AMM Slave Bridge v1.0

LogiCORE IP Product Guide

Vivado Design Suite

PG258 July 15, 2020

Table of Contents

Appendix A: Debugging

Appendix B: Additional Resources and Legal Notices

Introduction

The Xilinx® LogiCORE™ AMM Slave Bridge IP core connects Avalon slaves with AXI interface masters. The IP translates AXI4-Lite and AXI4 interface transactions into Avalon bridge transactions. Hence, it is also referred to as AXI to Avalon Memory Mapped (AMM) Bridge. This IP allows parameter configuration to match Avalon bridge slave interface properties and enables seamless interface with the AXI interface system.

Features

- Supports configurable AXI4-Lite and AXI4 interface
- Supports 32-bit data width for AXI4-Lite interface
- Supports up to 1,024-bit data width for memory mapped AXI interfaces
- Support for fixed and variable wait
- Support for fixed and variable latency
- AXI response generation if no response signal from Avalon slave
- Data phase timeout logic
- Byte and word addressing

LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale+™ families UltraScale™ families Zynq® UltraScale+ MPSoC 7 Series FPGAs
Supported User Interfaces	AXI4, AXI4-Lite, Avalon
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Verilog
Example Design	Verilog
Test Bench	Verilog
Constraints File	Xilinx Design Constraints (XDC)
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	
Release Notes and Known Issues	Master Answer Record: N/A
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
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 top-level block diagram for the Xilinx® LogiCORE™ IP AMM Slave Bridge is shown in Figure 1-1. The AMM Slave Bridge core translates AXI4 transactions into Avalon transactions. The bridge functions as a slave on the AXI4 interface and as a master on the Avalon interface.

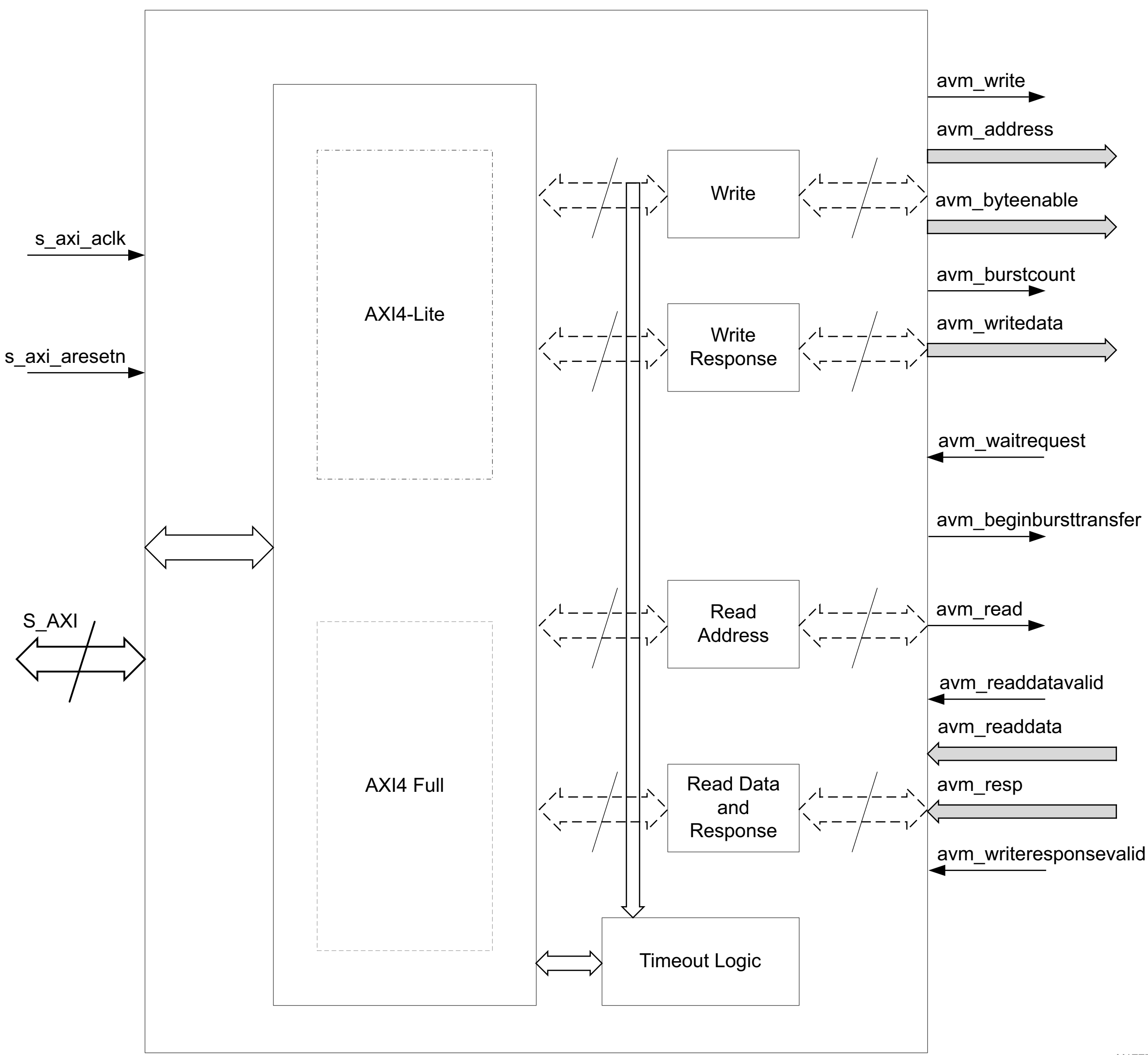

Figure 1-1: **AMM Slave Bridge Top-Level Block Diagram**

Feature Summary

AXI4-Lite Slave Interface

The AXI4-Lite slave interface module provides a bidirectional slave interface to the AXI interface. The AXI address width is from 1 to 64 bits and the data width is 32-bits. When both write and read transfers are simultaneously requested on the AXI4-Lite interface, the read request has higher priority than the write request. This module also contains the data phase timeout logic for generating a SLVERR response on the AXI interface when an Avalon slave does not respond.

The AXI4-Lite slave features include:

- Applicable for simple Avalon peripheral slave with register interface
- If Avalon has shared read and write channels, in the case of simultaneous read/write on the AXI interface, the read has higher priority over write
- 32-bit data width
- 1 to 64-bits address width
- AXI4 response generation in the case of Avalon response signal not present
- Ready generation depending on fixed or variable wait states for handshaking
- Bridge generates error response, if invalid address transactions are issued
- Configurable timeout logic for graceful completion of AXI transaction in case of no response from the Avalon slave

AXI4 Slave Interface

The AXI4 slave interface module provides a bidirectional slave interface to the AXI interface. The maximum outstanding read transactions is four.

In summary the Avalon supports the following:

- If Avalon has shared read and write channels, in the case of simultaneous read/write on the AXI interface, the read has higher priority over write
- 32 to 1,024-bits wide data bus
- 32 to 64-bits wide address bus
- AXI response generation even when Avalon response signal is not present
- Burst support
- Support for outstanding reads

- Supports symmetric data width and synchronous clock

Avalon Master Interface

Avalon master interface is an Altera®-compliant interface.

- Separate address, data, and control lines
- 32 to 1,024-bits wide data bus in AXI4 mode
- 32 to 64-bits wide address bus in AXI4 mode
- 1 to 64-bits wide address bus in AXI4-Lite mode
- 32-bit wide data bus in AXI4-Lite mode
- Synchronous operation

Applications

The AMM Slave Bridge has the following applications:

- **Converting AXI4-Lite Traffic to Avalon Traffic** – Used for register accessing or small data transfers
- **Converting AXI4 Traffic to Avalon Traffic** – Used for higher data transfers like a memory using burst

Unsupported Features

- Does not support setup and hold time Avalon bus interface properties.
- Bridge does not perform any upsizing/downsizing of data width.
- Bridge supports active-High Avalon control signals (`read/write/wait request/readdatavalid/writeresponsevalid`).
- Outstanding transactions are not supported in AXI4-Lite mode of IP.
- Does not support pipelined/outstanding write transactions as Avalon protocol does not support this feature.
- Narrow burst support.
- The bridge does not break the AXI4 transaction into child transactions. Ensure that burst value coming from the AXI4 master (`arlen` and/or `awlen`) transaction is within the permissible limits of the Avalon slave. This restriction is applicable when the burst count width of Avalon slave < 9.

Licensing and Ordering

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 about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Standards

- Processor Interface, AXI4-Lite: see the *Vivado Design Suite: AXI Reference Guide* (UG1037) [Ref 5]
- Avalon Interface Specifications [Ref 2]

Performance

For full details about performance and resource utilization, visit the Performance and Resource Utilization web page.

The performance characterization of this core was compiled using the margin system methodology. The details of the margin system characterization methodology are described in the *Vivado Design Suite User Guide: Designing With IP* (UG896) [Ref 10].

Latency

Latency varies with the `waitrequest` and slave read latency. End-to-end latency depends on the AXI to Avalon transition delay (Avalon slave latency).

AXI4-Lite

Read Latency:

- `axi_arvalid` to `avm_read` : 1 clock + Slave `waitrequest` delay
- `avm_readdatavalid` to `axi_rvalid` : 1 clock cycle

Write Latency:

- `axi_awvalid` to `avm_write` : 1 clock + Slave `waitrequest` delay
- `axi_awvalid` to `awready` : 2 clock + Slave `waitrequest` delay
- `avm_writeresponsevalid` to `axi_bvalid` : 1 clock cycle

AXI4

Read Latency:

- `axi_arvalid` to `avm_read` : 3 to 5 clock cycles + Slave `waitrequest` delay
- `avm_readdatavalid` to `axi_rvalid` : 3 clock cycles

Write Latency:

- `axi_awvalid` to `avm_write` : 1 clock + Slave `waitrequest` delay
- `axi_awvalid` to `awready` : 2 clocks + Slave `waitrequest` delay
- `avm_writeresponsevalid` to `axi_bvalid` : 1 clock

Throughput

The throughput of AMM Slave Bridge is calculated by using the following formula:

Throughput = (Number of beats transferred × data width) / (Time taken to complete the transaction)

Table 2-1 shows the throughput data for this IP core.

Table 2-1: **Throughput**

Configuration	Read (with one outstanding) Throughput in Gb/s	Read (with four outstanding) Throughput in Gb/s	Write Throughput in Gb/s
Burst length = 32	5.25	6.1	5.85
Burst length = 64	5.77	6.3	6.2
Burst length = 128	6.07	6.35	6.25
Burst length = 256	6.23	6.37	6.33

Notes:

1. Throughput data for a 32-bit AXI data width and an AXI interface running at 200 MHz. The theoretical throughput for this configuration is 6.4 Gb/s. The AMM Slave Bridge throughput increases with higher burst length and higher outstanding read transaction support.

Resource Utilization

For full details about performance and resource utilization, visit the Performance and Resource Utilization web page.

Table 2-2 shows the device utilization for this IP core.

Table 2-2: **Device Utilization**

Bridge Mode	Configuration			LUTs	FFs	LUTFFs	36k Block RAMs	18k Block RAMs	Max Freq (MHz)
	C_HAS_WAIT_ REQUEST	C_HAS_READ_ DATA_ VALID	C_NUM_ ADDRESS_ RANGES						
AXI4-Lite	1	1	0	186	126	125	0	0	600
AXI4-Lite	1	1	1	225	126	123	0	0	600
AXI4-Lite	0	0	1	219	180	110	0	0	600
Bridge Mode	**C_S_AXI_DATA_ WIDTH**	**C_S_AXI_ ADDR_WIDTH**	**C_NUM_ OUTSTANDING**	**LUTs**	**FFs**	**LUTFFs**	**36k Block RAMs**	**18k Block RAMs**	**Max Freq (MHz)**
AXI4	32	32	4	292	266	183	1	1	400
AXI4	256	40	4	304	275	182	7	2	400
AXI4	1024	64	1	385	315	209	14	2	400

Port Descriptions

Table 2-3 shows the AMM Slave Bridge signals.

Table 2-3: **AMM Slave Bridge Interface Signals**

Signal Name	Interface	Presence	I/O	Initial State	Description
s_axi_aclk	Clock	M	I	–	AXI clock
s_axi_aresetn	Reset	M	I	–	AXI reset
s_axi_*	S_AXI	M	I	–	AXI slave interface (Address width increase with respect to legacy ports as per register space requirement)
avm_address	Avalon	M	O	0x0	Avalon address bus
avm_byteenable	Avalon	O	O	0xFFFF	Active-High output. Enables specific byte lane(s) during transfers on interfaces of width > 8 bits. Each bit in byteenable corresponds to a byte in writedata and readdata.
avm_write	Avalon	M	O	0	Asserted to indicate a write transfer. If present, writedata is required.
avm_writedata	Avalon	M	O	0x0	Avalon read data bus
avm_read	Avalon	M	O	0	Asserted to indicate a read transfer. If present, readdata is required.
avm_waitrequest	Avalon	O	I	–	Avalon address and write data ready
avm_readdatavalid	Avalon	O	I	–	Avalon read data valid signal. It is mandatory port for burst transfers.
avm_readdata	Avalon	M	I	–	Avalon read data bus
avm_resp	Avalon	O	I	–	The response signal is an optional signal that carries the response status. ***Note:*** This signal is shared across write and read transactions.
avm_writeresponsevalid	Avalon	O	I	–	Indicates whether the write response is valid or not
avm_burstcount	Avalon	O	O	0x0	Indicates the number of transfers in each burst
avm_beginbursttransfer	Avalon	O	O	0	Asserted for the first cycle of a burst to indicate when a burst transfer is starting

Notes:

1. O in the “Presence” column of table indicates Optional signal.

Designing with the Core

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

General Design Guidelines

Timeout

The AMM Slave Bridge provides a feature to terminate the transaction if response is not received from the Avalon side. For example, if the timeout is set to 32 and the bridge does not receive any `response/waitrequest` from the Avalon slave, then the bridge drops transfers to the Avalon slave and closes the AXI transaction gracefully. This is helpful in ensuring AXI does not get stalled. In case of a timeout, the bridge terminates the AXI with an error response (slave error).

Endianness

Both AXI4 and Avalon are little-endian.

Bridge Error Conditions

The bridge responds with an error response (slave error) when a timeout condition is hit.

Clocking

The IP has a single clock domain and the AXI4 and Avalon interface clock domain is clocked by `s_axi_aclk`.

Resets

The IP has a single reset, and the AXI4 and Avalon interface domain is reset by `s_axi_aresetn`.

AMM Slave Bridge Operation (AXI4-Lite)

The AMM Slave Bridge Operation has a write and read transaction.

Write Transaction

The write transaction has three functions:

- Address
- Data
- Write Response

Address

For a write transaction, the AMM Slave Bridge expects address and data to be asserted concurrently on the AXI side. The AMM Slave Bridge supports byte aligned and word aligned address modes. For byte addressing, the address passed to the Avalon is same as what is received on the AXI. For word addressing, the last two bits of the AXI address are truncated before sending to the Avalon interface.

Data

When the data and address are available on the AXI bus, the AMM Slave Bridge asserts a write signal and updates the address and write data buses of the Avalon slave.

Ready Generation When Variable Wait

If the Avalon slave has the `waitrequest` port, then the `wready` and `awready` are generated based on the `waitrequest`.

Ready Generation When Fixed Wait

If the Avalon slave does not have the `waitrequest` port, then the `wready` and `awready` are generated after the number of clock cycles of `awvalid` assertion as specified in the IP configuration.

Write Response

When write control signals are present when response generation is supported by the slave, response is generated using the `writeresponsevalid` and `resp` signals of slave.

When write control signals are not present in the slave, for every write transaction the bridge generates valid write responses to the AXI master.

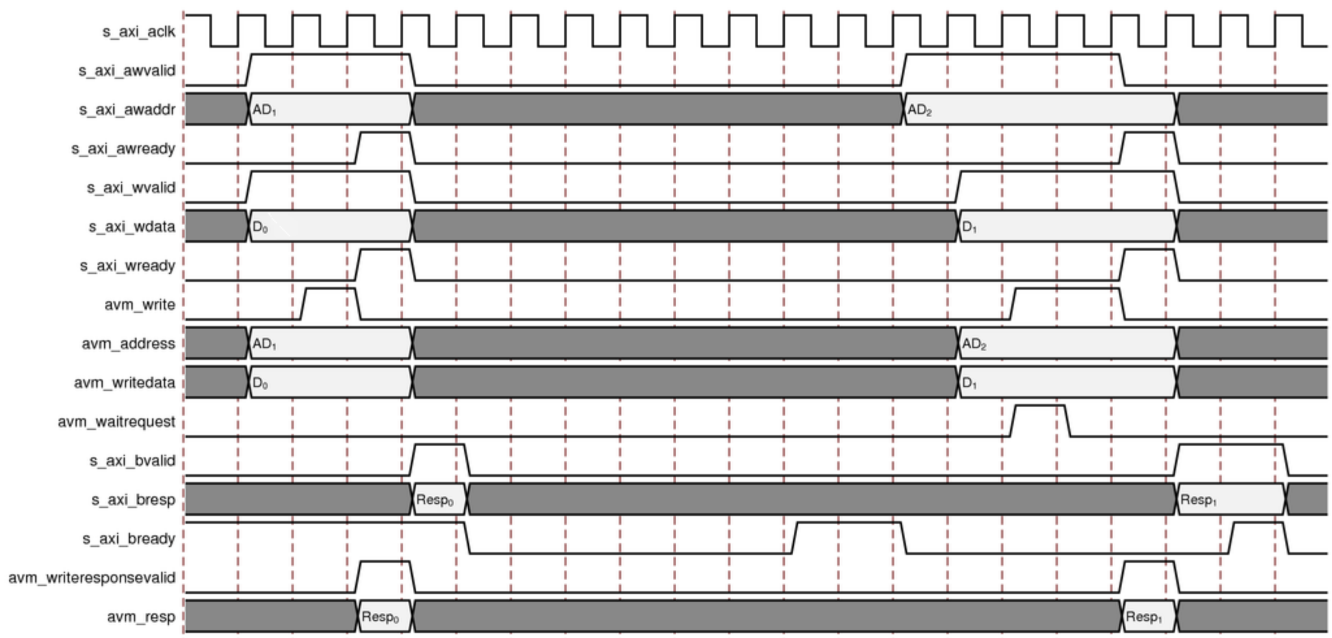

Figure 3-1: **AXI4-Lite Write Response Timing Diagram**

Read Transaction

The read transaction has three functions:

- Address
- Data
- Read Response

Address

A read on the Bridge is initiated when read address is available on the AXI bus. In the event of a simultaneous write and read access on AXI side, the read gets the preference.

Ready Generation When Variable Wait

If the Avalon slave has the `waitrequest` signal, then the `arready` signal is generated based on the status of `waitrequest`.

Ready Generation When Fixed Wait

In absence of the `waitrequest` signal, the `arready` is generated after the fixed clock cycles as specified in the IP configuration.

Data

When Avalon Slave accepts the address, it puts the data on the `readdata` bus.

Read Valid Generation When Variable Wait

If the slave has the `readdatavalid` signal, the data is sampled by the Avalon Slave Bridge when this signal is asserted.

Read Valid Generation When Fixed Wait

In absence of the `readdatavalid` signal, the Avalon Slave Bridge samples the data after the fixed number of clock cycles of address acceptance as specified in the IP configuration.

Read Response

When the read control signals are present and response generation is supported by the slave, the read response is generated using the `readdatavalid` (or, after the fixed latency) and `resp` signals of slave.

When write control signals are not present in the slave, for every read transaction the bridge generates valid read responses to the AXI master.

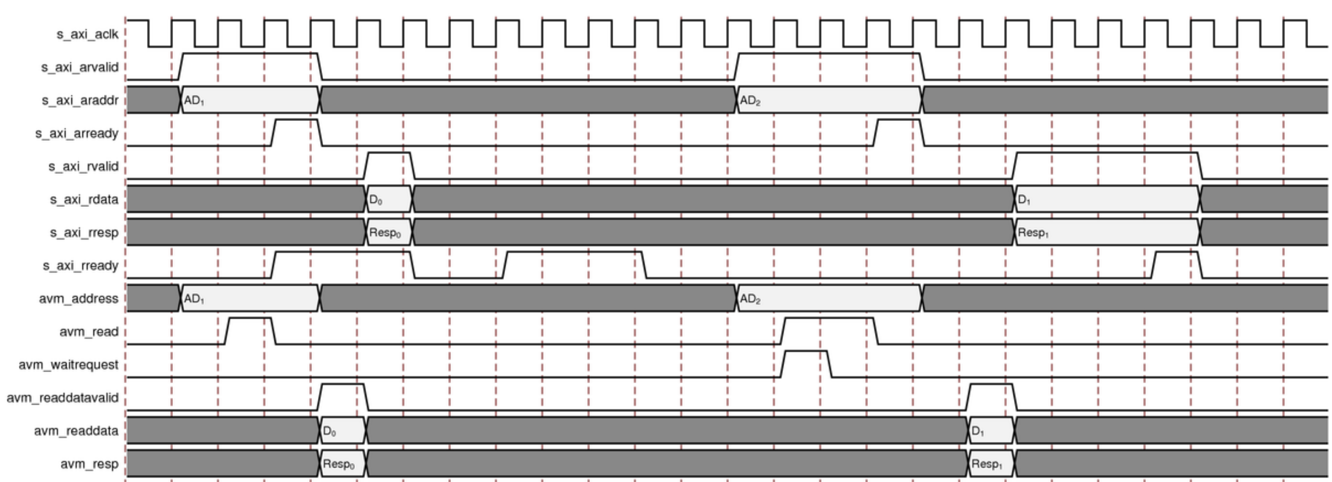

Figure 3-2: **AXI4-Lite Read Response Timing Diagram**

Bridge Operation (AXI4)

The AMM Slave Bridge IP supports burst feature in AXI4 mode. Ensure that the burst length issued over the AXI Master transaction does not exceed the slave supported burst count. The maximum supported burst length is one less than the configured Avalon burst count width.

IMPORTANT: *The AXI bus interface property (MAX_BURST_LENGTH) in the Vivado IP integrator is updated based on the burst count width value. During the IP validation, it might result in a critical warning if the properties do not match. Ensure that the master interface property matches the bridge interface property.*

Write Operation

The write operation has three functions:

- Address
- Data
- Response

Address

The AMM Slave Bridge accepts the AXI4 write address in the idle state. The write address is not accepted if the bridge is performing a read access.

Data

When the address is accepted, the bridge then starts accepting the data. The `wready` signal is generated based on the `waitrequest` signal from the Avalon slave.

Response

The AMM Slave Bridge captures the response from the slave and puts it on write response channel. If the slave does not have any write response, the bridge terminates the response channel with an OKAY response.

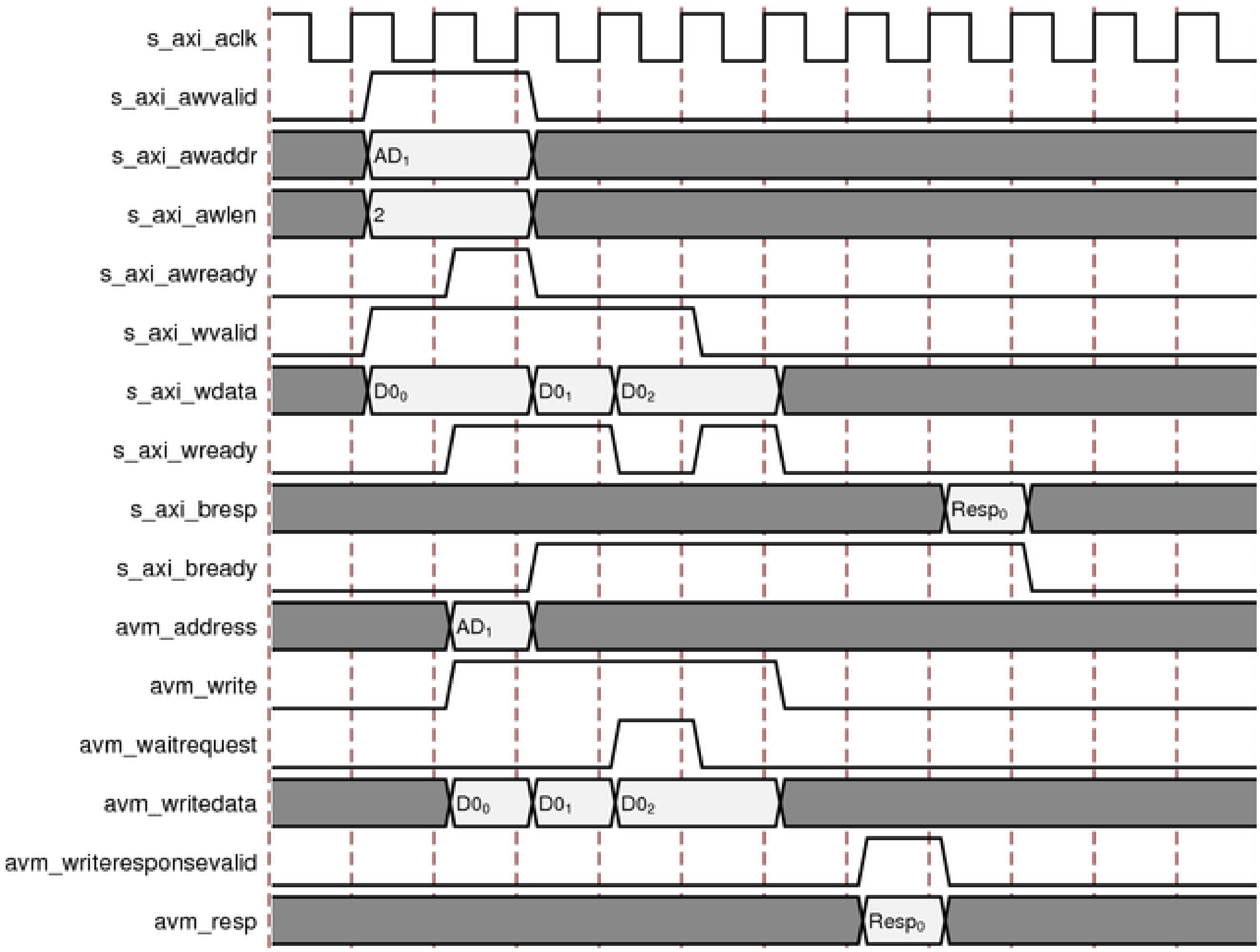

Figure 3-3: **AXI4 Write Response Timing Diagram**

Read Operation

The read operation has three functions:

- Address
- Data
- Response

Address

The AMM Slave Bridge accepts the AXI4 read address in the idle state. The bridge supports (and accepts) up to four outstanding read requests. In the event of a write and read address coming simultaneously, the read gets the preference.

Data

The AMM Slave Bridge accepts the `readdata` from the slave and stores it in a FIFO to support outstanding transaction from the AXI4 master. In case of back pressure from the AXI4 interface, the bridge can hold up to four transaction data. The AXI4 then reads the data from the FIFO.

Response

The AMM Slave Bridge captures the response from the slave and stores it in the FIFO along with the data. If the slave does not support response generation, the AMM Slave Bridge assumes the response channel to be OKAY.

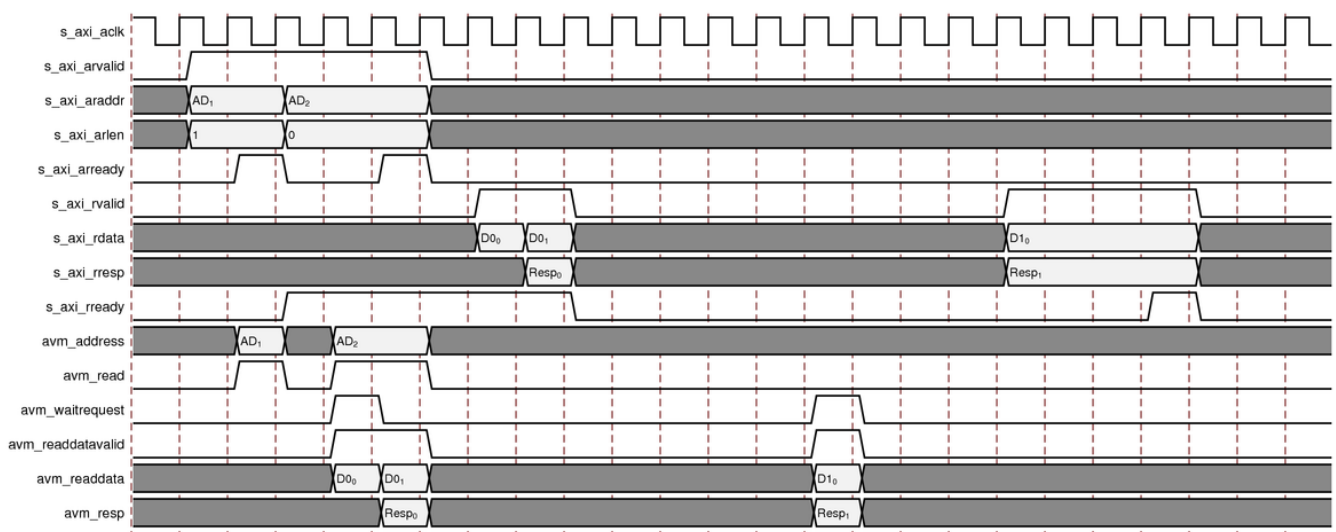

Figure 3-4: **AXI4 Read Response Timing Diagram**

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 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 9]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 10]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 11]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 12]

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 Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 9] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do 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 10] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 11].

Note: Figures in this chapter are an illustration of the Vivado Integrated Design Environment (IDE). The layout depicted here might vary from the current version.

Figure 4-1 shows the AMM Slave Bridge Vivado IDE main configuration screen.

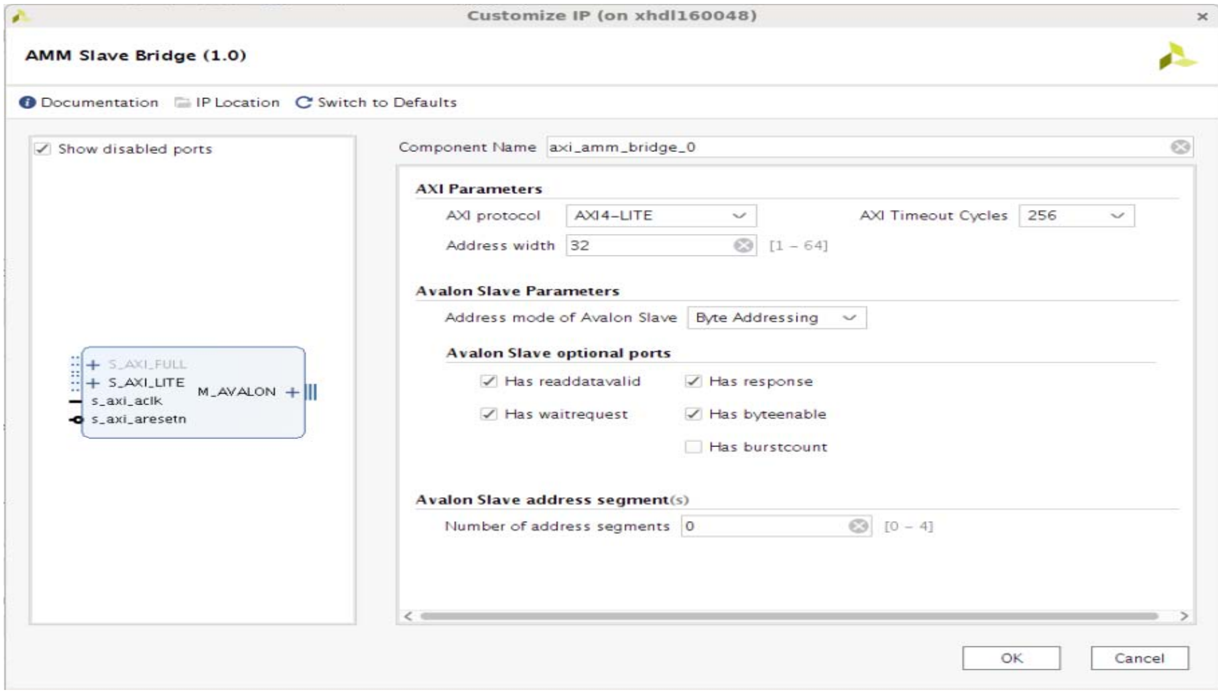

Figure 4-1: **AMM Slave Bridge Customize IP**

The following settings are generally applicable:

- **Component Name** – The component name is used as the base name for output files generated for the module. Names must begin with a letter and can include the following characters: a to z, 0 to 9, and "_".
- **AXI Protocol** – Select the AXI protocol (AXI4-Lite/AXI4).
- **AXI Timeout Cycles** – Select the AXI timeout cycles:
    - Select the duration of timeout counter.
    - This counter kicks in when the response or acknowledge from the Avalon is not received within that time. The IP gracefully completes the AXI transaction by issuing an ERROR response.
- **Address Width** – Select the address width of the slave. Set the width of the address to match the Avalon slave.

- **Has Response** – If this port is present, the AMM Slave Bridge translates the Avalon response to an AXI response. If not present, the bridge prepares an AXI response for all transactions.
- **Has Byteenable** – Select to enable specific byte lane(s) during transfers on interfaces of width greater than 8 bits. Each bit in `byteenable` corresponds to a byte in `writedata` and `readdata`.
- **Has Burstcount** – Enable this signal if the port is present in the Avalon slave. This configures width of the burst count port. In AXI4-Lite mode, the AMM Slave Bridge drive burst length of 1 on this port.

* **AMM Slave Address Segment** – The AMM Slave Bridge supports up to four address segments to allow different address space in the Avalon slave. Enter the ranges for address segment. No two segments can overlap. A value of 0 indicates that the entire address range is valid. For values other than 0, if the AXI transaction address doesn't fall within any of the mentioned ranges, the transaction returns a slave error response on the AXI interface and no command is generated on the Avalon interface. For example, if the address segments are customized as mentioned in the below figure, any address below 0x100 and any address from and above 0x500 reports a slave error ("SLVERR").

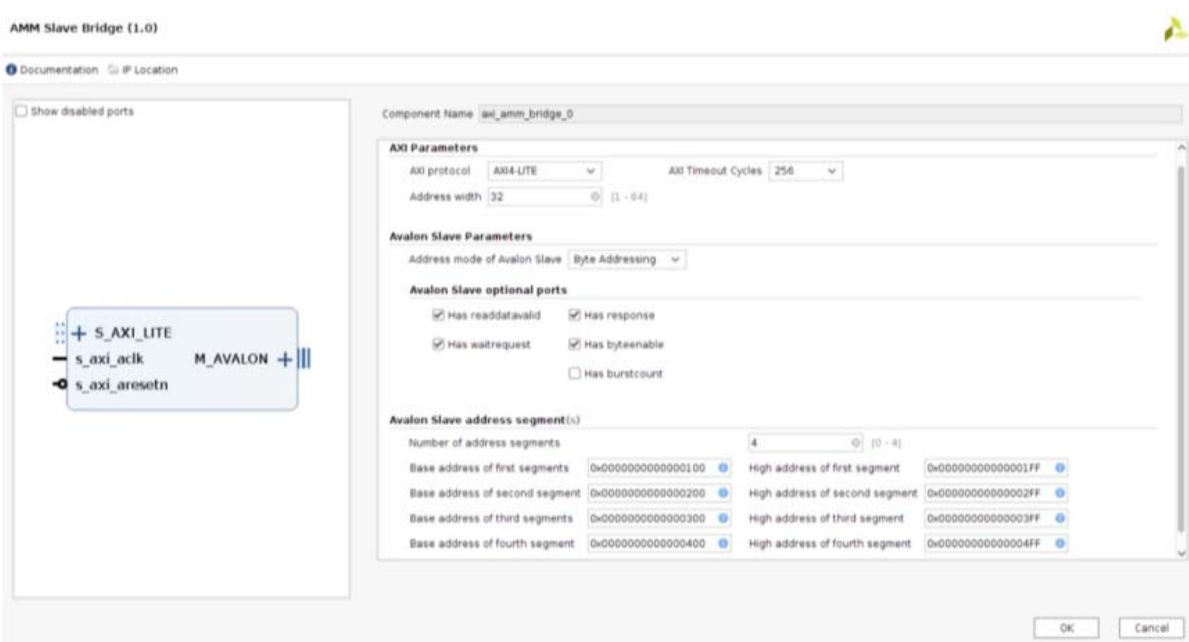

Figure 4-2: **AMM Slave Bridge Parameters**

User Parameters

Table 4-4 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 4-4: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE/IP Integrator Parameter	Default Value	Range	Description
C_PROTOCOL	AXI4-Lite	AXI4-Lite, AXI4	AXI protocol
C_S_AXI_ADDR_WIDTH	32	1 to 64	Address width of AXI master and Avalon slave
C_S_AXI_DATA_WIDTH	32	32 (When AXI mode is AXI4Lite) 32 to 1,024 when AXI mode is AXI4	Data width of AXI master and Avalon slave
C_NUM_ADDRESS_RANGES	0	0 to 4	Number of valid address segments
C_BASE1_ADDR	0x00000000	0x00000000 to 0xFFFFFFF0	Base address of first segment
C_BASE2_ADDR	0x00000004	First Segment High Address to 0xFFFFFFF0	Base address of second segment should be more than First Segment High Address and less than Second Segment High Address
C_BASE3_ADDR	0x00000008	Second Segment High Address to 0xFFFFFFF0	Base address of third segment
C_BASE4_ADDR	0x0000000C	Third Segment High Address to 0xFFFFFFF0	Base address of fourth segment
C_HIGH1_ADDR	0x00000003	0x0000000F to 0xFFFFFFF0	High address of first segment
C_HIGH2_ADDR	0x00000005	First Segment High Address to 0xFFFFFFF0	High address of second segment
C_HIGH3_ADDR	0x00000009	Second Segment High Address to 0xFFFFFFF0	High address of third segment
C_HIGH4_ADDR	0x0000000F	Third Segment High Address to 0xFFFFFFF0	High address of fourth segment
C_HAS_WAIT_REQUEST	Yes	No, Yes	When this parameter is set to 1, waitrequest port is enabled in the bridge.
C_USE_BYTEENABLE	Yes	No, Yes	ByteEnable support
C_FIXED_WRITE_WAIT	1	1 to 255	For interfaces that do not use the waitrequest signal, write wait time specifies the timing in number of clock cycles before a slave accepts a write.

Table 4-4: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE/IP Integrator Parameter	Default Value	Range	Description
C_FIXED_READ_WAIT	1	1 to 255	For interfaces that do not use the waitrequest signal, read wait Time indicates the timing in number of clock cycles before the slave accepts a read command.
C_HAS_READ_DATA_VALID	Yes	No, Yes	When fixed, read latency time is defined, readdatavalid is disabled in bridge.
C_READ_LATENCY	1	1 to 63	For interfaces that do not use the readdatavalid signal, read latency indicates the timing in number of clock cycles after the address acceptance data is valid on rdata bus.
C_DPHASE_TIMEOUT(2)	32	32, 64, 218, 256	Timing out the transaction when there is no response from Avalon slave for programmed clock cycle duration. This avoids hang scenarios and safely terminates the current transaction.
C_ADDRESS_MODE	Byte addressing	Byte addressing, Word addressing	Specifies the unit for addresses. Address Translation for 32-bit data width. Byte Addressing: 0x0, 0x4, 0x8, 0xC. Word Addressing: 0x0, 0x1, 0x2, 0x3.
C_HAS_RESPONSE	Yes	No, Yes	If supported, it is a Shared response for Read/Write and writeresponsevalid port is enabled. If not supported, bridge generates AXI response, writeresponsevalid port is disabled.
C_AVM_BURST_WIDTH	4	1 to 11	To find the bus width of burst count
C_S_AXI_ID_WIDTH	1	0 to 32	ID width of AXI when the parameter value is 0, the corresponding RID, WID, and BID signals are not enabled.
C_BEGIN_BURST_TRANSFER	1	0, 1	Beginbursttransfer signal enablement.

Table 4-4: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE/IP Integrator Parameter	Default Value	Range	Description
C_NUM_OUTSTANDING	2	1 to 4	Maximum outstanding read transactions.

Notes:

1. Base address should always be aligned to data width.
2. Whenever there is no response from Avalon slave for the configured C_DPHASE_TIMEOUT clock cycles, bridge deasserts the control signals and terminates the AXI transaction with an error response. This feature is added to avoid hang scenarios in a system. When this occurs, transaction on the Avalon side is terminated prematurely. It might lead to protocol violations on the Avalon side.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 10].

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 12].

IMPORTANT: *For cores targeting 7 Series FPGAs 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 10].

Packaging Avalon Slave Interface

The new Avalon bus interface is created and is available in the Vivado 2016.3 release. For system migration, the Avalon slave IP has to be packaged for Vivado using the Avalon bus interface (Figure 4-3). This allows Avalon IP to integrate with the Vivado IP integrator system.

To package the Avalon slave IP, follow the steps mentioned in the *Vivado Design Suite Tutorial: Creating and Packaging Custom IP* (UG1119) [Ref 8].

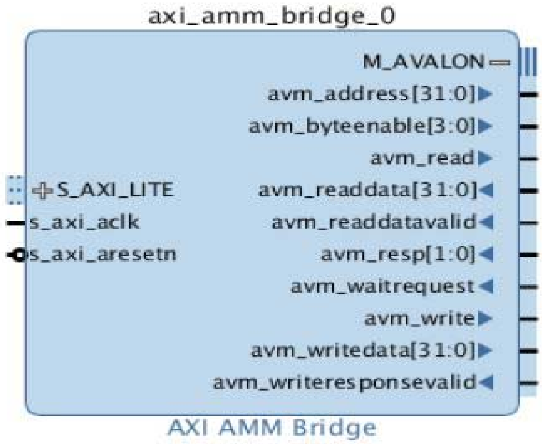

Figure 4-3: **AMM Slave Bridge Bus Interface**

Creating an IP Integrator Design

Figure 4-4 shows an IP integrator design that uses the `amm_slave_bridge` and a packaged Avalon memory map slave (`avalon_slave_0`).

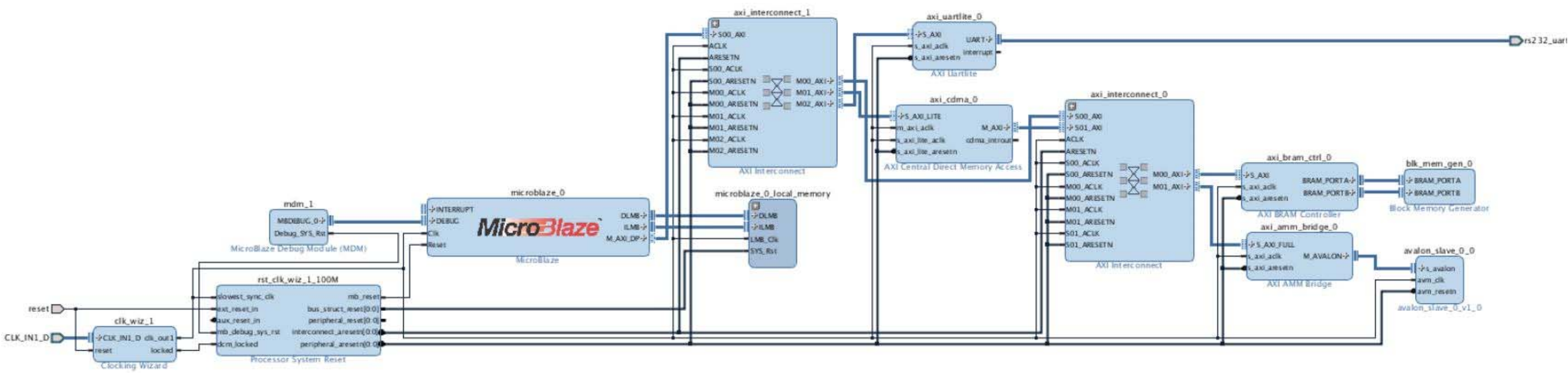

Figure 4-4: **AMM Slave Bridge with an IP Integrator Design**

Example Design

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

Overview

The example design demonstrates the usage of the amm_slave_bridge. It uses the axi_traffic_generator IP to generate the AXI4-Lite traffic. The Avalon slave is modeled as a memory with the Avalon interface. The axi_traffic_generator (ATG) initiates writes to the Avalon slave. The ATG then reads this data and compares the same.

The example design contains the following:

- An instance of the AMM Slave Bridge core
- Clocking wizard to generate clock signals for the example design
- Traffic generator for AXI4 and AXI4-Lite interfaces
- An instance of the Avalon slave memory model
- IP is verified by data comparison by writing to slave memory and is read back from the same location

Note: The example design and the Avalon slave are not generic designs. The example design and the Avalon slave are generated based on the IP configuration. The Avalon slave in the example design works only with the corresponding bridge configuration. Xilinx does not recommend modifying the ATG COE configuration.

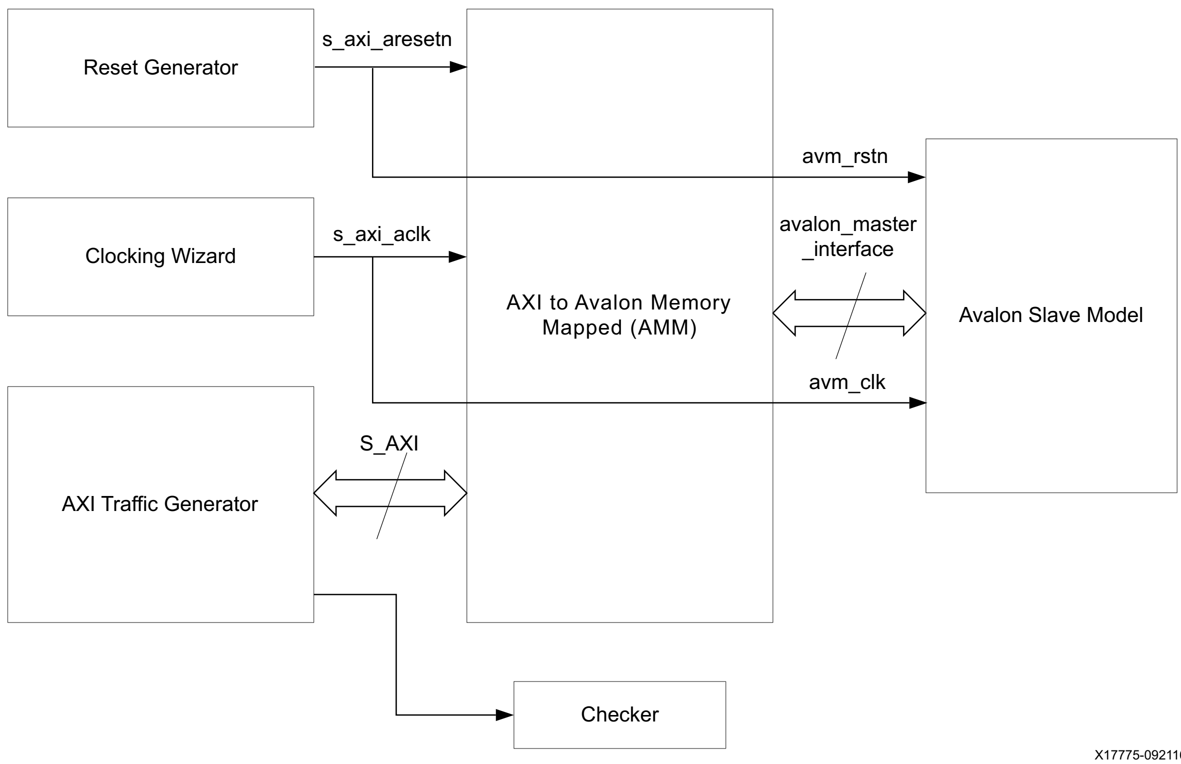

Figure 5-1: **AMM Slave Bridge Example Design**

Implementing the Example Design

After following the steps described in Chapter 4, Customizing and Generating the Core, implement the example design using the following instructions:

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 IDE window.
4. In the Flow Navigator (left pane), click **Run Implementation** and follow the directions.

Example Design Directory Structure

In the current project directory, a new project named `<component_name>_example` is created and the files are generated in `<component_name>_example.src/sources_1/ip/<component_name>/` directory. This directory and its subdirectories contain all the source files that are required to create the AMM Slave Bridge example design.

The example design directory is created in <component_name>_example.src/sources_1/imports/<component_name>. It contains the following generated example design top files:

- <component_name>_exdes.v – Top-level HDL file for the example design
- <component_name>_ava_slv.v – Example design slave model

Simulating the Example Design

Using the example design delivered as part of the AMM Slave Bridge, you can quickly simulate and observe the behavior of the core.

Setting up the Simulation

The Xilinx simulation libraries must be mapped to the simulator. To set up the Xilinx simulation models, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 12]. To switch simulators, click **Simulation Settings** in the Flow Navigator (left pane). In the Simulation options list, change **Target Simulator**.

The example design supports functional (behavioral) and post-synthesis simulations. For information on how to run simulation, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 12].

Simulation Results

The simulation script compiles the AMM Slave Bridge example design and supporting simulation files. It then runs the simulation and checks if it was successful.

If the test passes, the following message is displayed:

```
Test Completed Successfully
```

If the test freezes, the following message is displayed:

```
Test Hanged
```

If the test fails, the following message is displayed:

```
Test Failed
```

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 AMM Slave Bridge i.e., AXI to Avalon Memory Mapped (AMM), 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 AMM Slave 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.

Technical Support

Xilinx provides technical support at the 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 AMM Slave 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 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 14].

Additional Resources and Legal Notices

Xilinx Resources

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

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this product guide:

1. Instructions on how to download the Arm® AMBA® AXI specifications are at ARM AMBA Specifications. See the:
   - AMBA AXI4 Protocol Specification
   - AMBA4 AXI4-Stream Protocol Specification
2. Avalon Interface Specifications
3. *AMM Master Bridge LogiCORE IP Product Guide* (PG287)
4. *7 Series FPGAs Data Sheet: Overview* (DS180)
5. *Vivado Design Suite: AXI Reference Guide* (UG1037)
6. *Xilinx Software Development Kit User Guide: System Performance Analysis* (UG1145)
7. *System Performance Analysis of an* SoC (XAPP1219)
8. *Vivado Design Suite Tutorial: Creating and Packaging Custom IP* (UG1119)
9. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
10. *Vivado Design Suite User Guide: Designing with IP* (UG896)
11. *Vivado Design Suite User Guide: Getting Started* (UG910)
12. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
13. *ISE to Vivado Design Suite Migration Guide* (UG911)
14. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
15. *Vivado Design Suite User Guide: Implementation* (UG904)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/16/2020	1.0	Updated AXI and Avalon addresses for the 32-bit and the 64-bit Word-Addressing modes.
06/21/2019	1.0	Added the UltraScale family to the list of supported devices. Updated AMM Slave Address Segment section.
04/04/2018	1.0	Changed existing title ‘AXI to Avalon Memory Mapped Bridge IP’ to ‘AMM Slave Bridge’.
10/05/2016	1.0	Initial Xilinx release.

Please Read: Important Legal Notices