AXI4-Stream Interconnect v1.1

LogiCORE IP Product Guide

Vivado Design Suite

PG035 October 4, 2017

XILINX
ALL PROGRAMMABLE™

Table of Contents

Appendix C: Additional Resources and Legal Notices

Introduction

The AXI4-Stream Interconnect is a key interconnect infrastructure IP core that enables the connection of heterogeneous master/slave AMBA® AXI4-Stream protocol compliant endpoint IP. The AXI4-Stream Interconnect routes connections from one or more AXI4-Stream master channels to one or more AXI4-Stream slave channels.

Features

The AXI4-Stream Interconnect IP provides the following capabilities:

- Configurable multiple master to multiple slave (up to 16x16) capable cross-point switch
- Arbitrary TDATA byte width conversion
- Synchronous and asynchronous clock rate conversion
- Configurable datapath FIFO buffers including store and forward (packet) capable FIFOs
- Optional register slice at boundaries to ease timing closure
- Support for multiple clock domains

LogiCORE™ IP Facts Table	
Core Specifics	
Supported Devices(1)	UltraScale+™ UltraScale™ Zynq®-7000, Virtex®-7, Kintex®-7, Artix®-7, Spartan®-7
Supported User Interfaces	AXI4-Stream
Resources	See Performance and Resource Utilization.
Provided with Core	
Design Files	Verilog
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Xilinx Design Constraints (XDC)
Simulation Model	Behavioral Verilog
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 ARM® AMBA® 4 specification builds on the AMBA 3 specifications by adding new interface protocols to provide greater interface flexibility in designs with open standards. The AXI4-Stream interface supports low resource, high bandwidth unidirectional data transfers. It is well-suited for FPGA implementation because the transfer protocol allows for high frequency versus clock latency trade-offs to help meet design goals.

The AXI4-Stream protocol is derived from the single AXI3 write channel. It has no corresponding address or response channel and is capable of non-deterministic burst transactions (undefined length). The protocol interface signal set adds optional signals to support data routing, end of transaction indicators, and null-beat modifiers to facilitate management and movement of data across a system. These characteristics are suitable for transferring large amounts of data efficiently while keeping gate count low.

The protocol interface consists of a master interface and a slave interface. The two interfaces are symmetric and point-to-point, such that the master interface output signals can connect directly to the slave interface input signals. Utilizing this concept, it is possible to design AXI4-Stream modules that have a slave interface input channel and a master interface output channel. Because the master and slave interfaces are symmetric, any number of these modules can be daisy-chained together by connecting the master interface output channel of one module to the slave interface input channel of another module and so on. The function of the modules can be a multitude of different options such as buffering, data transforming, or routing.

The AXI4-Stream Interconnect IP is a powerful collection of modules that provides a rich set of functions to interconnect AXI4-Stream masters and slaves. The IP core is capable of performing data switching/routing, data width conversion, pipelining, clock conversion, and data buffering. Parameters and IP configuration graphical user interfaces (GUIs) are used to configure the core to suit each of the requirements of the system designer.

Feature Summary

- AXI4-Stream compliant
  - Supports all AXI4-Stream defined signals: TVALID, TREADY, TDATA, TSTRB, TKEEP, TLAST, TID, TDEST, and TUSER
    - TDATA, TSTRB, TKEEP, TLAST, TID, TDEST, and TUSER are optional
    - Programmable TDATA, TID, TDEST, and TUSER widths (TSTRB and TKEEP width is TDATA width/8)
  - Per port ACLK/ARESETn inputs (supports clock domain crossing)
  - Per port ACLKEN inputs (optional)
- Core switch
  - 1-16 masters
  - 1-16 slaves
  - Full slave side arbitrated crossbar switch
  - Slave input to master output routing based on TDEST value decoding and comparison against base and high value range settings
  - Round-Robin and Priority arbitration
  - Arbitration suppress capability to prevent head-of-line blocking
  - Native switch data width 8, 16, 24, 32, 48, ... 4,096 bits (any byte width up to 512 bytes)
  - Arbitration tuning parameters to arbitrate on TLAST boundaries, after a set number of transfers, and/or after a certain number of idle clock cycles
  - Optional pipeline stages after internal TDEST decoder and arbiter functional blocks
  - Programmable connectivity map to specify full or sparse crossbar connectivity
- Built in data width conversion
- Each master and slave connection can independently use data widths of 8, 16, 24, 32, 48, ... 4,096 bits (any byte width up to 512 bytes)
- Built-in clock-rate conversion
  - Each master and slave connection can use independent clock rates.
  - Synchronous integer-ratio (N:1 and 1:N) conversion to the internal crossbar native clock rate.
  - Asynchronous clock conversion (uses more storage and incurs more latency than synchronous conversion).

- Optional register-slice pipelining
  - Available on each AXI4-Stream channel connecting to each master and slave device
  - Facilitates timing closure by trading-off frequency versus latency
  - One latency cycle per register-slice, with no loss in data throughput in the register slice under all AXI4-Stream handshake conditions
- Optional datapath FIFO buffering:
  - Available on datapaths connecting to each master and each slave.
  - 16, 32, 64, 128,...32,768 deep (16-deep and 32-deep are LUT-RAM based, otherwise are block RAM based)
  - Normal and Packet FIFO modes (Packet FIFO mode is also known as store-and-forward in which a packet is stored and only released downstream after a TLAST packet boundary is detected.)
  - FIFO data count outputs to report FIFO occupancy
- Additional Error Flags
- Error flags to detect conditions such as: `TDEST` decode error, sparse `TKEEP` removal, and packer error.

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

The AXI4-Stream Interconnect core is a collection of submodules centered around the AXI4-Stream Switch. There are submodules grouped together in datapaths before and after the switch that allow for data manipulation and flow control. Each individual submodule consists of AXI4-Stream protocol compliant master and slave interfaces. A block diagram of the AXI4-Stream Interconnect is shown in Figure 2-1.

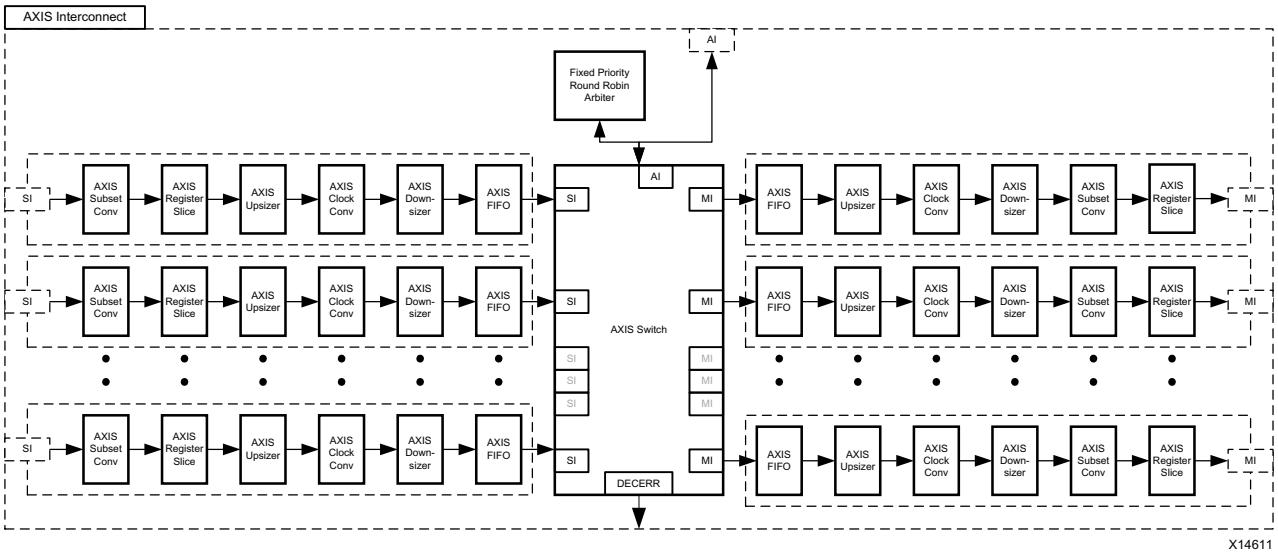

Figure 2-1: **AXI4-Stream Interconnect Block Diagram**

The AXI4-Stream Switch supports up to 16 masters to 16 slaves in a full or sparse crossbar configuration using the AXI4-Stream signal `TDEST` as the routing designator.

As shown in Figure 2-1 an AXI4-Stream Master can connect to the Slave Interface (SI) of the AXI4-Stream Interconnect. Similarly, an AXI4-Stream Slave can connect to the Master Interface (MI) of the AXI4-Stream Interconnect. The AXI4-Stream Switch supports sparse configurations, SI side decoding, MI side arbitration (one arbiter per MI port), and various arbitration modes.

The datapath consists of submodules that contain one SI port and one MI port chained together between the external SI bus to the switch or from the switch to the external MI bus. The datapath allows for data width conversion, buffering, and clock conversion to and from the AXI4-Stream Switch. All submodules including the AXI4-Stream Switch are optionally instantiated based on parameters to allow full flexibility in design.

The submodules within the AXI4-Stream Interconnect are presented in Table 2-1.

Table 2-1: **Submodules within the AXI4-Stream Interconnect**

Submodule Name	Description	Inferred or Explicit instantiation
Subset Converter	Adds or removes optional signals from an AXI4-Stream bus interface.	Limited inference. Infers a TKEEP internally to the interconnect if there is a data width converter and TID or TDEST or TLAST signals are present.
Register Slice	Provides a mechanism to bridge connections between Masters/Slaves within different ACLK domains.	Inferred
Data Width Converter	Allows expansion of the AXI4-Stream TDATA width by aggregating multiple transfers into one transfer or allows reduction of the AXI4-Stream TDATA width by splitting a transfer into multiple transfers of smaller TDATA width.	Inferred
Data FIFO	Provides AXI4-Stream data storage.	Explicit
Switch	Allows routing from multiple masters to multiple slaves.	Inferred when there is more than one master or more than one slave in the system.

AXI4-Stream Switch and Arbiter

The AXI4-Stream Switch supports 1:N, M:1, and M:N configurations. It connects up to 16 masters to 16 slaves. The AXI4-Stream `TDEST` signaling is required for 1:N and M:N configurations.

The AXI4-Stream Switch only supports static routing through fixed base-high `TDEST` ranges for each MI. A single `TDEST` map applies to all MIs. Each MI is arbitrated independently (slave side arbitration). The AXI4-Stream Switch performs SI-side parallel decoding. Unmapped TDEST transfers will signal a decode error and drop the transfer.

The internal arbiter can perform fixed priority arbitration or round robin arbitration. The Arbiter/Switch can arbitrate on a per transfer basis or at packet boundaries (signaled by `TLAST` or after a configurable number of active or idle transfers.)

AXI4-Stream Clock Converter

Clock converters are necessary in the AXI4-Stream protocol for converting masters operating at different clock rates to slaves. Typically, the AXI4-Stream Interconnect should be clocked at the same rate as the fastest slave; devices not running at that same rate need to be converted. Synchronous clock converters are ideal because they have the lowest latency and smaller area. However, they are only viable if both clocks are phase-aligned, integer clock ratios, and the fmax requirements are able to be met. Asynchronous clock converters are a generic solution able to handle both synchronous/asynchronous clocks with arbitrary phase alignment. The trade-off is that there is a significant increase in area and latency associated with asynchronous clock converters. If global clock enables are configured, additional logic is generated to handle clock enables independently for each clock domain. There is a clock converter module available in every datapath and it is instantiated if either the clocks are specified as asynchronous or have different synchronous clock ratios. The clock converter module performs the following functions:

- A clock-rate reduction module performs integer (N:1) division of the clock rate from its input (SI) side to its output (MI) side.
- A clock-rate acceleration module performs integer (1:N) multiplication of clock rate from its input (SI) to output (MI) side.
- Asynchronous clock rate conversion between the input and output uses an internal FIFO Generator instantiated module.
- Clock Enable crossing logic that handles different `ACLKEN` signals per clock domain.

Figure 2-2 shows the clock converter with support for independent `ACLKEN` signals on its SI and MI.

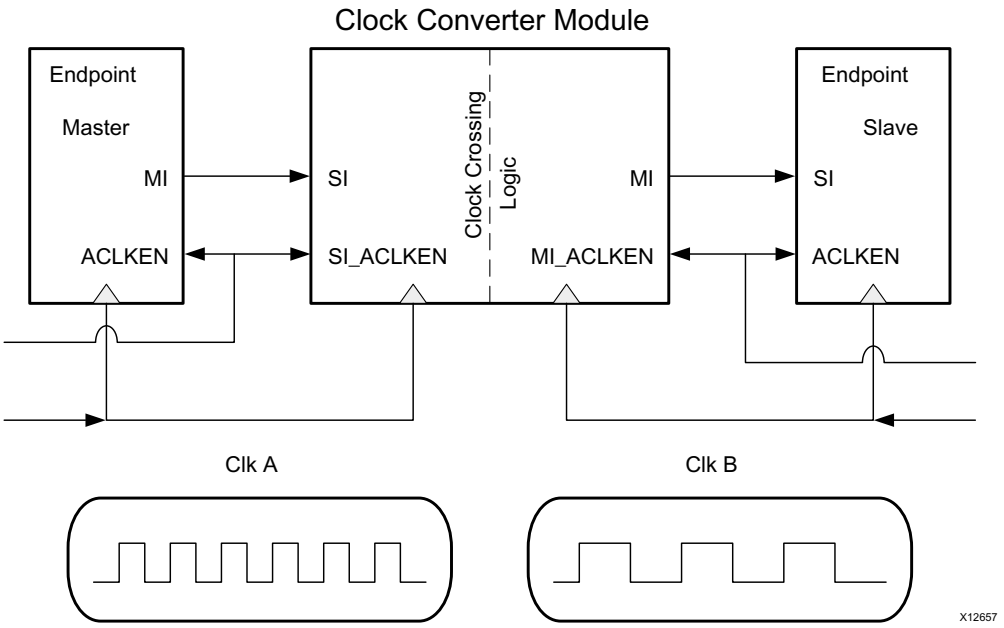

Figure 2-2: **Clock Converter Module Block Diagram**

AXI4-Stream Data Width Converter

Data width converters (upsizer/downsizer) are required when interfacing different data width cores with each other. One data width conversion module is available to handle all supported combinations of data widths.

The conversion follows the AMBA® AXI4-Stream Protocol Specification with regards to ordering and expansion of TUSER bits. The width converter does not process any special TUSER encoding formats; it only maps TUSER bits across the width conversion function using the algorithm specified in the AXI4-Stream protocol specification. Depending on the usage/meaning of TUSER to the endpoint IP, additional external logic might be required to manipulate TUSER bits that have been transformed by the width converter. The number of TUSER bits per TDATA bytes must remain constant between input and output.

Up-conversion requires that each incoming beat that is composed of the new larger beat consists of identical TID and TDEST bits and no intermediate TLAST assertions. Partial data might be flushed when either the TLAST bit is received or TID/TDEST changes before enough data is accumulated to send out a complete beat. Unassigned bytes are flushed out as null bytes.

RECOMMENDED: *Xilinx recommends that the* TKEEP *signal output is monitored if the* TID/TDEST/TLAST *signal is present.*

Any non-integer multiple byte ratio conversion (N:M) is accomplished by calculating the lowest common multiple (LCM) of N and M and then up-converting from N:LCM then down-converting from LCM:M.

Up-conversion features: Range: Input 1-256 bytes, Output 2-512 bytes

- Supports full range of 1:N byte ratio conversions
- Minimum latency of 2 + N clock cycles in 1:N byte ratio up-conversion.

Down-conversion features:

- Range: Input 2-512 bytes, output 256-1 bytes
- Supports full range of N:1 byte ratio conversions
- Minimum Latency: two clock cycles

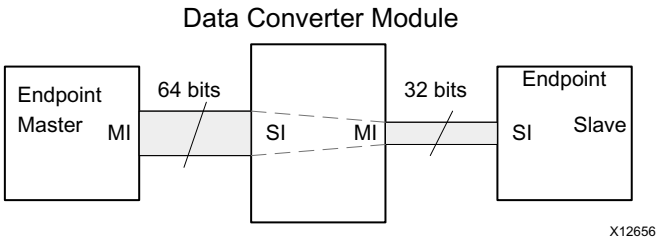

Figure 2-3: **Data Width Converter (Down Conversion) Module Block Diagram**

AXI4-Stream Register Slice

The register slice is a multipurpose pipeline register that is able to isolate timing paths between master and slave. The register slice is designed to trade-off timing improvement with area and latency necessary to form a protocol compliant pipeline stage. Implemented as a two-deep FIFO buffer, the register slice supports throttling by the master (channel source) and/or slave (channel destination) as well as back-to-back transfers without incurring unnecessary idle cycles. The module can be independently instantiated at all port boundaries.

AXI4-Stream Data FIFO Buffer

The FIFO module is capable of providing temporary storage (a buffer) of the AXI4-Stream data. The FIFO Buffer module should be used in between two endpoints when:

- More buffering than a register slice is desired.
- Store and forward: to accumulate a certain number of bytes from the master before forwarding them to the slave (packet mode.)

The FIFO module can also implement asynchronous clock conversion so when asynchronous clock conversion and FIFOs are enabled on the same interface, redundant FIFOs are not instantiated. The FIFO module uses the Xilinx® LogiCORE™ IP FIFO Generator module. This supports native AXI4-Stream with the following features:

- Variable FIFO depths
- FIFO data widths from 8 to 1,024 bits

  1,024-bit FIFO data width limit is a FIFO Generator restriction that limits the payload width of the transfer being buffered in the FIFO. This limit might be removed in a future version of FIFO Generator, at which point the AXIS FIFO also supports FIFO buffering for transfers with payloads including TDATA widths of up to 4,096 bits. The width of the FIFOs are determined by the width of the SWITCH component.
- Independent or common clock domains
- Symmetric aspect ratios
- Asynchronous active-Low reset
- Selectable memory type. The memory type is inferred as distributed block RAM for depths of 32 or less and block RAM for all others.
- Operates in First-Word Fall-Through mode (FWFT)
- Occupancy interface
- Both FIFO Generator rd_data_count and wr_data_count are passed as separate outputs synchronized to the read side and write side clock domains.

Standards

This core has bus interfaces that comply with the ARM® AMBA AXI4-Stream Protocol Specification Version 1.0.

Performance

The performance of the AXI4-Stream Interconnect core is limited only by the FPGA logic speed. The core utilizes only block RAMs, LUTs, and registers and contains no I/O elements. The values presented in this section should be used as an estimation guideline; actual performance can vary.

Maximum Frequencies

For details about performance, visit Performance and Resource Utilization.

Latency

The latency in the IP core can vary on an interface-to-interface basis, depending on how the IP core is configured. The latency is calculated in clocked cycles and is measured as the time that it takes from the assertion of the slave interface TVALID signal to the first assertion of the master interface TVALID signal. The latency for each of the individual submodules is listed in Table 2-2. To obtain the minimum latency for your system, you must add up the values shown in the Table 2-2. The latency specifications assume that the master interface TREADY signal input is always asserted. The back-to-back delay is the number of clock cycles that back-to -back transfers can be accepted by the module. This can be observed by counting how many cycles slave interface TREADY is Low after a transfer is accepted on the interface.

Table 2-2: **Latency by Module Type**

Module Type	Latency (clocks)	Back-To-Back Delay (clocks)	Description
Register Slice	1	0	Adding a register slice always adds one cycle of latency. There is no back-to-back delay.
Data Width Converter (upsizer)	[data width ratio]	0	The latency varies based on the data width ratio. For the slave interface datapath, the ratio is from the slave interface data width to the switch data width. For the master interface datapath, the ratio is from the switch data width to the master interface data width. Example: If a 1:4 byte data converter is used, the latency of the module will be 4 clock cycles.
Data Width Converter (downsizer)	1	[data width ratio]-1	The back-to-back delay varies based on the data width ratio. For the slave interface datapath, the ratio is from the slave interface data width to the switch data width. For the master interface datapath, the ratio is from the switch data width to the master interface data width. Example: If a 2:1 byte data converter is used, then the module can only accept transfers every other cycle.

Table 2-2: **Latency by Module Type *(Cont'd)***

Module Type	Latency (clocks)	Back-To-Back Delay (clocks)	Description
Synchronous Clock Converter (speed-up)	1	0	The synchronous clock converter latency is reported as units of the slave interface clock.
Synchronous Clock Converter (speed-down)	1	[clock ratio]-1	The synchronous clock converter latency is reported as units of the slave interface clock. Back-to-back delay varies based on the clock ratio. Example: If using a synchronous 150 MHz-to-50 MHz 3:1 clock converter (clock ratio of 3), the back-to-back delay will be 2 clock cycles.
Asynchronous Clock Converter	Not Defined	0	The latency associated with an asynchronous clock converter can vary greatly depending on the clocks. It can be expected to see latencies of 5 clock cycles or more. See the *FIFO Generator LogiCORE IP Product Guide* (PG057) [Ref 1] for more details.
FIFO Generator AXI4-Stream (Normal) FIFO	3	0	The FIFO when configured in normal mode will output data as soon as it is possible. See the *FIFO Generator LogiCORE IP Product Guide* (PG057) [Ref 1] for more details.
FIFO Generator AXI4-Stream Packet FIFO	Until TLAST is received or FIFO is full.	0	The FIFO when configured in packet mode will output data only when a TLAST is received or the FIFO has filled. See the *FIFO Generator LogiCORE IP Product Guide* (PG057) [Ref 1] for more details.
Crossbar Switch	2	0	The output latency of the switch under ideal conditions is 2 clock cycles. There is 1 cycle of latency for the TDEST decode and 1 cycle of latency for the arbiter grant (if idle) to make up the 2 cycles listed in the table. If the output pipeline is enabled, add 1 cycle. The back-to-back delay is defined in this case for an already granted arbitration. Back-to back-arbitration will result in 1 cycle delays between transactions. To reduce arbitration cycles, set the **Arbitrate on maximum number of transfers** parameter to a higher number, or set **Arbitrate on TLAST** to **Yes** if the design allows for it.

Throughput

The throughput of a datapath through the AXI4-Stream Interconnect is calculated as *TDATA width x clock frequency* of each of the paths determined by the SI interface, the switch, and the MI interface. The minimum throughput of an individual path for which the transfer traverses determines the overall throughput of the datapath.

For example, a 2x2 configured AXI4-Stream Interconnect is configured as follows:

- S00 interface at 256 bits x 250 MHz (64,000 Mb/s)
- S01 interface at 128 bits x 100 MHz (12,800 Mb/s)
- Switch at 256 bits at 200 MHz (51,200 Mb/s)
- M00 interface at 512 bits x 250 MHz (128,000 Mb/s)
- M01 interface at 32 bits x 200 MHz (6,400 Mb/s)

Calculating the max throughput for each of the paths:

- S00->Switch->M00: 51,200 Mb/s
- S01->Switch->M00: 12,800 Mb/s
- S00->Switch->M01: 51,200 Mb/s
- S01->Switch->M01: 6,400 Mb/s

The slowest theoretical throughput of the system can be seen as that path from S01->Switch->M01 as limited by the throughput calculated for the M01 interface. The switch is capable of performing multiple transfers simultaneously between masters and slaves; therefore, total simultaneous throughput is calculated as S00->Switch->M00 + S01->Switch->M01 throughputs or 57,600 Mb/s.

RECOMMENDED: *The arbiter inserts dead cycles on back-to-back transfers from the same slave interface to the master interface. Therefore, reduce the switch arbitration events by setting* ***Arbitrate on TLAST transfer*** *to* ***Yes****, or by increasing the* ***Arbitrate on maximum number of transfers*** *parameter.*

Resource Utilization

For details about resource utilization, visit Performance and Resource Utilization.

Port Descriptions

Global Signals

The global signals listed in Table 2-3 are always present on every configuration.

Table 2-3: **Global Signals**

Signal	Direction	Description
ACLK	Input	Global Clock Signal. Drives the clocks on the AXI4-Stream Switch and is the primary clock to the system.
ARESETN	Input	Global Reset Signal. This active-Low signal drives the reset pins on the AXI4-Stream Switch and is the primary reset of the system.
ACLKEN	Input	Global ACLK Enable signals. Drives the ACLKEN pins on the AXI4-Stream Switch and is the primary ACLKEN of the system.

Slave Interface Signals

Table 2-4 lists the signals associated with the Slave Interface buses. The number of buses is configurable and the signals in Table 2-4 are replicated for each port. The *nn* denoted for the signals starts at 00 and increments by one up to 15 for each Slave Interface bus instantiated.

Table 2-4: **Signals Associated with the Slave Interface Buses**

Signal	Direction	Description
Snn_AXIS_ACLK	Input	Clock signal. All inputs/outputs of this bus interface are rising edge aligned with this clock.
Snn_AXIS_ARESETN	Input	Active-Low synchronous reset signal
Snn_AXIS_ACLKEN	Input	Clock enable signal
Snn_AXIS_TVALID(1)	Input	TVALID indicates that the master is driving a valid transfer. A transfer takes place when both TVALID and TREADY are asserted.
Snn_AXIS_TREADY(1)	Output	TREADY indicates that the slave can accept a transfer in the current cycle.
Snn_AXIS_TDATA [(C_Mnn_AXIS_TDATA_WIDTH-1):0](1)	Input	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.

Table 2-4: **Signals Associated with the Slave Interface Buses *(Cont'd)***

Signal	Direction	Description
Snn_AXIS_TSTRB [((C_Mnn_AXIS_TDATA_WIDTH/8)-1):0][(1)]	Input	TSTRB is the byte qualifier that indicates whether the content of the associated byte of TDATA is processed as a data byte or a position byte.
Snn_AXIS_TKEEP [((C_Mnn_AXIS_TDATA_WIDTH/8)-1):0][(1)]	Input	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 data stream.
Snn_AXIS_TLAST[(1)]	Input	TLAST indicates the boundary of a packet.
Snn_AXIS_TID [C_NATIVE_TID_WIDTH-1:0][(1)]	Input	TID is the data stream identifier that indicates different streams of data.
Snn_AXIS_TDEST [(C_NATIVE_TDATA_WIDTH-1):0][(1)]	Input	TDEST provides routing information for the data stream.
Snn_AXIS_TUSER [(C_Snn_AXIS_TUSER_WIDTH-1):0][(1)]	Input	TUSER is user-defined sideband information that can be transmitted alongside the data stream.
Snn_ARB_REQ_SUPPRESS	Input	Active-High signal to skip this bus on the next arbitration cycle. While the signal is asserted, this bus does not receive the next arbitration. If this bus already has arbitration granted, it remains granted until the arbitration cycle is completely normally.
Snn_DECODE_ERR	Output	One-hot output indicates that a incoming transfer has a TDEST value that did not map to a valid Master Interface. Invalid TDEST transfers are dropped.
Snn_SPARSE_TKEEP_REMOVED	Output	Not Implemented
Snn_PACKER_ERR	Output	Not Implemented
Snn_FIFO_DATA_COUNT[31:0]	Output	Indicates the write count inside the DATA FIFO. Does not produce valid output when using Packet Mode FIFO.

Notes:

1. This signal description is taken from the ARM AMBA Protocol Specification.

Master Interface Signals

Table 2-5 lists the signals associated with the Master Interface buses. The number of buses is configurable and the signals are replicated for each port. The *nn* denoted for the signals starts at 00 and increments by one up to 15 for each Master Interface bus instantiated.

Table 2-5: **Signals Associated with the Master Interface Buses**

Signal	Direction	Description
Mnn_AXIS_ACLK	Input	Clock signal. All inputs/outputs of this bus interface are rising edge aligned with this clock.
Mnn_AXIS_ARESETN	Input	Active-Low synchronous reset signal
Mnn_AXIS_ACLKEN	Input	Clock enable signal
Mnn_AXIS_TVALID(1)	Output	TVALID indicates that the master is driving a valid transfer. A transfer takes place when both TVALID and TREADY are asserted.
Mnn_AXIS_TREADY(1)	Input	TREADY indicates that the slave can accept a transfer in the current cycle.
Mnn_AXIS_TDATA [(C_Mnn_AXIS_TDATA_WIDTH-1):0](1)	Output	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.
Mnn_AXIS_TSTRB [((C_Mnn_AXIS_TDATA_WIDTH/8)-1):0](1)	Output	TSTRB is the byte qualifier that indicates whether the content of the associated byte of TDATA is processed as a data byte or a position byte.
Mnn_AXIS_TKEEP [((C_Mnn_AXIS_TDATA_WIDTH/8)-1):0](1)	Output	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 data stream.
Mnn_AXIS_TLAST(1)	Output	TLAST indicates the boundary of a packet.
Mnn_AXIS_TID [C_NATIVE_TID_WIDTH-1:0](1)	Output	TID is the data stream identifier that indicates different streams of data.
Mnn_AXIS_TDEST [(C_NATIVE_TDATA_WIDTH-1):0](1)	Output	TDEST provides routing information for the data stream.
Mnn_AXIS_TUSER [(C_Snn_AXIS_TUSER_WIDTH-1):0](1)	Output	TUSER is user-defined sideband information that can be transmitted alongside the data stream.

Table 2-5: **Signals Associated with the Master Interface Buses *(Cont'd)***

Signal	Direction	Description
Mnn_SPARSE_TKEEP_REMOVED	Output	One-hot output indicates that at least one Null Byte was introduced into TDATA due to width conversion, but because the TKEEP signal is not present, the Null Byte will not be detected by the downstream slave.
Mnn_PACKER_ERR	Output	Not Implemented
Mnn_FIFO_DATA_COUNT[31:0]	Output	Indicates the write count inside the DATA FIFO. Does not produce valid output when using Packet Mode FIFO.

Notes:

1. This signal description is taken from the ARM AMBA Protocol Specification.

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the AXI4-Stream Interconnect core.

General Design Guidelines

The first step is to establish the topology of the system. This requires an understanding of the main interface characteristics of each AXI4-Stream master and slave that needs to communicate with each other. AXI4-Stream masters and slaves then need to be grouped by desired connectivity into a system with one or more AXI4-Stream Interconnect blocks, tying them together so they can exchange data.

In general, try to establish the system partitioning/topology to use multiple smaller/simpler interconnects than a single large interconnect, especially in systems with a large number of devices. For example, combinations of N master x 1 slaves interconnects and 1 master x N slave interconnects are generally preferable to MxN interconnects in terms of area, latency, and throughput. MxN interconnect when required for performance or connectivity requirements should try to limit the number of endpoints or specify sparse connectivity to reduce resource utilization.

The Xilinx *AXI Reference Guide* (UG1037) [Ref 2] provides information about AXI4-Stream protocol usage guidelines and conventions; much of the AXI system optimizations information described for AXI Interconnect is applicable to the AXI4-Stream Interconnect. The Xilinx *AXI Reference Guide* should be reviewed before designing or structuring systems around the AXI4-Stream Interconnect.

After the number and topology of AXI4-Stream Interconnects has been determined, the next step is to tailor each AXI4-Stream Interconnect to have the correct set of optional interface signals and set signals widths as needed. This sets up the interface signal set for the AXI4-Stream Interconnect to ensure that data can be exchanged and routed as needed across the system.

The Interconnect should be optimized and fine-tuned to fit its application. This includes tuning FIFOs, width converters, clock converters, arbiters, and register slices (pipeline stages) as needed to balance area, timing, performance, and ease-of-use.

Clocking

The AXI4-Stream Interconnect provides clock inputs for every SI slot, MI slot, and for the core switch inside the IP. The clock inputs for all active interfaces including the core switch must be driven. Even if the AXI Interconnect is configured for synchronous 1:1 clocking across the interconnect, ensure that all clock inputs for enabled interfaces are driven.

The AXI4-Stream Interconnect supports per interface selection of optional clock converters. This allows the AXI4-Stream Interconnect to be connected to the endpoint AXI4-Stream Interconnect IP operating in different clock domains. Synchronous clock converters can be used when the endpoint IP has a phase-aligned, integer multiple clock ratio to the core switch clock. This is often the case when the same mixed-mode clock manager (MMCM) or phase-locked loop (PLL) is driving synchronous integer ratio clocks because the MMCM/PLL can also ensure phase alignment across their clock outputs. An asynchronous clocking mode allows the attached endpoint IP to run at a completely unrelated clock frequency or phase to the core switch clock. Embedded asynchronous FIFOs from the FIFO Generator are instantiated inside the AXI4-Stream Interconnect to handle data transfers across asynchronous clock domains in a robust manner.

An optional feature for clock enables (ACLKEN ports) allows an extra level of control for gating clocks. The clock enable signals can be used to control which clock edges are seen as real transfer cycles. Clock enables can be used for purposes like preserving global clock buffers, debug by stepping the clock enable to step through operational states in the system, and dynamic power savings. Clock enables can be controlled independently on each interface port and for the core switch.

Resets

The AXI4-Stream Interconnect provides active-Low reset inputs for every SI slot, MI slot, and for the core switch inside the IP. Each reset input must be synchronized to the associated ACLK input of the interface. The AXI4-Stream Interconnect requires that all reset inputs externally be synchronized to their respective clock domains as necessary to prevent metastability. To ensure data is not lost during reset deassertion across multiple interfaces of the AXI4-Stream Interconnect (operating in potentially different clock domains), the AXI4-Stream Interconnect deasserts all TREADY and TVALID outputs until the clock cycle after their source logic has internally exited reset. Any endpoint IP driving TREADY or TVALID inputs to the AXI4-Stream Interconnect should also deassert these signals until the clock cycle after they have exited reset internally.

These guidelines ensure that endpoint IP cores can internally come out of reset at different times (due to internal reset pipelining) and no data will be exchanged until both are internally out of reset.

RECOMMENDED: *Xilinx recommends that any endpoint deassert* TREADY *and* TVALID *within 8 clock cycles of reset assertion.* ARESETn *should also be asserted for at least 16 cycles of the slowest system clock to ensure that all AXI interfaces in the system enter reset and have time to deassert their* TREADY/TVALID *outputs before coming back out of reset.*

.

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 can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 3]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 4]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 6]

Note: This core is not available in the IP integrator.

Customizing and Generating the Core

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

After creating the project, the IP core is available for selection in the IP catalog, located at **AXI Infrastructure > AXI4-Stream Interconnect**.

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

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

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 4] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5].

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

Locate the IP core in the Vivado Design Suite and click it once to select it. Details regarding the solution are displayed in the Details window. To configure the IP, double-click the AXI4-Stream Interconnect row in the IP catalog.

This launches the customization dialog box, shown in Figure 4-1.

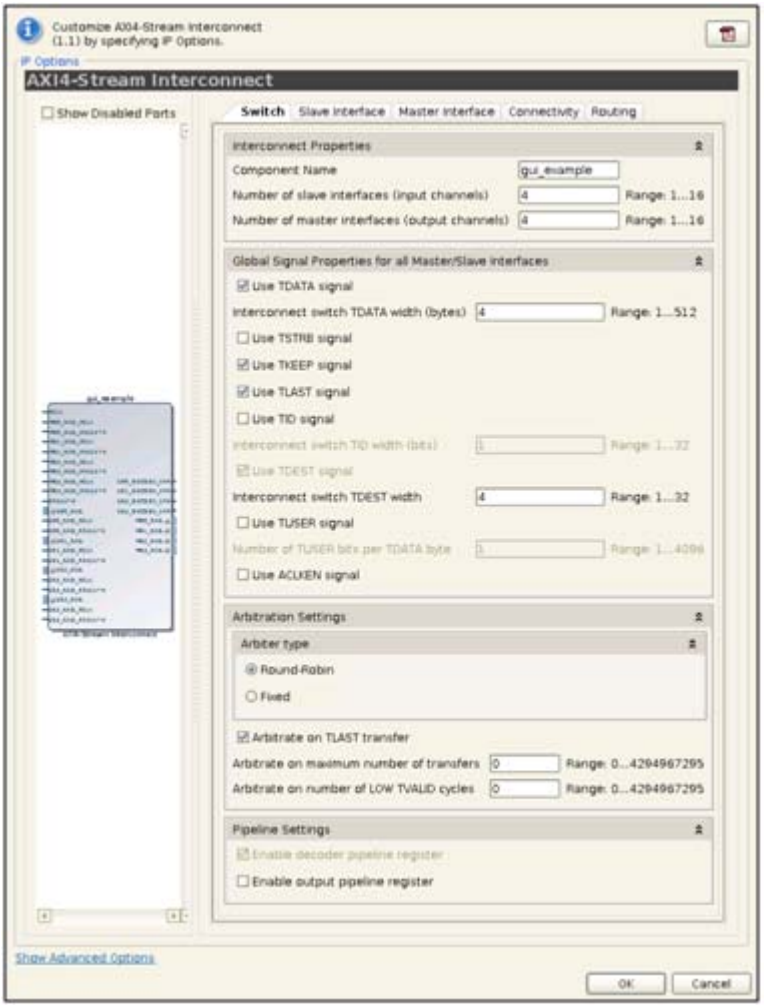

Figure 4-1: **Customization Dialog Box - Switch Settings Tab**

Review each of the available options across the five tabs, and modify them as desired so that the AXI4-Stream Interconnect solution meets the requirements of the larger project into which it is integrated. The following subsections discuss the options in detail.

Tab 1 – Switch Settings

Interconnect Properties

- **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 "_".

- **Number of Slave Interfaces (Input Channels)**

  Determines the number of AXI4-Stream slave interfaces in the design. Each slave interface can connect to one AXI4-Stream master. Up to 16 slave interfaces are supported. This option affects the number of slave interfaces that appear in subsequent pages. The first slave interface is designated as S00, the next slave interface is designated as S01, and this numbering continues to the sixteenth slave interface designated as S15.

- **Number of Master Interfaces (Output Channels)**

  Determines the number of AXI4-Stream master interfaces in the design. Each master interface can connect to one AXI4-Stream slave. Up to 16 master interfaces are supported. This option affects the number of master interfaces that appear in subsequent pages. The first master interface is designated as S00, the next master interface is designated as S01, and this numbering continues to the sixteenth master interface designated as S15.

Global Signal Properties for all Master/Slave Interfaces

- **Use TDATA Signal**

  If checked, this check box specifies if the optional `TDATA` signal is present on all the AXI4-Stream interfaces. If it is not present, the `TSTRB` and `TKEEP` signals cannot be enabled and the Interconnect switch TDATA Width parameter is fixed to 1.

- **Interconnect Switch TDATA Width (bytes)**

  Specifies the width in bytes of the interconnect switch and internal datapaths. The slave/master interfaces can be configured to have a different TDATA width than the internal width. This configuration is on the next page. A latency and area penalty is incurred if the slave/master interfaces TDATA width differ from the interconnect TDATA width. This parameter is an integer and can vary from 1 to 512 bytes or 8 to 4,096 bits.

- **Use TSTRB Signal**

  If checked, this check box specifies if the optional `TSTRB` signal is present on all the AXI4-Stream interfaces. This option can only be enabled if the **Use TDATA Signal** option is also enabled.

- **Use TKEEP Signal**

  If checked, this check box specifies if the optional TKEEP signal is present on all the AXI4-Stream interfaces. This option can only be enabled if the **Use TDATA Signal** option is also enabled.

RECOMMENDED: *Xilinx recommends that the* TKEEP *signal is utilized if there is any increase in TDATA width between the slave interface to interconnect switch or interconnect switch to the master interface and the* TID *or* TDEST *or* TLAST *signals are also enabled. This is recommended because the upsizer conversion can introduce null beats if the* TID, TDEST, *or* TLAST *signals transition such that they require the upsizer to flush a transfer early.*

- **Use TLAST Signal**

  If checked, this check box specifies if the optional TLAST signal is present on all the AXI4-Stream interfaces. This allows the option for packet mode FIFOs, and the **Arbitrate on TLAST transfer** arbitration setting.

- **Use TID Signal**

  If checked, this check box specifies if the optional TID signal is present on all the AXI4-Stream interfaces. This enables the **Interconnect switch TID Width (bits)** property.

- **Interconnect switch TID Width (bits)**

  This property specifies the width of the TID signal if enabled. Values between 1 and 32 are allowed. The TID width is consistent throughout the switch and all the interfaces.

- **Use TDEST Signal**

  If checked, this check box specifies if the optional TDEST signal is present on all the AXI4-Stream interfaces. This enables the **Interconnect switch TDEST Width (bits)** property. This option is mandatory if the **Number of Master Interfaces** is greater than 1. The TDEST signal is used by the interconnect switch to route slave interfaces transfers to a master interface.

- **Interconnect switch TDEST Width (bits)**

  This property specifies the width of the TDEST signal if enabled. If **Number of Master Interfaces** is 1, the minimum allowed value is 1. Otherwise, the minimum is calculated by the formula:

  ```
  ceil(log2(Number of Master Interfaces))
  ```

  The maximum value allowed is 32. The TDEST width is consistent throughout the switch and all the interfaces.

- **Use TUSER Signal**

  If checked, this check box specifies if the optional TUSER signal is present on all the AXI4-Stream interfaces. This enables the **Number of TUSER bits per TDATA byte** property. The calculated width of the TUSER signal is displayed to the right of the check box. This width is calculated with the formula:

  Interconnect switch TDATA Width x Number of TUSER bits per TDATA byte.

- **Number of TUSER bits per TDATA byte**

  Specifies the number of TUSER bits per TDATA byte. There must be at least one TUSER bit per TDATA byte if the TUSER signal is enabled. The width of TUSER is not specified explicitly to enforce this relationship between TDATA and TUSER. This relationship is enforced to ensure that data width conversion, if used, is defined.

- **Use ACLKEN**

  If checked, this check box specifies if the optional ACLKEN signal is present on all the AXI4-Stream interfaces. Enabling the ACLKEN signal incurs area resource and latency penalties when using clock converters and FIFOs.

Arbitration Settings

The arbitration settings are available to modify when there is more than one slave interface specified. For each master interface, there is an arbiter module that handles multiple incoming requests. These arbitration settings configure each of the arbiters with the same settings. The arbiter protocol consists of three stages: request, grant, and done. The request signals are triggered by decoding the TDEST input on each slave interface. When multiple slave interfaces have asserted request signals to the same master interface, the **Arbiter Type** will affect which slave interface will be granted next. The **Arbitrate on TLAST transfer**, **Arbitrate on maximum number of transfers**, and **Arbitrate on number of LOW TVALID cycles** affects when the done signal will be asserted. If any of the three conditions for arbiter completion is met, the request/grant/done cycle is completed and a new cycle can begin.

- **Arbiter Type**

  There are two arbitration algorithms available to choose from. The Round-Robin arbitration algorithm arbitrates in a round-robin fashion between all the slave interfaces. The Fixed Priority arbitration algorithm selects the first slave interface S00 as the highest priority, followed by the second interface S01 as second highest priority, and so on.

- **Arbitrate on TLAST transfer**

  If checked, this setting indicates that a transaction is complete when a transfer with TLAST asserted passes from the slave interface to the master interface of the interconnect switch. The arbiter is then able to arbitrate to the next arbitration winner.

- **Arbitrate on maximum number of transfers**

  This setting specifies how many transfers to count before relinquishing the granted arbitration. If set to zero, then the number is infinite and **Arbitrate on TLAST transfer** must be set. If set to one, then after each transfer, the interconnect switch relinquishes control and requests another arbitration. If set to a value greater than one, then after the specified number of transfers, the arbitration is relinquished.
  For example, setting this value to 16 allows 16 transfers to pass from slave interface to master interface per each arbitration grant.

- **Arbitrate on number of LOW TVALID cycles**

  This setting allows relinquishing of a granted arbitration without a transfer (at least one transfer will always complete.) A watchdog timer is instantiated and counts the number of consecutive LOW TVALID signals between the granted SI and MI interfaces. When the requisite number of LOW TVALID cycles is counted as specified here, the switch relinquishes the granted arbitration and signals that the transaction is complete to the arbiter. If TVALID is asserted before the count is met, the count will reset back to zero. If this setting is set to 0, no watchdog timer is instantiated. If there is more than one slave interface, more than one master interface, and the **Arbitrate on maximum number of transfers** setting is greater than 1, this setting cannot be set to zero. This ensures that deadlock cannot occur.

Pipeline Settings

- **Decoder Pipeline**

  This option enables a pipeline stage at each of the slave interfaces of the interconnect switch. The decoded TDEST signal is registered before being sent to the arbiter. It adds one cycle of latency between when a transfer is presented at the switch and when the request is sent to the arbiter. This option is not enabled when there is only one slave interface, regardless of the status of this check box.

- **Output Pipeline**

  This option enables a pipeline stage at each of the master interfaces of the interconnect switch. It adds one cycle of latency and can help alleviate the timing paths associated with large multiplexers when larger designs are considered.

Tab 2 and 3 – Slave Interface/Master Interface Settings

The Slave Interface settings tab shown in Figure 4-2 allows configuration of the slave interfaces datapath settings. The master interface settings are identical to the slave interfaces settings, but act on the master interface datapath. The settings are arranged into tables with the settings grouped into columns, with each individual interface represented by each row.

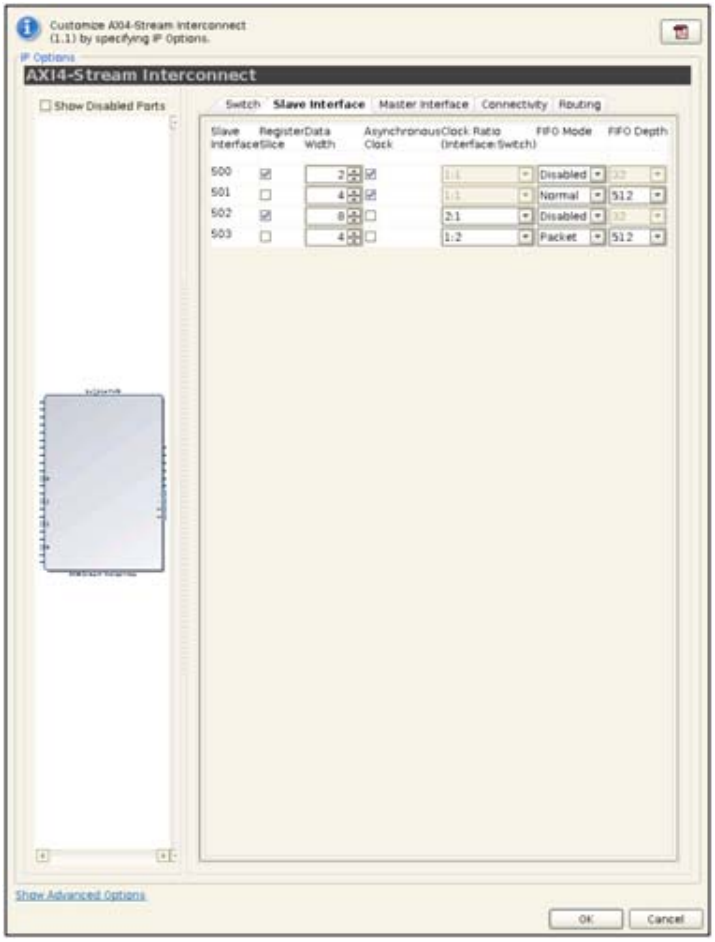

Figure 4-2: **Customization Dialog Box - Slave Interface Tab**

Register Slice

If checked, this check box enables a register slice pipeline stage at the interface boundary. This should be enabled if timing is not being met at the interface boundaries.

Data Width (Bytes)

The data width setting sets the width of the TDATA signal at the interface. If the TDATA width at the interface varies from the interconnect switch TDATA width, the data width converter is instantiated at the interface datapath. When the interconnect switch TDATA width is set on the first page, all the interfaces inherit the interconnect switch TDATA width as the default value. The data width can be set to any value from 1 to 4,096 bytes. Interface TDATA widths that vary from the interconnect switch TDATA width in non-integer multiples can incur high resource utilization.

Asynchronous Clock

If checked, this check box enables an asynchronous clock converter between the interface clocks and the switch clocks. The asynchronous clock converter is FIFO based using the FIFO Generator IP core. This allows the ports Sxx_AXIS_ACLK and Mxx_AXIS_ACLK clock inputs to be completely independent of ACLK (interconnect switch clock.) If the **Asynchronous Clock** option is selected, the **Clock Ratio** option is disabled.

Clock Ratio (Interface: Switch)

This pull-down setting allows a fixed number of synchronous clock ratio settings to be configured. This option incurs less area utilization and latency utilization than the asynchronous clock converter, but requires stricter clocking. This option should be used if interface-to-interconnect switch clocks are phase aligned, but vary in frequency by a multiple. A 1:1 clock ratio is the default value which should be set when the clocks are from the same source and no clock converter is necessary. This option is disabled if the **Asynchronous Clock** setting is set.

Data FIFO Mode

Datapath FIFOs are available to buffer DATA before and after the interconnect switch if desired. There are two modes of operation for the FIFO: Normal and Packet. A normal FIFO operates similar to a register slice and passes through data as soon as possible. A packet-based FIFO (also known as store and forward) is available when the **Use TLAST** option is enabled. When in packet mode, transfers that enter the FIFO are stored up until a TLAST signal is received. After TLAST is received all transfers up until the transfer with **TLAST** will progress forward. The FIFO is implemented utilizing the FIFO Generator IP Core.

Data FIFO Depth

This pull-down selects the depth of the FIFO if enabled by the **Data FIFO Mode** option. Depths of 16 and 32 are implemented using LUTs. Depths of 64 and greater are implemented using Block RAMs. The width of the FIFO is determined by the Interconnect switch TDATA width as specified on the Switch tab.

Tab 4 – Slave to Master Interface Switch Connectivity Map

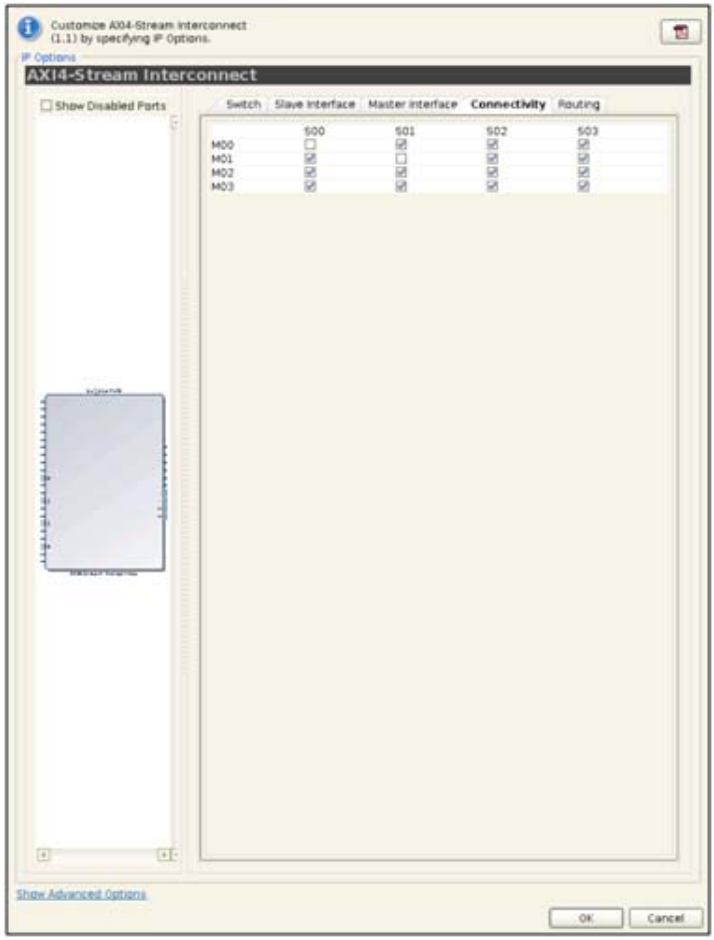

Figure 4-3: **Customization Dialog Box - Connectivity Map Tab**

The table shown in Figure 4-3 allows the configuration of the switch to optimize connectivity. By default full connectivity is enabled meaning that every slave interface is connected to every master interface. Deselecting a check box removes the connection from the slave interface listed in the column to the master interface in the row. Removing connections is desirable when it is known that a slave interface is never going to send transfers to a particular master interface. This removes unnecessary logic resulting in smaller area utilization, routing, and logic complexity.

Tab 5 – Routing Parameters

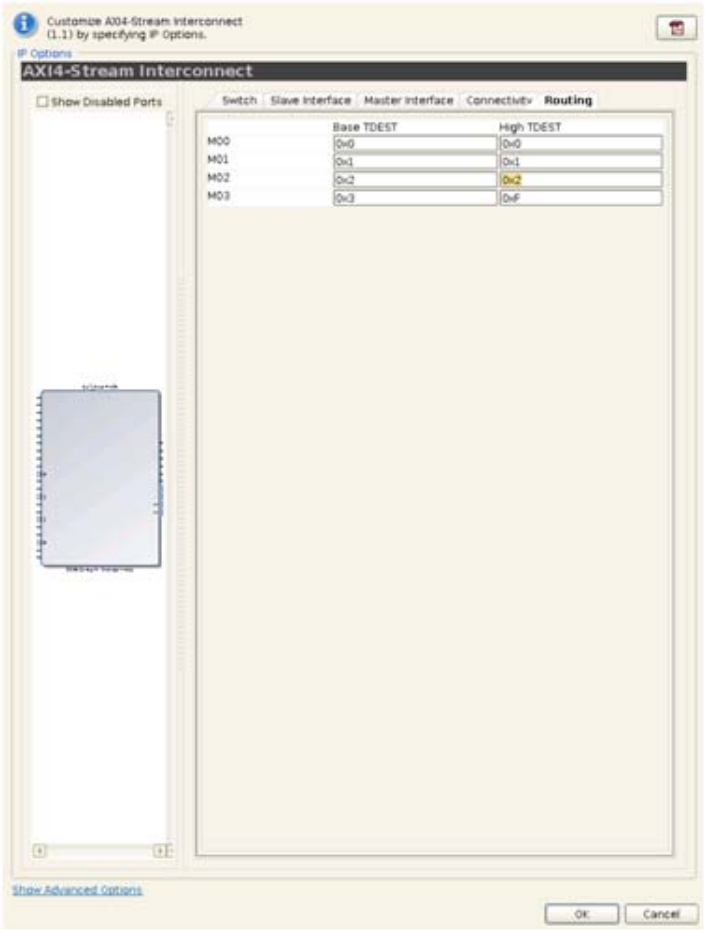

Figure 4-4: **Dialog Box - Routing Parameters Tab**

Routing Parameters

The routing parameters shown in Figure 4-4 set up the static decoding used by the switch to route slave interface transfers to master interfaces based on the TDEST signal. Each master interface must have a BASE/HIGH range that is within the valid width of the TDEST width specified previously. The BASE/HIGH pair must not overlap between master interfaces. When a transfer is received at the slave interface, the TDEST is decoded based on this table. A request is then sent to the arbiter for the master interface corresponding to the TDEST. If the slave interface is chosen by the arbiter, the transfer proceeds to the master interface.

Generating the Solution

Click **OK** to begin the generation process. The Vivado Design Suite generates and deliver a customized solution based on the provided option settings. When this process has completed, the customized IP is available under in the Project Manager under Sources with the name entered in the Component Name box during configuration.

User Parameters

Table 4-1 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter/Value	User Parameter/Value	Default Value
Number of slave interfaces (input channels)	c_num_si_slots	1
Number of master interfaces (output channels)	c_num_mi_slots	1
Use TDATA signal	has_tdata	TRUE
Interconnect switch TDATA width (bytes)	switch_tdata_num_bytes	1
Use TSTRB signal	has_tstrb	TRUE
Use TKEEP signal	has_tkeep	TRUE
Use TLAST signal	has_tlast	TRUE
Use TID signal	has_tid	TRUE
Interconnect switch TID width (bits)	c_switch_tid_width	1
Use TDEST signal	has_tdest	TRUE
Interconnect switch TDEST width	c_switch_tdest_width	1
Use TUSER signal	has_tuser	FALSE
Number of TUSER bits per TDATA byte	switch_tuser_bits_per_byte	1
Use ACLKEN signal	switch_use_aclken	FALSE
Arbiter type	arbiter_type	Round-Robin
Arbitrate on TLAST transfer	switch_packet_mode	FALSE
Arbitrate on maximum number of transfers	c_switch_max_xfers_per_arb	1
Arbitrate on number of LOW TVALID cycles	c_switch_num_cycles_timeout	0
Enable decoder pipeline register	c_switch_si_reg_config	1
Enable output pipeline register	c_switch_mi_reg_config	0
Synchronization Stages across Cross Clock Domain Logic	synchronization_stages	2
S<xx> Register Slice	c_s<xx>_axis_reg_config	0
S<xx> Data Width (Bytes)	s<xx>_axis_tdata_num_bytes	1
S<xx> Asynchronous Clock	c_s<xx>_axis_is_aclk_async	0
S<xx> Clock Ratio (Interface: Switch)	c_s<xx>_axis_aclk_ratio	12

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE Parameter/Value	User Parameter/Value	Default Value
4:1	48	
3:1	36	
2:1	24	
1:1	12	
1:2	6	
1:3	4	
1:4	3	
S<xx> FIFO MODE	s<xx>_axis_fifo_mode	0_(Disabled)
Disabled	0_(Disabled)	
Normal	1_(Normal)	
Packet	2_(Packet)	
S<xx> FIFO Depth	c_s<xx>_axis_fifo_depth	32
M<xx> Register Slice	c_m<xx>_axis_reg_config	0
M<xx> Data Width (Bytes)	m<xx>_axis_tdata_num_bytes	1
M<xx> Asynchronous Clock	c_m<xx>_axis_is_aclk_async	0
M<xx> Clock Ratio (Interface: Switch)	c_m<xx>_axis_aclk_ratio	12
4:1	48	
3:1	36	
2:1	24	
1:1	12	
1:2	6	
1:3	4	
1:4	3	
M<xx> FIFO MODE	m<xx>_axis_fifo_mode	0_(Disabled)
Disabled	0_(Disabled)	
Normal	1_(Normal)	
Packet	2_(Packet)	
M<xx> FIFO Depth	c_m<xx>_axis_fifo_depth	32
M<xx> S<yy> CONNECTIVITY	m<xx>_s<yy>_connectivity	TRUE
M<xx> Base TDEST	c_m<xx>_axis_basetdest	hex(<xx>)
M<xx> High TDEST	c_m<xx>_axis_hightdest	hex(<xx>)

Notes:

1. Where <xx> and <yy> are 0-padded integers between 0 and 15.

Output Generation

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

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite. The AXI4-Stream Interconnect does not generally require any additional timing constraints other than clock period constraints on its clocks inputs. When an asynchronous clock converter is utilized, the underlying FIFO Generator asynchronous FIFO is instantiated and the IP generates a core level constraint file to prevent timing paths crossing clock domains from causing false timing errors. Those constraints apply only to internal logic inside the AXI4-Stream Interconnect and are automatically generated with the IP.

IMPORTANT: *If an asynchronous clock converter is specified on an interface, do not use the same clock source for the clock between the interface and the switch.*

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 4].

Upgrading

For information about migrating a design from the ISE® Design Suite to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 7].

For customers upgrading the Vivado Design Suite or for upgrading to a more recent version of the IP core, important details (where applicable) about any port changes and other impact to user logic are included.

Debugging

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

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI4-Stream Interconnect core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the AXI4-Stream Interconnect core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

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

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

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

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

Master Answer Record for the AXI4-Stream Interconnect core

AR: 47407

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.

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

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 8].

AXI4-Stream Interface Debug

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

- If transmit `<interface_name>_tready` is stuck Low following the `<interface_name>_tvalid` input being asserted, the core cannot send data.
- If the receive `<interface_name>_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.

Appendix C

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. *FIFO Generator LogiCORE IP Product Guide* (PG057)
2. *AXI Reference Guide* (UG1037)
3. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
4. *Vivado Design Suite User Guide: Designing with IP* (U896)
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)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/04/2017	1.1	• Added Automotive Applications Disclaimer. • Added support for Spartan-7. • Updated the Direction for many of the signals in Table 2-5. • Modified the Arbitration Settings section in Chapter 4. • Updated the **Arbitrate on number of LOW TVALID cycles** setting in Chapter 4.
04/05/2017	1.1	• In the Resources row of the IP Facts table, replaced references to tables with a link to Performance and Resource Utilization on the web. • Replaced text in Maximum Frequencies and Resource Utilization sections with a link to Performance and Resource Utilization on the web.
11/18/2015	1.1	Added support for UltraScale+™ families and UltraScale™ architecture.
04/01/2015	1.1	• Updated resource utilization table. • Updated notes on arbitration configurations for achieving performance. • Added text to indicate the data width of the FIFO modules.
12/18/2012	1.1	• Updated for 14.4 and 2012.4. • Added information to Chapter 5, Constraining the Core • Updated Appendix B, Debugging

Date	Version	Revision
10/16/2012	1.1	Updated for 14.3 and 2012.3.
07/25/2012	1.1	• Added support for Vivado® Design Tools and Zynq®-7000 device. • Added performance characteristics and resource utilization numbers.
04/24/2012	1.0	Initial Release

Please Read: Important Legal Notices