

The timing diagram shown in the subsequent figures illustrate the operation for various read and write transfers.

Note: `m_axi_aclk` is tied to `s_ahb_hclk`. You can refer to `s_ahb_hclk` for AXI4 signals.

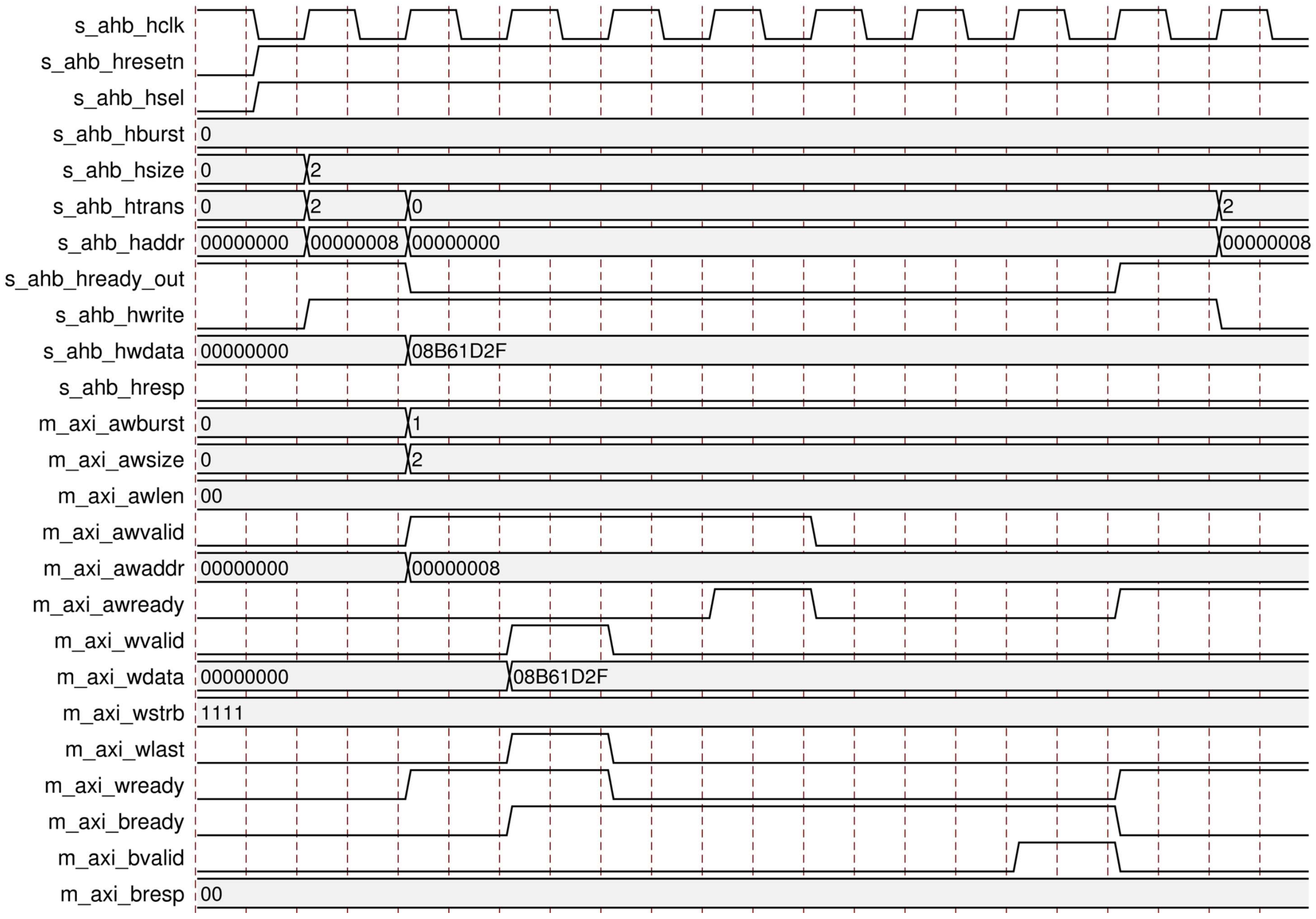


Figure 3-1: **Write Transfer with Burst Type SINGLE**

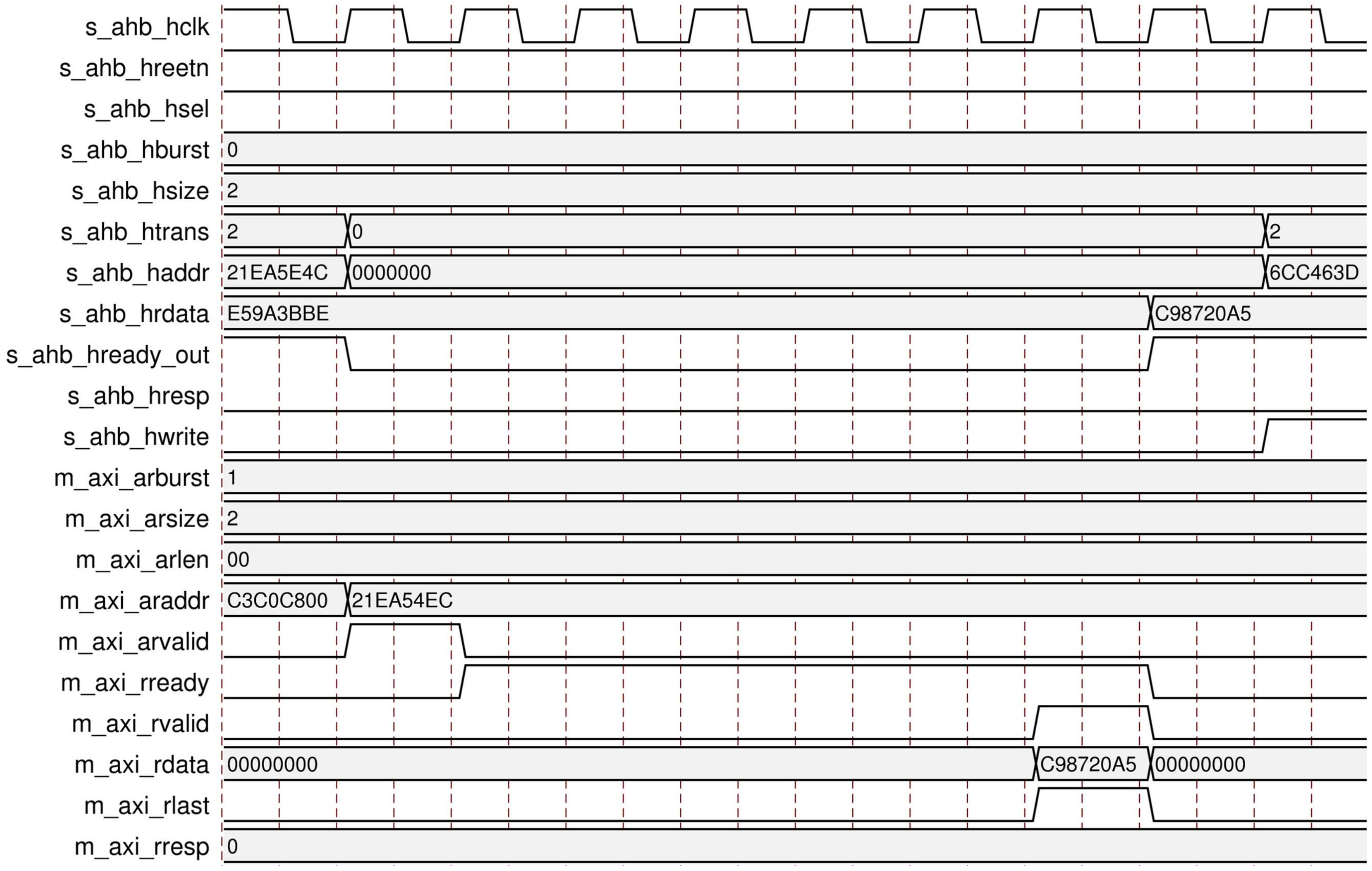


Figure 3-2: **Read Transfer with Burst Type SINGLE**

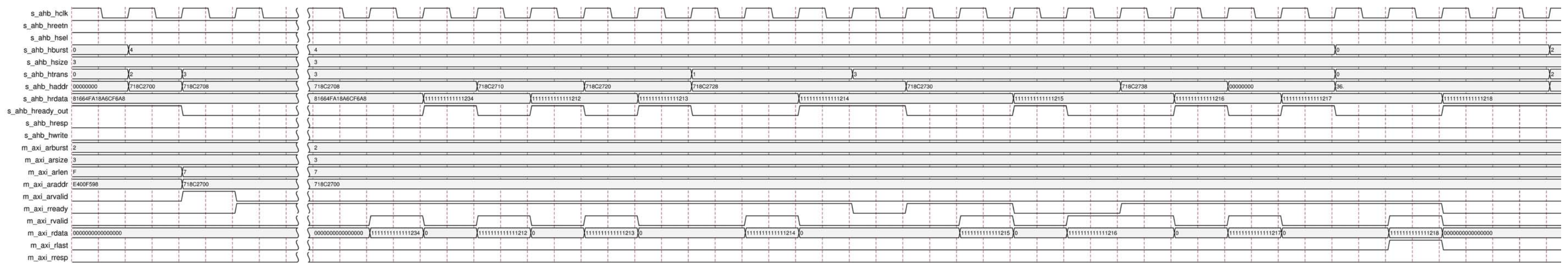


Figure 3-3: **Read Transfer with Burst Type WRAP8**

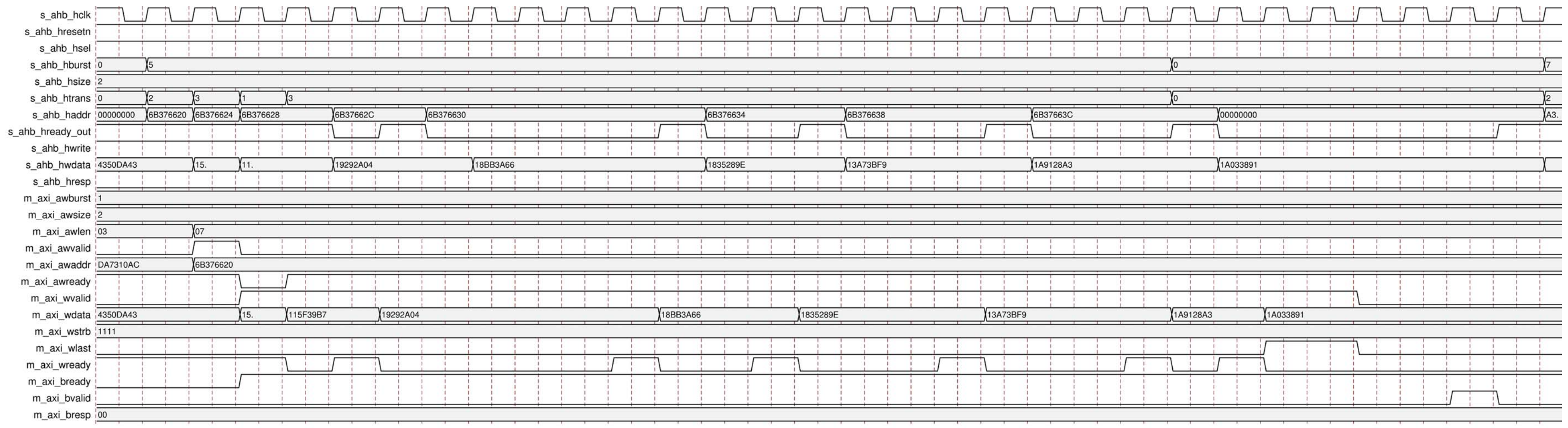


Figure 3-4: **Write Transfer with Burst Type INCR8**

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.

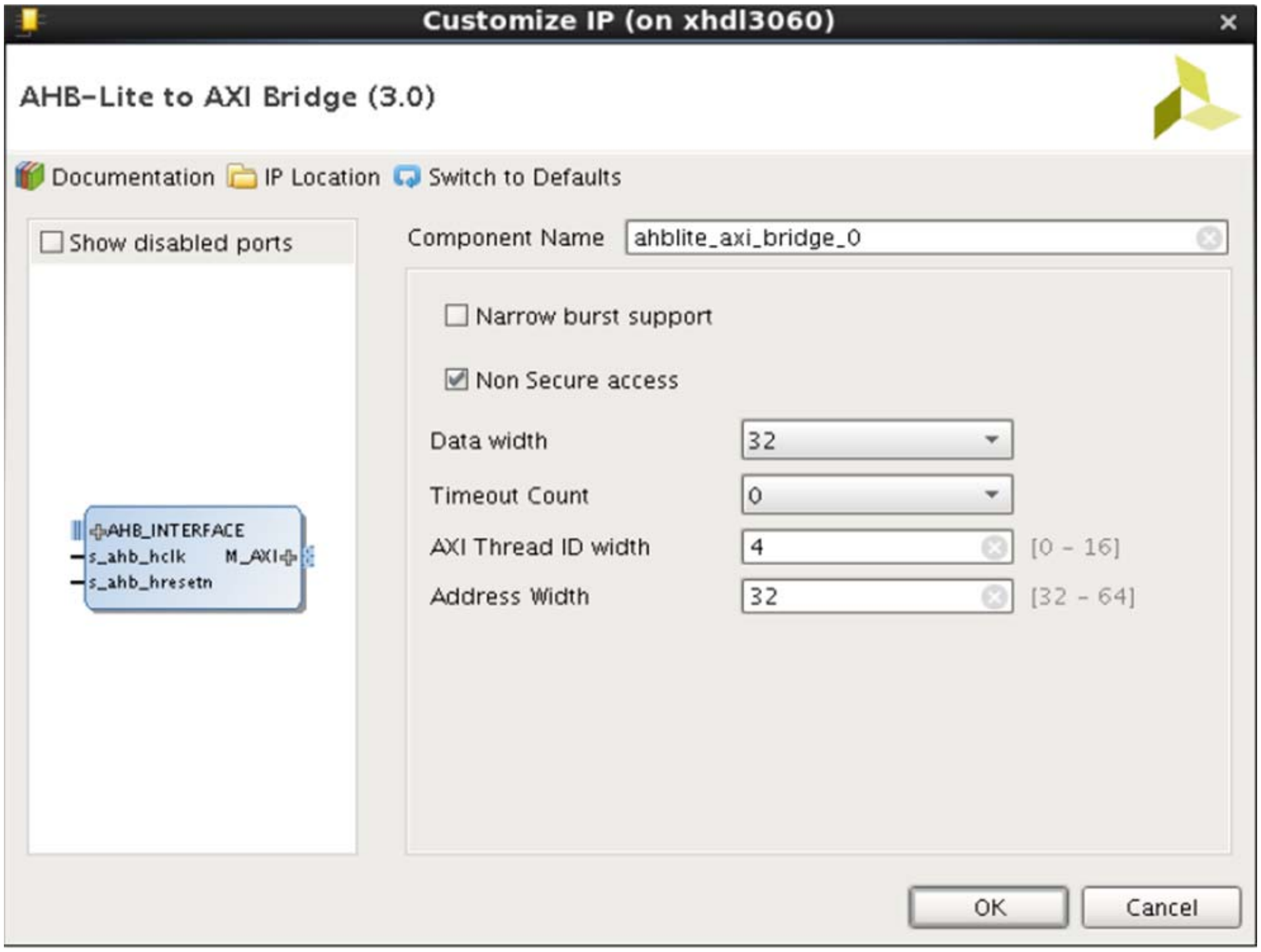
You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

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 AHB-Lite to AXI4 Bridge IP core.


Customize IP (on xhdl3060)

AHB-Lite to AXI Bridge (3.0)

Component Name: ahblite_axi_bridge_0

☐ Narrow burst support

☑ Non Secure access

Data width	32	
Timeout Count	0	
AXI Thread ID width	4	[0 - 16]
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 "_".
- **Narrow Burst Support** – Narrow transfer support makes the transactions narrower than data width.
- **Non-Secure Access** – Support secured data transfer at AXI side. To support backward compatibility, default value is True (for example, non-secure data transfer).
- **Data Width** – Select 32 or 64 AHB to AXI4 data width.
- **Timeout Count** – Indicates the number of AHB clocks to wait for AXI4 slave to respond.
- **AXI4 Thread ID Width** – Indicates width of the ID tags.
- **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 AHB-Lite transaction generator and AXI4 transaction checker.

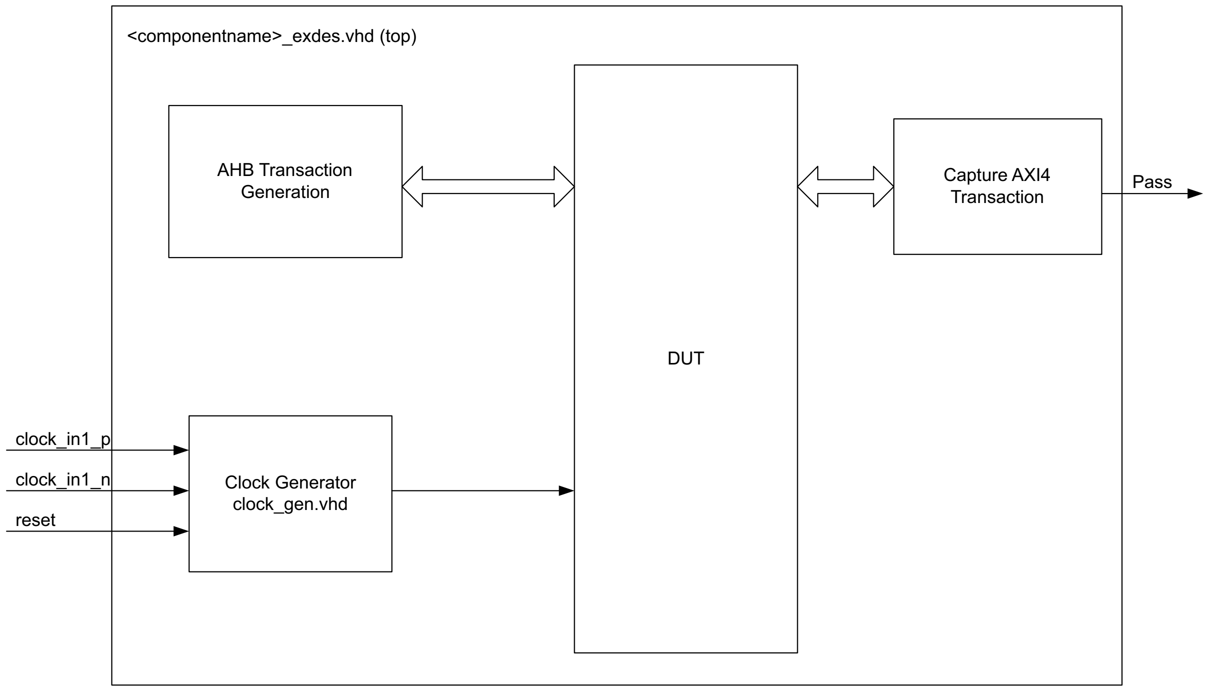

<componentname>_exdes.vhd (top)
AHB Transaction Generation
DUT
Capture AXI4 Transaction
Pass
clock_in1_p
clock_in1_n
reset
Clock Generator clock_gen.vhd

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

This example design demonstrates the transactions on AHB-Lite 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_ahb_hclk` of the DUT. The example design DUT is kept under reset until MMCME2 is locked.
- **AHB Transaction Generation** – Supports the write and read transactions on the AHB-Lite interface of the bridge.

- **Capture AXI4 Transaction** – Serves as the AXI4 slave to bridge and manages write and read transactions from the AXI4 interface of the bridge.

Implementing the Example Design

After following the steps described in Customizing and Generating the Core, page 20 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 AHB-Lite to AXI4 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 AHB-Lite to AXI4 Bridge example design. The top-level test bench generates 200 MHz clock and drives initial reset to the example design.

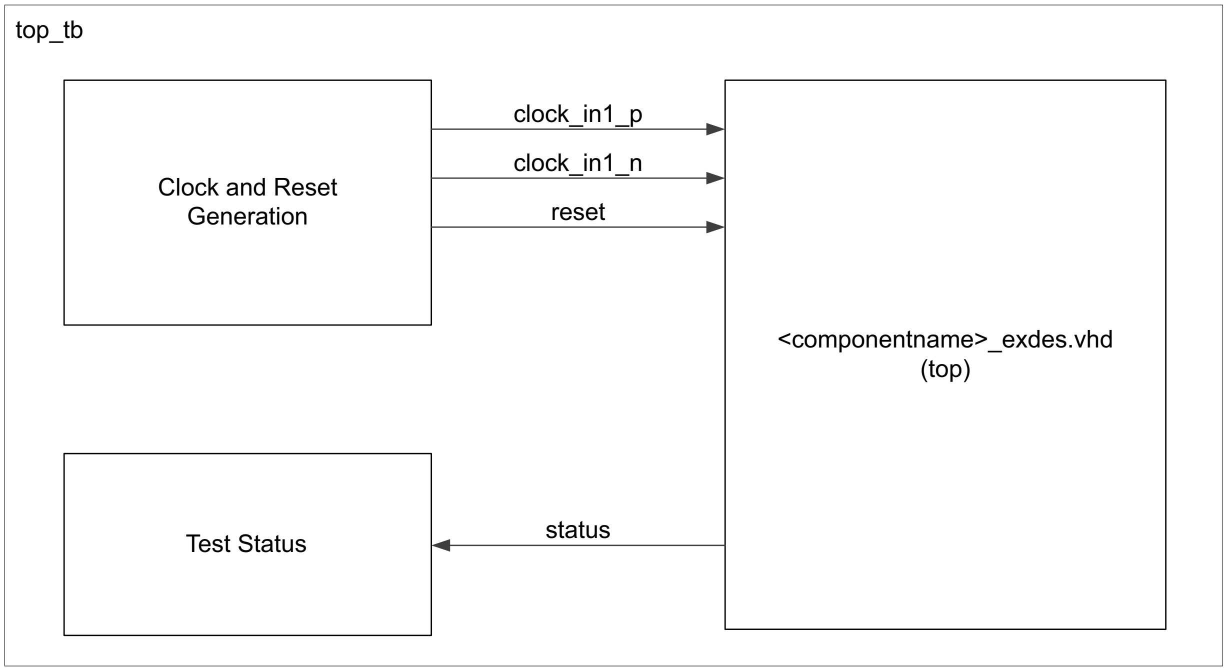


Figure 6-1: **AHB-Lite to AXI4 Bridge Example Design Test Bench**

Simulating the Example Design

Using the AHB-Lite to AXI4 Bridge example design (delivered as part of the AHB-Lite to AXI4 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 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.
```

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 AHB-Lite to AXI4 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 AHB-Lite to AXI4 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 AHB-Lite to AXI4 Bridge

AR: 57791

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 AHB-Lite to AXI4 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	• Added AHB and AXI4 address width description in IP Facts. • Updated m_ahb_haddr description in I/O Signal Description. • Added Important note Port Descriptions section. • 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	• Updated Table 2-3: I/O Signal Description. • Updated Fig. 4-1 Customize IP Dialog box and added Non-Secure Access description.
12/18/2013	2.1	Added UltraScale support.
10/02/2013	2.1	Initial Xilinx public release of document in product guide format. Replaces DS825.

Please Read: Important Legal Notices