AXI Memory Mapped to Stream Mapper v1.1

LogiCORE IP Product Guide

PG102 April 5, 2017

XILINX®

XILINX

Table of Contents

Introduction

The AXI Memory Mapped to Stream Mapper IP (axi_mm2s_mapper) is used to encode and decode AXI4 Memory-Mapped (AXI4-MM) transactions into AXI4-Stream (AXI4-S) transfers, allowing AXI-MM transactions to be transported across AXI4-S networks.

Features

- Encapsulates AXI4-MM slave interface transactions onto two AXI4-S interfaces.
  - Supports AXI4 only.
- Expands AXI4-S transaction into AXI4-MM master interface transactions.
- Supports both encapsulation (S_AXI interface) and expansion (M_AXI interface) in a single module.
  - Allows for cross communication with AXI4-MM masters and slaves on both sides of AXI4-S link while only using two AXI4-S interfaces.
- AXI4-S TDATA width can be set independently of the AXI4-MM interface. When necessary, an AXI4-MM transfer can be split into multiple AXI4-S transfers to support desired AXI4-S TDATA width.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale+™ Families, UltraScale Architecture, Virtex®-7, Kintex®-7, Artix®-7
Supported User Interfaces	AXI4, AXI4-Stream
Resources	See Table 2-1.
Provided with Core	
Design Files	Verilog RTL
Example Design	Verilog
Test Bench	Verilog
Constraints File	Not Provided
Simulation Model	Verilog Behavioral
Supported S/W Driver(2)	N/A
Tested Design Flows(3)	
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 listing of supported devices, see the Vivado IP Catalog.
2. Standalone driver details can be found in the SDK directory (*<install_directory>*/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from the Xilinx Wiki page.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The function of the AXI Memory-Mapped to Stream Mapper IP (axi_mm2s_mapper) is to encapsulate AXI4 Memory-Mapped (AXI4-MM) transactions onto a pair of AXI4-Stream (AXI4-S) interfaces. This allows use of the AXI4-S components which are generally smaller in area, faster in frequency, and allow more flexibility in system designs. The AXI Memory-Mapped to Stream Mapper IP is intended to be used in pairs with one side of the pair converting the AXI4-MM transactions to AXI4-Stream transactions and the other half to perform in the inverse operation to expand the AXI4-S transactions to AXI4-MM transactions.

The AXI4-MM Write Address, Read Address, and Write Data channels are mapped onto one AXI4-S master interface while the Read Data and Write Response channels are mapped onto one AXI4-S slave interface. Together the two AXI4-S interfaces can carry the five AXI4-MM channels by multiplexing them in time. As the burst length of the AXI4-MM transaction is increased, the write data bandwidth lost due to time multiplexing will be minimized. Read data bandwidth can be maximized if the number of write responses sent during read transactions are minimized.

Each IP instance is capable of supporting both AXI4-MM master and AXI4-MM slave interfaces to support master/slave communication on both sides of the axi_mm2s_mapper pairs. The AXI4-S TDATA width can be configured to any arbitrary number of bytes.

IMPORTANT: *Each side of the axi_mm2s_mapper pair must be configured identically.*

When configuring TDATA widths that are smaller than the encapsulation size of the AXI4-MM transfer, the transfer is broken into multiple AXI4-S transfers and then re-assembled at the endpoint seamlessly.

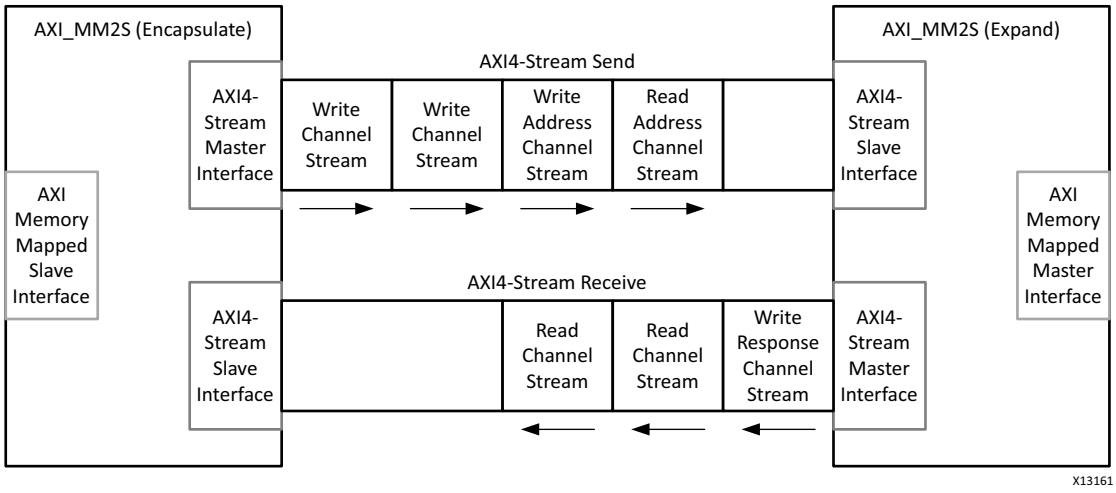

Figure 1-1: **AXI MM2S Encapsulate and Expand**

Feature Summary

The IP module always has two AXI4-Stream interfaces: one master and one slave. The interfaces can be configured to have any `TDATA` width between 1 and 512 bytes. The interfaces also contain the `TKEEP`, `TLAST`, and `TID` (width of 3 bits) signals.

The IP module can be configured to have either a AXI4-MM master interface, AXI4-MM slave interface or both. The interfaces support variable ADDR, ID and USER signals.

A single clock and active-Low reset is supported.

Unsupported Features

Only the AXI4-MM interface is supported. If using AXI3 or AXI4-Lite, one conversion module should be used to convert to AXI4-MM before the mapper encapsulation. A second conversion module should then convert back to the desired protocol after the mapper expansion.

The AXI4-MM signals `REGION` and `QOS` are not supported. If these signals are used in the system they can be mapped onto the `USER` signals.

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

Product Specification

Figure 2-1 shows a simplified block diagram of the AXI Memory Mapped to Stream Mapper. The paths designated in blue are related to the AXI4 slave interface (encapsulation), while the paths designated in red are related to the AXI4 master interface (expansion). The data width converters are instantiated optionally when the TDATA width is smaller than the vector of all the signals in the corresponding AXI4-MM channel.

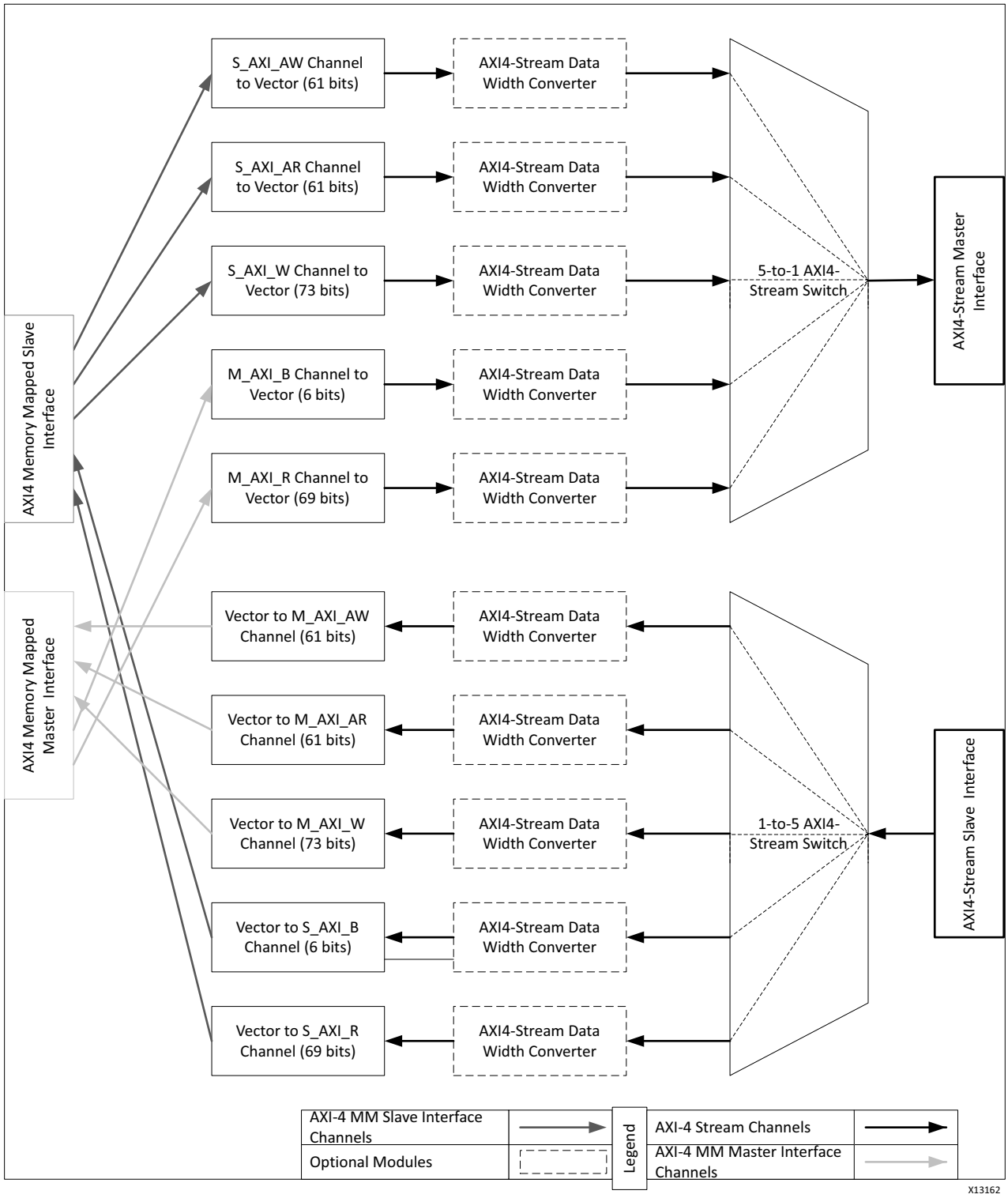

Figure 2-1: **Top-Level Block Diagram**

TDATA Packing

The AXI4 Memory mapped transfers are packed onto the AXI4-Stream TDATA signal. This section describes how the AXI4 transactions are mapped.

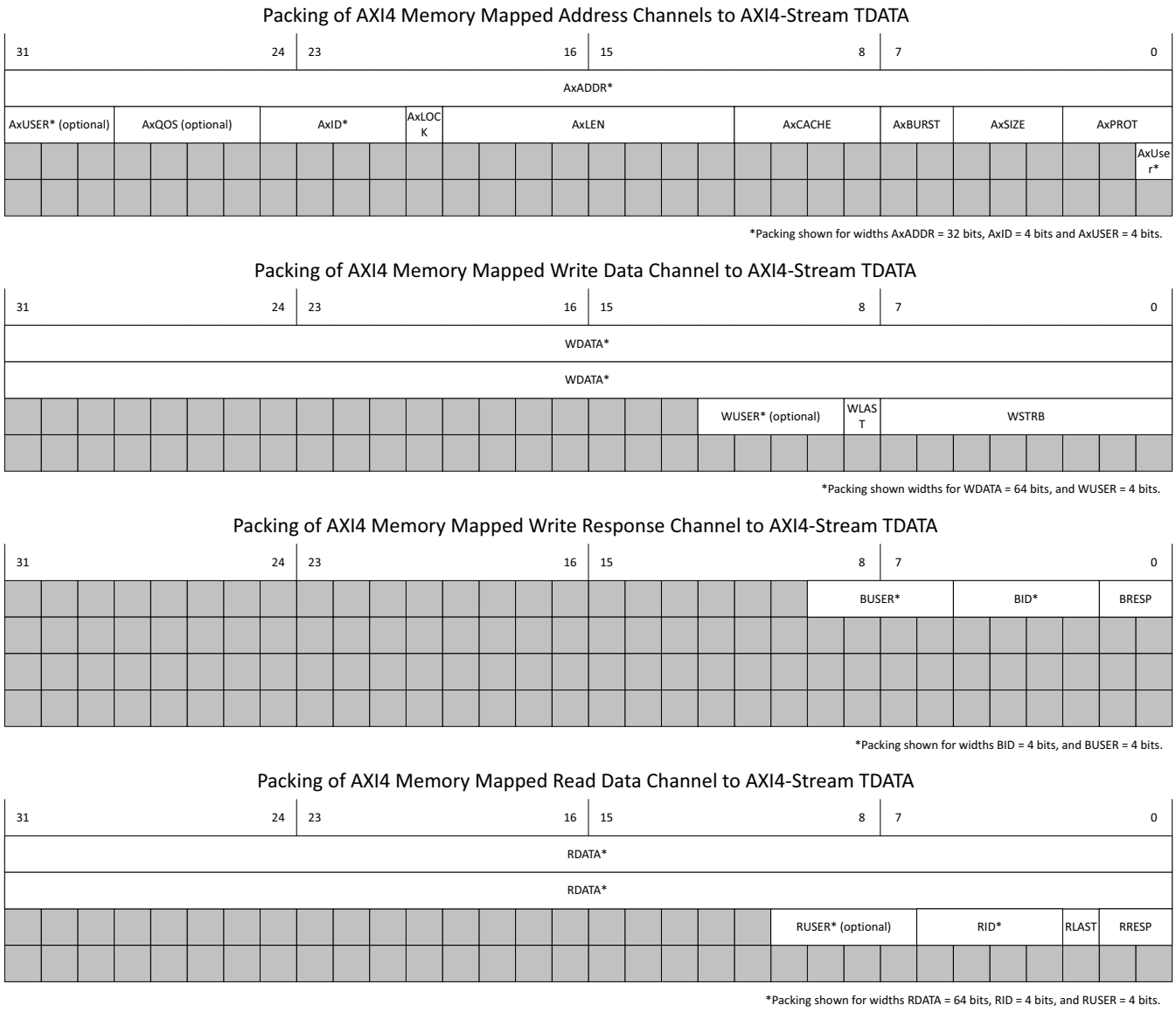

Figure 2-2: **AXI4 Memory Mapped to AXI4-Stream TDATA Packing**

The AXI4 Memory Mapped to AXI4-Stream TDATA packing is shown in Figure 2-2.

The AXI4 Memory Mapped signals are packed onto the LSB of the TDATA signals. With an address of 32 bits and a 4-bit ID, 9 bytes of TDATA are needed to map the address channels. If the address or ID are reduced by 1 bit, then the transfer can fit into 8 bytes of TDATA.

Adding USER bits may cause a spillover into nine or more bytes, which may cause degraded performance.

The write data channel requires 5 bytes of TDATA if a WDATA width of 4 bytes is configured. Assuming no user signals, the number of TDATA bytes required for each of the WDATA widths in the write channel can be calculated as:

$$W_{TDATA} = W_{WDATA} + ceil\left(\frac{W_{WDATA} + 1}{8}\right)$$

where W_x is the width in bytes of signal x.

The read address, write address, and write data channels are multiplexed onto one outgoing Stream and are zero padded at the end to extend the packing width to be an integer multiple of the specified TDATA width. A downsizer will then be used to appropriately split the transactions if necessary.

The write response channel always requires 1 byte of TDATA or more if BID exceeds 6 bits.

The read data channel always requires a minimum of the number of RDATA bytes + 1.

The write response and the read data channels are expanded from one incoming stream. An upsizer is used to merge the transactions back to one transfer if necessary.

Standards

The AXI interfaces conform to the Advanced Microcontroller Bus Architecture (AMBA®) AXI version 4 specification from Advanced RISC Machine (ARM®). See the *ARM AMBA AXI Protocol v2.0* [Ref 1].

Performance

Maximum performance can be obtained by choosing an AXI4-Stream TDATA width that is greater than the width of the widest channel when converted to a vector. If the TDATA width is smaller than the AXI4-MM channel, the transfer must be converted into multiple AXI4-S transfers.

Resource Utilization

Resources required for the axi4_mm2s_mapper core have been estimated for the Kintex®-7 XC7K325T-FFG900-1 FPGA (Table 2-1). These values were generated using the Vivado® IP Catalog. They are derived from post-implementation reports, and may deviate slightly under different circumstances.

Table 2-1: **Kintex-7 XC7K325T-FFG900-1 FPGA Resource Estimates**

Feature	LUTs	FFs	Block RAMs	DSP48A
Widths(bits): • ADDR = 24 • DATA = 128 • ID = 4 • TDATA = 64	844	898	0	0

Port Descriptions

The following ports are described in this section:

- Global Ports
- AXI4 Memory Mapped Ports
- AXI4-Stream Ports

Global Ports

Table 2-2: **Global Port Descriptions**

Name	Description
ACLK	The global clock signal. All signals are sampled on the rising edge of ACLK.
ARESETN	The global reset signal. `ARESETN` is active-Low.

AXI4 Memory Mapped Ports

This IP core uses the standard AXI4 memory mapped ports. See the ARM AMBA specification [Ref 2] for a description of the ports.

AXI4-Stream Ports

Table 2-3: **AXI4-Stream Port Descriptions**

Name	Description
S_AXIS_TVALID M_AXIS_TVALID	TVALID indicates that the master is driving a valid transfer. A transfer takes place when both TVALID and TREADY are asserted.
S_AXIS_TREADY M_AXIS_TREADY	TREADY indicates that the slave can accept a transfer in the current cycle.
S_AXIS_TDATA [(C_AXIS_TDATA_WIDTH-1):0] M_AXIS_TDATA [(C_AXIS_TDATA_WIDTH-1):0]	TDATA is the primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
S_AXIS_TKEEP [(C_AXIS_TDATA_WIDTH/8-1):0] M_AXIS_TKEEP [(C_AXIS_TDATA_WIDTH/8-1):0]	TKEEP is the byte qualifier that indicates whether the content of the associated byte of TDATA is processed as part of the data stream. Associated bytes that have the TKEEP byte qualifier deasserted are null bytes and can be removed from the stream.
S_AXIS_TLAST M_AXIS_TLAST	TLAST indicates the boundary of a packet.
S_AXIS_TID [(C_AXIS_TID_WIDTH-1):0] M_AXIS_TID [(C_AXIS_TID_WIDTH-1):0]	TID provides routing information for the data stream.

Designing with the Core

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

Clocking

The IP core has one clock input, ACLK, that must be synchronous to all AXI interfaces on the module.

Resets

The IP core has one active-Low reset input, ARESETN, that must be synchronous with the ACLK signal.

Protocol Description

The AXI interfaces follow the standard AXI protocol. They use the VALID/READY handshake signals to pass data. For specific information about the ARM AXI interfaces, see the ARM documentation [Ref 1].

Customizing and Generating the Core

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

Vivado Integrated Design Environment

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

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

For details, see the sections, "Working with IP" and "Customizing IP for the Design" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 4] and the "Working with the Vivado IDE" section in the *Vivado Design Suite User Guide: Getting Started* (UG810) [Ref 6].

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 8] 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 you can run the `validate_bd_design` command in the Tcl console.

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

Interface

The options for customizing the AXI Memory Mapped to Stream Mapper IP, shown in Figure 4-1, are described below.

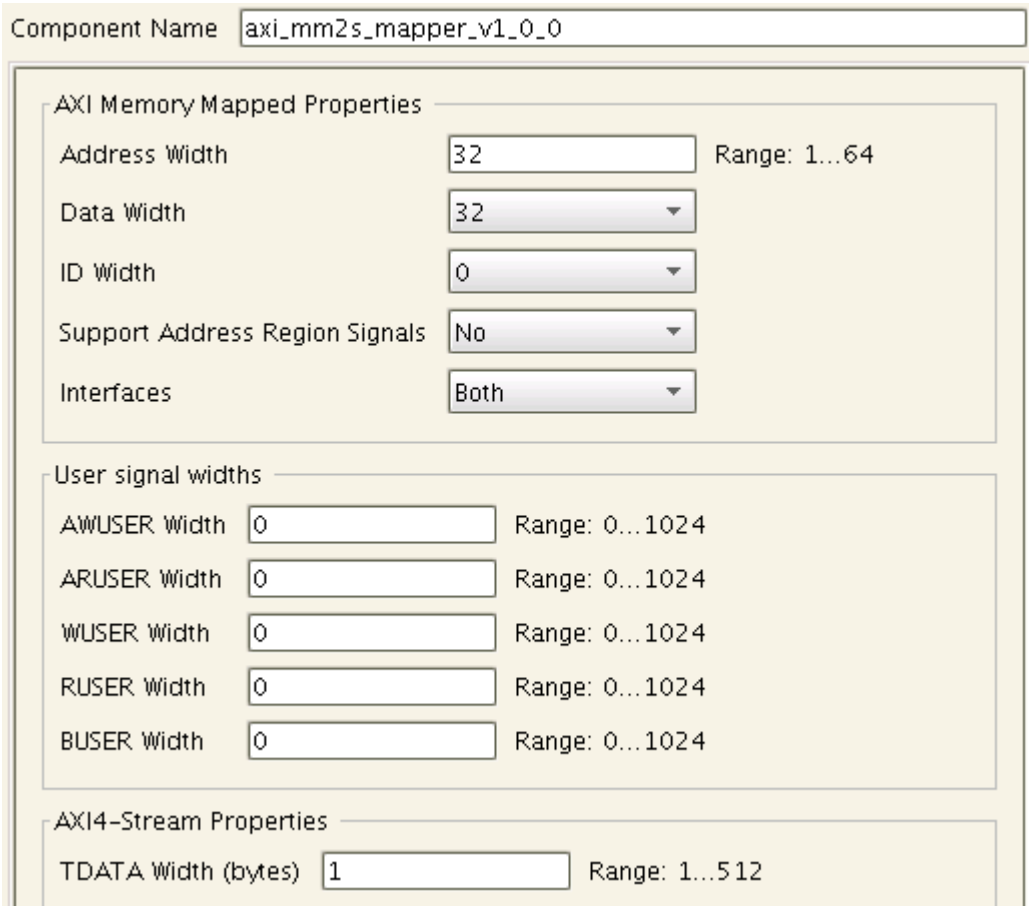

Figure 4-1: **Vivado IP Catalog: Customize IP**

AXI Memory Mapped Properties

Address Width

- Description: Width of all ADDR signals for S_AXI and M_AXI interfaces.
- Format/Range: Integer (12-64)
- Default Value: 32

Data Width

- Description: Width of all DATA signals for S_AXI and M_AXI interfaces.
- Format/Range: Integer (32, 64, 128, 256, 512, 1024)
- Default Value: 32

ID Width

- Description: Width of all ID signals on the S_AXI and M_AXI interfaces.

- Format/Range: Integer (0-32)
- Default Value: 0

Supports Address Region Signals

- Description: Adds `Region` signals to the S_AXI and M_AXI interfaces. This option will be removed in a future release.
- Format/Range: String (Yes, No)
- Default Value: No

RECOMMENDED: *Do not change this option.*

Interfaces

- Description: Specifies which AXI4-MM interfaces are present on the IP
- Format/Range: String (Both, M_AXI, S_AXI)
- Default Value: Both

User Signal Widths

AWUSER Signal Width

- Description: Width of `AWUSER` signals (if any) for all AXI4-MM interfaces.
- Format/Range: Integer (0-1024)
- Default Value: 0

ARUSER Signal Width

- Description: Width of `ARUSER` signals (if any) for all AXI4-MM interfaces.
- Format/Range: Integer (0-1024)
- Default Value: 0

WUSER Signal Width

- Description: Width of `WUSER` signals (if any) for all AXI4-MM interfaces.
- Format/Range: Integer (0-1024)
- Default Value: 0

RUSER Signal Width

- Description: Width of `RUSER` signals (if any) for all AXI4-MM interfaces.
- Format/Range: Integer (0-1024)
- Default Value: 0

BUSER Signal Width

- Description: Width of `BUSER` signals (if any) for all AXI4-MM interfaces.
- Format/Range: Integer (0-1024)
- Default Value: 0

AXI4-Stream Properties

TDATA Width (bytes)

- Description: Specifies the width in bytes of the `TDATA` signal on the M_AXIS and S_AXIS interfaces.
- Format/Range: Integer (1-512)
- Default Value: 1

Output Generation

For details, see "Generating IP Output Products" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 4].

Constraining the Core

There are no constraints associated with this core.

Simulation

This chapter contains information about simulating IP in the Vivado® Design Suite environment. 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 7].

Synthesis and Implementation

This chapter contains information about simulating and implementing in the Vivado® Design Suite environment.

For details about synthesis and implementation, see "Synthesizing IP" and "Implementing IP" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 4].

Example Design

The example design output product is available that demonstrates basic core functionality for the customized IP. The example design is an independent Vivado project populated with the customized IP along with additional IPs including another `axi_mm2s_mapper` instance, example master(s), example slave(s), clocking and reset blocks. A synthesizable top-level HDL file is provided that instantiates and wires together the IPs shown in Figure 8-1. If the parent Vivado project is configured for a Xilinx supported board, then the physical board constraints are also provided. A simulation-only demo test bench for the example design is also provided and discussed in further in the Chapter 6, Test Bench.

IMPORTANT: *The example design does not exhaustively demonstrate all the features of the IP, and is not a verification test bench.*

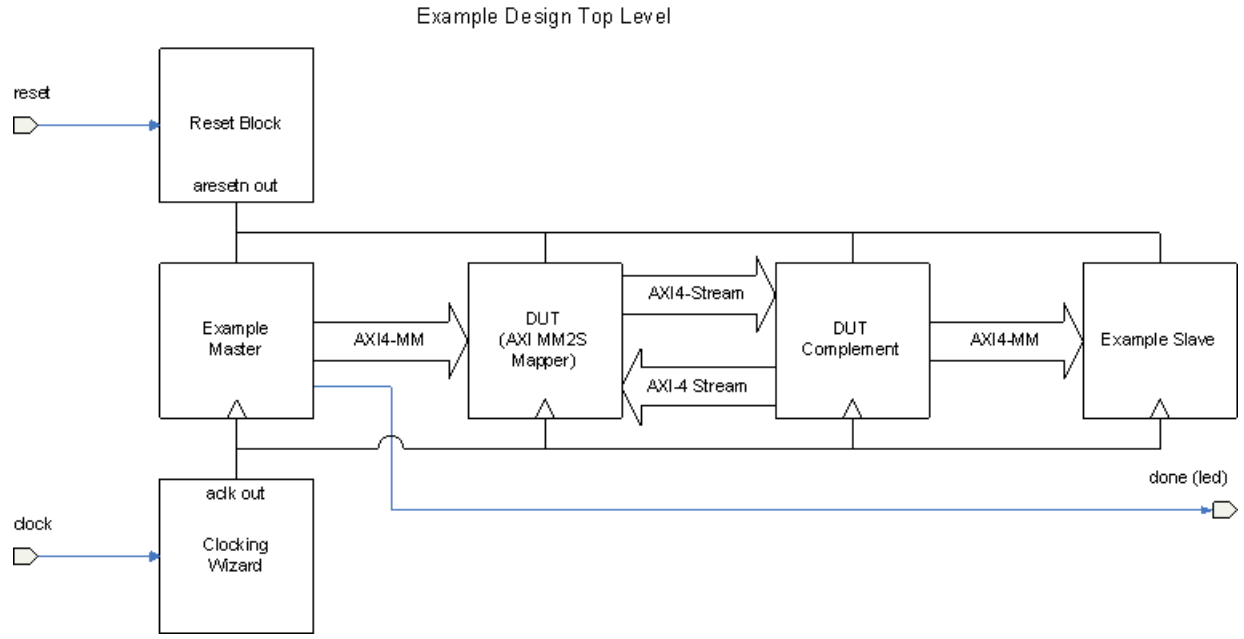

Figure 8-1: **Schematic of the Example Design Top Level**

Functionality

The example design shows basic functionality by instantiating a synthesizable AXI4 example master which sends transfers to an AXI4 example slave. The device under test

(axi_mm2s_mapper) is connected to its complement instantiation over AXI4-Stream and those are both connected to masters and slaves. After a fixed number of transfers, the master completes and asserts the done output signal. The example slave receives the transfers and performs a null operation on the payload to emulate data processing. After all transfers are received by the example slave the idle output signal is asserted. The example master and example slave generates a subset of AXI4-Stream protocol compliant transfers only. When the example master done output signal and the example slave idle output signal are both asserted then the done output pin is asserted. If this project is configured for a Xilinx reference board then the done signal is tied to an LED output.

A Clocking Wizard core is present to interface with an external differential clock input and to provide a clock output suitable for the design to easily meet timing closure. If the project is configured for a Xilinx reference board, then the Clocking Wizard is configured to specify the differential clock input constraints for the board.

A LogiCore Processor System Reset block is present to interface with an external reset input and to provide a de-bounced active Low system reset. If the project is configured for a Xilinx reference board, then the Processor System Reset (Proc Sys Reset) core is configured to specify the reset input constraint for the board.

Test Bench

A behavioral Verilog test bench that wraps around the example design top level is provided when the example design output product is generated. The test bench provides clocking and reset stimulus to the example design top level to run simulations on the example design. It monitors the done output to signal simulation completion. The test bench is useful for getting familiar with the signaling on the core by observing the simulation waveforms. The test bench will work with all simulation outputs from behavioral RTL through post-implementation timing.

In the example design, the simulation sources file set includes the test bench. To run the test bench, select the **Run Simulation** option in the **Vivado Flow Navigator**. Once the simulation is open issue the "run all" command to run the simulation to completion. Output similar to code shown below will be seen if the simulation completes successfully.

```
 1937.60ns: exdes_tb: Starting testbench
 2017.60ns: exdes_tb: Asserting reset for 16 cycles
 2097.60ns: exdes_tb: Reset complete
68480.00ns: exdes_tb: SIMULATION PASSED
68480.00ns: exdes_tb: Test Completed Successfully
```

For more information with running the simulation test bench, see the *Vivado Design Suite User Guide: Getting Started* (UG810) [Ref 6].

Migrating and Upgrading

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

Migrating to the Vivado Design Suite

For information about migrating to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 3].

Upgrading in the Vivado Design Suite

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

Parameter Changes

There are no parameter changes.

Port Changes

There are no port changes.

Other Changes

There are no other changes.

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 AXI Memory Mapped to Stream Mapper, the Xilinx Support web page (Xilinx Support web page) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for opening a Technical Support WebCase.

Documentation

This product guide is the main document associated with the AXI Memory Mapped to Stream Mapper. 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 are listed below, and can also be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool message(s)

- Summary of the issue encountered

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

Answer Records for the AXI Memory Mapped to Stream Mapper Core

AR 54420

Technical Support

Xilinx provides technical support in 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.

Xilinx provides premier technical support for customers encountering issues that require additional assistance.

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

1. Open a WebCase by selecting the WebCase link located under Support Quick Links.

- A block diagram of the video system that explains the video source, destination and IP (custom and Xilinx) used.

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

Debug Tools

There are many tools available to address AXI Memory Mapped to Stream Mapper 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.

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

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

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

Reference Boards

Various Xilinx development boards support AXI Memory Mapped to Stream Mapper. These boards can be used to prototype designs and establish that the core can communicate with the system.

- 7 series FPGA evaluation boards
  - KC705
  - KC724

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The ChipScope debugging tool is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the ChipScope debugging tool for debugging the specific problems.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `LOCKED` port.
- If your outputs go to 0, check your licensing.

Interface Debug

AXI4-Stream Interfaces

If data is not being transmitted or received, check the following conditions:

- If transmit `m_axis_tready` is stuck low following the `s_axis_tvalid` input being asserted, the core cannot send data.
- If the receive `s_axis_tvalid` is stuck low, the core is not receiving data.
- Check that the `ACLK` inputs are connected and toggling.
- Check core configuration.
- Add appropriate core specific checks.

Additional Resources

Xilinx Resources

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

Xilinx Support web page.

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

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

References

These documents provide supplemental material useful with this product guide:

1. ARM® AMBA® AXI Protocol v2.0 (ARM IHI 0022C)
2. *AMBA AXI4-Stream Protocol Specification*
3. *ISE to Vivado Design Suite Migration Guide* (UG911)
4. *Vivado Design Suite User Guide: Designing with IP* (UG896)
5. *Vivado Design Suite User Guide: Programming and Debugging* (UG908).
6. *Vivado Design Suite User Guide: Getting Started* (UG810)
7. *Vivado Design Suite User Guide: Logic Simulation (*UG900*)*
8. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/05/2017	1.1	Removed loopback testing from IP Facts.
04/04/2016	1.1	Removed references to Aurora protocol.
11/18/2015	1.1	Added UltraScale+ support.
04/01/2015	1.1	Updated Master AR.
10/02/2013	1.1	Added Test Bench and Example Design chapters.
03/20/2013	1.0	Initial Xilinx release. Originally released as Early Access for 2012.4 in 12/18/2012.

Notice of Disclaimer