

## Introduction

The LogiCORE™ IP AXI Interconnect core connects one or more AXI memory-mapped master devices to one or more memory-mapped slave devices. The AXI interfaces conform to the AMBA® AXI version 4 specification from ARM®, including the AXI4-Lite control register interface subset.

**Note:** The AXI Interconnect core is intended for memory-mapped transfers only; AXI4-Stream transfers are not applicable. IP with AXI4-Stream interfaces are generally connected to one another, and to DMA IP.

The AXI Interconnect core is provided as a non-encrypted, non-licensed (free) processor core (pcore) in the Xilinx® Platform Studio (XPS) software. The core is also provided in the ISE® Design Suite for use in non-embedded designs via the CORE Generator™ tool flow.

## Features

The XPS tool flow provides access to all features of the AXI Interconnect core. The CORE Generator tool flow provides access to a subset of these features, as described in this section.

## XPS Supported Features

These features of the AXI Interconnect core are supported by the XPS tool flow:

- AXI protocol compliant (AXI3, AXI4, and AXI4-Lite), and includes:
  - Burst lengths up to 256 for incremental (INCR) bursts
  - Converts AXI4 bursts > 16 beats when targeting AXI3 slave devices by splitting transactions
  - Generates REGION outputs for use by slave devices with multiple address decode ranges
  - Propagates USER signals on each channel, if any; independent USER signal width per channel (optional)
  - Propagates Quality of Service (QoS) signals, if any; not used by the AXI Interconnect core (optional)
- Interface data widths:
  - AXI4: 32, 64, 128, 256, 512, or 1024 bits
  - AXI4-Lite: 32 bits
- 32-bit address width

LogiCORE IP Facts Table					
Core Specifics					
Supported Device Family <sup>(1)</sup>	Zynq™-7000, Artix™-7, Virtex®-7, Kintex™-7, Virtex-6, Spartan®-6				
Supported User Interfaces	AXI4, AXI4-Lite, AXI3				
	Resources				Frequency
Configuration	LUTs	FFs	DSP Slices	Block RAMs	Max. Freq.
Config1	N/A	N/A	N/A	N/A	N/A
Provided with Core					
Documentation	Product Specification				
Design Files	Verilog, VHDL				
Example Design	Figure 1, page 7				
Test Bench	Not Provided				
Constraints File	User Constraints File (UCF)				
Simulation Model	Not Provided				
Tested Design Tools					
Design Entry Tools	ISE Design Suite 13.3, PlanAhead™ tool, XPS				
Simulation <sup>(2)</sup>	Mentor Graphics ModelSim, Cadence Incisive Enterprise Simulator (IES)				
Synthesis Tools	XST 13.3				
Support					
Provided by Xilinx @ <a href="http://www.xilinx.com/support">www.xilinx.com/support</a>					

1. For a complete list of supported derivative devices, see [IDS Embedded Edition Derivative Device Support](#).
2. For the supported versions of the tools, see the [ISE Design Suite 13: Release Notes Guide](#).

## XPS Supported Features (*Continued*)

- The Slave Interface (SI) of the core can be configured to comprise 1-16 SI slots to accept transactions from up to 16 connected master devices. The Master Interface (MI) can be configured to comprise 1-16 MI slots to issue transactions to up to 16 connected slave devices.
  - When connecting one master to one slave, the AXI Interconnect core can optionally perform address range checking. It can also perform any of the optional data-width conversions, clock-rate conversions, protocol conversions, register pipelining, and datapath buffering functions.
  - When connecting one master to one slave, and not performing any conversions or address range checking, the AXI Interconnect core is implemented as wires, with no resources, no delay, and no latency.
- Built-in data-width conversion:
  - Each master and slave connection can independently use data widths of 32, 64, 128, 256, 512, or 1024 bits wide:
    - The internal crossbar can be configured to have a native data width of 32, 64, 128, 256, 512, or 1024 bits.
    - Data-width conversion is performed for each master and slave connection that does not match the crossbar native data width.
  - When converting to a wider interface (upsizing), data is packed (merged) when permitted by address channel control signals (CACHE modifiable bit is asserted).
  - When converting to a narrower interface (downsizing), burst transactions are split into multiple transactions if the maximum burst length would otherwise be exceeded.
- 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).
  - The AXI Interconnect core exports reset signals resynchronized to the clock input associated with each SI and MI slot.
- Built-in AXI4-Lite protocol conversion:
  - The AXI Interconnect core can connect to any mixture of AXI4 and AXI4-Lite masters and slaves.
  - The AXI Interconnect core saves transaction IDs and restores them during response transfers, when connected to an AXI4-Lite slave.
    - AXI4-Lite slave devices do not need to sample or store IDs.
  - The AXI Interconnect core detects illegal AXI4-Lite transactions from connected AXI4 masters, such as any transaction that results in a burst of more than one word. The AXI Interconnect core generates a protocol-compliant error response to the connected master, and does not propagate the illegal transaction to the AXI4-Lite slave device.
    - The AXI4-Lite protocol conversion does not convert AXI4/AXI3 bursts into a sequence of single-beat transactions for AXI4-Lite slaves, so as to minimize resource utilization of the AXI4-Lite solution.
  - Write and Read transactions are single-threaded to AXI4-Lite slave devices, propagating only a single address at a time, which typically nullifies the resource overhead of separate AXI write and read address signals.
- Built-in AXI3 protocol conversion:
  - The AXI Interconnect core splits burst transactions of more than 16 beats from connected AXI4 masters into multiple transactions of no more than 16 beats when connected to an AXI3 slave device.
- Optional register-slice pipelining:
  - Available on each AXI channel connecting to each master and slave device.

- Facilitates timing closure by trading-off frequency vs. latency.
- One latency cycle per register-slice, with no loss in data throughput under all AXI handshake conditions.
- Optional datapath FIFO buffering:
  - Available on Write and Read datapaths connecting to each master and each slave.
  - 32-deep LUT-RAM based.
  - 512-deep block RAM based.
- Selectable Interconnect Architecture:
  - Crossbar mode (Performance optimized):
    - Shared-Address, Multiple-Data (SAMD) crossbar architecture.
    - Parallel crossbar pathways for Write data and Read data channels. When more than one Write or Read data source has data to send to different destinations, data transfers can occur independently and concurrently, provided AXI ordering rules are met.
    - Sparse crossbar datapaths according to configured connectivity map, resulting in reduced resource utilization.
    - One shared Write address arbiter, plus one shared Read address arbiter. Arbitration latencies typically do not impact data throughput when transactions average at least three data beats.
  - Shared Access mode (Area optimized):
    - Shared write data, shared read data, and single shared address pathways.
    - Issues one outstanding transaction at a time.
    - Minimizes resource utilization.
- Supports multiple outstanding transactions (crossbar mode):
  - Supports connected masters with multiple reordering depth (ID threads).
  - Supports up to 16 bit wide ID signals (system-wide).
  - Supports write response re-ordering, Read data re-ordering, and Read Data interleaving
  - Configurable Write and Read transaction acceptance limits for each connected master.
  - Configurable Write and Read transaction issuing limits for each connected slave.
  - Optional single-thread mode (per connected master) reduces thread control logic by allowing one or more outstanding transactions from only one thread ID at a time.
- “Single-Slave per ID” method of cyclic dependency (deadlock) avoidance:
  - For each ID thread issued by a connected master, the Interconnect allows one or more outstanding transactions to only one slave device for Writes and one slave device for Reads, at a time.
- Fixed priority and round-robin arbitration:
  - 16 configurable levels of static priority.
  - Round-robin arbitration is used among all connected masters configured with the lowest priority setting (priority 0), when no higher priority master is requesting.
  - Any SI slot that has reached its acceptance limit, or is targeting an MI slot that has reached its issuing limit, or is trying to access an MI slot in a manner that risks deadlock, is temporarily disqualified from arbitration, so that other SI slots can be granted arbitration.
- Supports TrustZone security for each connected slave as a whole:
  - If configured as a secure slave device, only secure AXI accesses are permitted.
  - Any non-secure accesses are blocked and the AXI Interconnect core returns a DECERR response to the connected master.
- Support for Read-only and Write-only masters and slaves, resulting in reduced resource utilization.

## CORE Generator Tool Supported Features

These features of the AXI Interconnect core are supported by the CORE Generator tool flow:

- AXI protocol compliant (AXI4 only), including:
  - Burst lengths up to 256 for incremental (INCR) bursts
  - Propagates Quality of Service (QoS) signals, if any; not used by the AXI Interconnect core (optional)
- Interface data widths:
  - 32, 64, 128, 256, 512, or 1024 bits
- Address width: 12 to 64 bits
- Up to 16 Slave interfaces (to accept transactions from up to 16 connected master devices) and one Master Interface (to issue transactions to one connected slave device).
  - When connecting one master to one slave, the AXI Interconnect core can optionally perform any of the optional data-width conversions, clock-rate conversions, register pipelining, and datapath buffering functions.
- Built-in data-width conversion:
  - Each master and slave connection can independently use data widths of 32, 64, 128, 256, 512, or 1024 bits:
    - The internal crossbar can be configured to have a native data width of 32, 64, 128, 256, 512, or 1024 bits.
    - Data-width conversion is performed for each master and slave connection that does not match the crossbar native data width.
  - When converting to a wider interface (upsizing), data is packed (merged) when permitted by address channel control signals (CACHE modifiable bit is asserted).
  - When converting to a narrower interface (downsizing), burst transactions are split into multiple transactions if the maximum burst length otherwise would be exceeded.
- 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).
  - The AXI Interconnect core exports reset signals resynchronized to the clock rate of each connected master and slave.
- Optional register-slice pipelining:
  - Available on all AXI channels connecting to each master and slave device.
  - Facilitates timing closure by trading-off frequency vs. latency.
  - One latency cycle per register-slice, with no loss in data throughput under all AXI handshake conditions.
- Optional datapath FIFO buffering:
  - Available on Write and Read datapaths connecting to each master and each slave.
  - 32-deep LUT-RAM based.
  - 512-deep block RAM based.
- Supports multiple outstanding transactions:
  - Supports connected masters with multiple reordering depth (ID threads).
  - Supports up to 8 bit wide ID signals from each connected master device (produces up to 12 bit wide ID output).
  - Supports write response re-ordering, Read data re-ordering, and Read Data interleaving

- Configurable Write and Read transaction acceptance limits for each connected master.
- Configurable Write and Read transaction issuing limits for connected slave.
- Fixed priority and round-robin arbitration:
  - 16 configurable levels of static priority.
  - Round-robin arbitration is used among all connected masters configured with the lowest priority setting (priority 0), when no higher priority master is requesting.
  - Any master device that has reached its acceptance limit is temporarily disqualified from arbitration, so that other connected masters can be granted arbitration.
- Support for Read-only and Write-only master devices, resulting in reduced resource utilization.

### Summary of CORE Generator Tool Flow Limitations

With respect to the features described in this document, the CORE Generator tool flow has these limitations:

- The process of defining an address map and all features relating to address decoding and selecting among multiple target slave devices is reserved for the XPS flow at this time. When used in the CORE Generator tool flow, the Interconnect supports connecting to only one slave device, and all transaction addresses are simply propagated. There are no conditions that result in the Interconnect producing a decode error (DECERR) response when used in the CORE Generator tool flow.
- Only AXI4 (memory-mapped) protocol is supported. When used in the CORE Generator tool flow, the Interconnect is intended only to connect to memory-type slave devices, such as an external memory controller, and not to control-register (AXI4-Lite) slaves.
- USER signals are not supported.
- Register slices are selectable per interface and apply to all AXI channels when enabled. The register slice type used per channel is fixed: Fully registered for W and R channels, Light weight for AW, AR, and B channels.
- Only full crossbar mode is supported, providing independent write and read operations for both address and data transfers. The “Shared Access” mode is not supported. Sparse crossbar connectivity is not applicable, as there is only one connected slave device.
- The width of ID signals is selected globally and applies to all Slave Interface (SI) ports. The Interconnect always adds four high-order bits when issuing the Master Interface ID signals, which indicate the originating SI index number.
- TrustZone security is not provided as a service of the Interconnect. (AW/ARPROT signals are propagated to the slave device.)

### AXI Interconnect Core Limitations

These limitations apply to the AXI Interconnect core as a whole (both the XPS and CORE Generator tool flows):

- The AXI Interconnect core does not support these AXI3 features:
  - Atomic locked transactions. This feature was retracted by AXI4 protocol. A locked transaction is changed to a non-locked transaction and propagated by the MI.
  - Write interleaving. This feature was retracted by AXI4 protocol. AXI3 master devices must be configured as if connected to a slave with Write interleaving depth of one.
- AXI4 QoS signals do not influence arbitration priority. QoS signals are propagated from SI to MI.
- The AXI Interconnect core does not convert multi-beat bursts into multiple single-beat transactions when targeting an AXI4-Lite slave device.
- The AXI Interconnect core does not support low-power mode or propagate the AXI C channel signals.
- The AXI Interconnect core does not time-out if the destination of any AXI channel transfer stalls indefinitely. All connected AXI slaves must respond to all received transactions, as required by AXI protocol.

- The AXI Interconnect core provides no address remapping.
- The AXI Interconnect core provides no built-in conversion to non-AXI protocols, such as APB.
- The AXI Interconnect core does not have clock-enable (ACLKEN) inputs. Consequently, the use of ACLKEN is not supported among memory-mapped AXI interfaces in Xilinx systems.

**Note:** The ACLKEN signal is supported for Xilinx AXI4-Stream interfaces.

## Definitions, Acronyms, and Abbreviations

Table 1 provides a list of acronyms, abbreviations, and specific definitions used in this document.

Table 1: Definitions, Acronyms, and Abbreviations

Item	Description
AXI	The generic term for all implemented AXI protocol interfaces.
master device or connected master	An IP or device (or one of multiple interfaces on an IP) that generates AXI transactions out from the IP onto the wires connecting to a slave IP.
slave device or connected slave	An IP or device (or one of multiple interfaces on an IP) that receives and responds to AXI transactions coming in to the IP from the wires connecting to a master IP.
master interface (generic)	An interface of an IP or module that generates out-bound AXI transactions and thus is the initiator (source) of an AXI transfer. On AXI master interfaces, AWVALID, ARVALID, and WVALID are outputs, and RVALID and BVALID are inputs.
slave interface (generic)	An interface of an IP or module that receives in-bound AXI transactions and becomes the target (destination) of an AXI transfer. On AXI slave interfaces, AWVALID, ARVALID, and WVALID are inputs, and RVALID and BVALID are outputs.
SI	AXI Interconnect Slave Interface: For XPS flow, Vectored AXI slave interface receiving in-bound AXI transactions from all connected master devices. For CORE Generator tool flow, one of multiple slave interfaces connecting to one master device.
MI	AXI Interconnect Master Interface: For XPS flow, Vectored AXI master interface generating out-bound AXI transactions to all connected slave devices. For CORE Generator tool flow, one master interface connecting to one slave device.
SI slot	Slave Interface Slot: A slice of the Slave Interface vector signals of the AXI Interconnect core that connect to a single master device.
MI slot	Master Interface Slot: A slice of the Master Interface vector signals of the AXI Interconnect core that connect to a single slave device.
SI-side	A module interface closer to the SI side of the AXI Interconnect core.
MI-side	A module interface closer to the MI side of the AXI Interconnect core.
Crossbar	Module at the center of the AXI Interconnect core that routes address, data and response channel transfers between various SI slots and MI slots.
SI hemisphere	Conversion and storage modules of the AXI Interconnect core located between the SI and crossbar.
MI hemisphere	Conversion and storage modules of the AXI Interconnect core located between the crossbar and MI.
upsizer	Data width conversion function in which the datapath width gets wider when moving in the direction from the SI-side toward the MI-side (regardless of write/read direction).
downsizer	Data width conversion function in which the datapath width gets narrower when moving in the direction from the SI-side toward the MI-side (regardless of write/read direction).

## Functional Description

Figure 1 shows the top-most AXI Interconnect core block diagram.

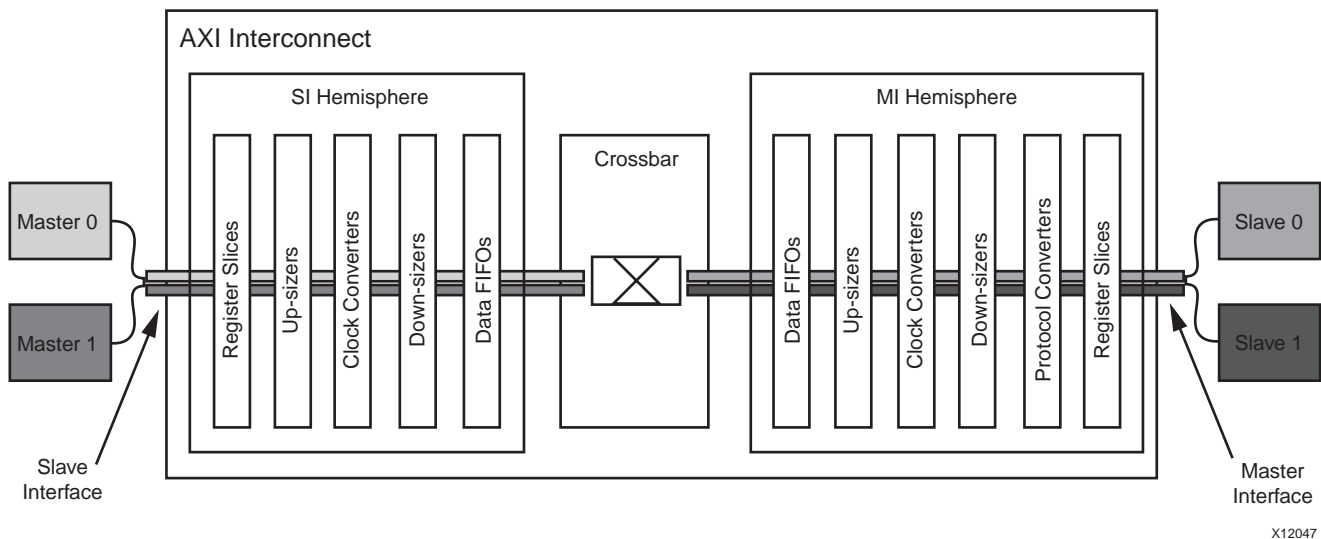


Figure 1: AXI Interconnect Core Diagram

The AXI Interconnect core consists of the SI, the MI, and the functional units that comprise the AXI channel pathways between them. The SI accepts Write and Read transaction requests from connected master devices. The MI issues transactions to slave devices. At the center is the *crossbar* that routes traffic on all the AXI channels between the various devices connected to the SI and MI. The AXI Interconnect core also comprises other functional units located between the crossbar and each of the interfaces that perform various conversion and storage functions. The crossbar effectively splits the AXI Interconnect core down the middle between the SI-related functional units (*SI hemisphere*) and the MI-related units (*MI hemisphere*).

The following subsection describes the use models for the AXI Interconnect core.

### Use Models

The AXI Interconnect core connects one or more AXI memory-mapped master devices to one or more memory-mapped slave devices. These use cases are described:

- [Pass Through](#)
- [Conversion Only](#)
- [N-to-1 Interconnect](#)
- [1-to-N Interconnect](#)
- [N-to-M Interconnect \(Crossbar Mode\)](#)
- [N-to-M Interconnect \(Shared Access Mode\)](#)



## Pass Through

When there is only one master device and only one slave device connected to the AXI Interconnect core, and the AXI Interconnect core is not performing any optional conversion functions or pipelining, all pathways between the slave and master interfaces degenerate into direct wire connections with no latency and consuming no logic resources. Figure 2 shows the Pass Through diagram. The AXI Interconnect core does, however, continue to resynchronize the INTERCONNECT\_ARESETN input to each of the slave and master interface clock domains for any master or slave devices that connect to the ARESET\_OUT\_N outputs, which consumes a small number of flip-flops.

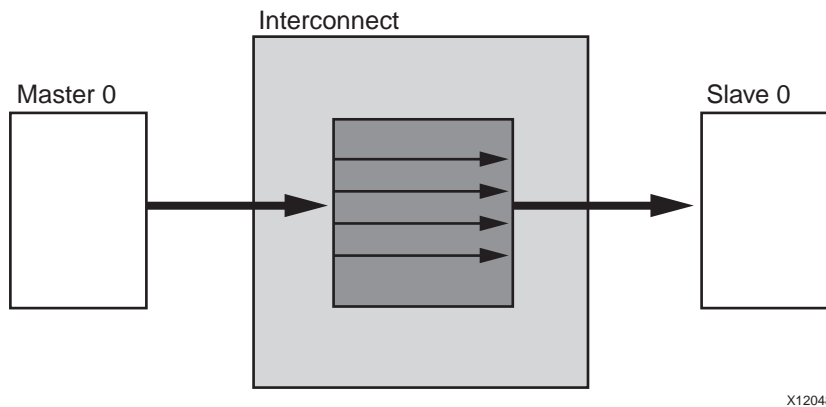


Figure 2: Pass-through AXI Interconnect Use Case

X12048

## Conversion Only

The AXI Interconnect core can perform various conversion and pipelining functions when connecting one master device to one slave device. These conversion and pipelining functions are:

- Data width conversion
- Clock rate conversion
- AXI4-Lite slave adaptation
- AXI-3 slave adaptation
- Pipelining, such as a register slice or data channel FIFO

In these cases, the AXI Interconnect core contains no arbitration, decoding, or routing logic (unless optional address range checking is enabled). There could be latency incurred, depending on the conversion being performed. Figure 3 shows an example of a one-to-one conversion use case.

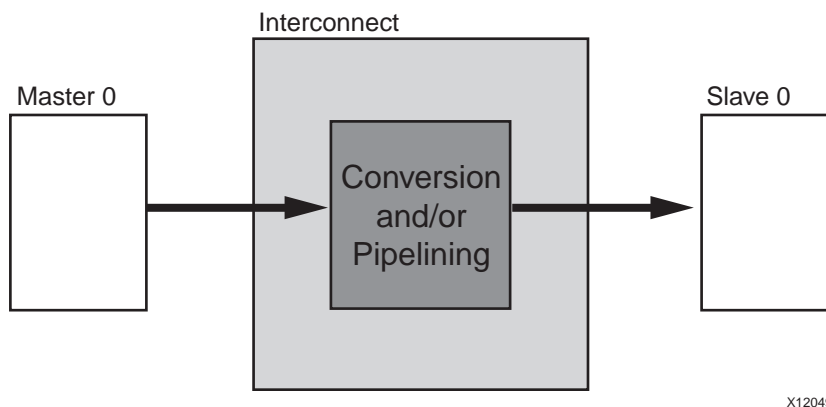


Figure 3: 1-to-1 Conversion AXI Interconnect Use Case

X12049



## N-to-1 Interconnect

A common degenerate configuration of the AXI Interconnect core occurs when multiple master devices arbitrate for access to a single slave device, typically a memory controller.

In these cases, address decoding logic might be unnecessary and omitted from the AXI Interconnect core (unless the optional address range validation is enabled). Any of the optional conversion functions, such as data width and clock rate conversion, can also be performed in this configuration as shown in [Figure 4](#).

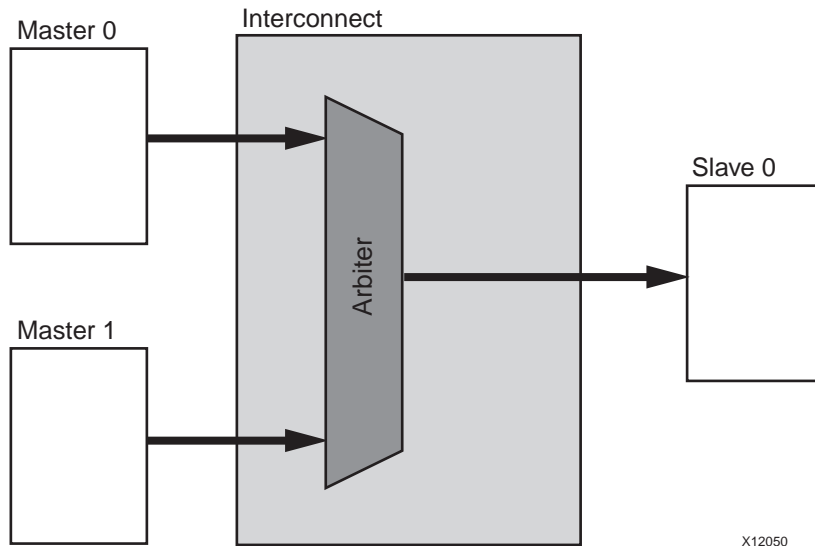


Figure 4: N-to-1 AXI Interconnect

## 1-to-N Interconnect

Another degenerative configuration of the AXI Interconnect core occurs when a single master device, typically, a processor, accesses multiple memory-mapped slave peripherals. In these cases, arbitration (in the address and Write data paths) is not performed, as shown in [Figure 5](#).

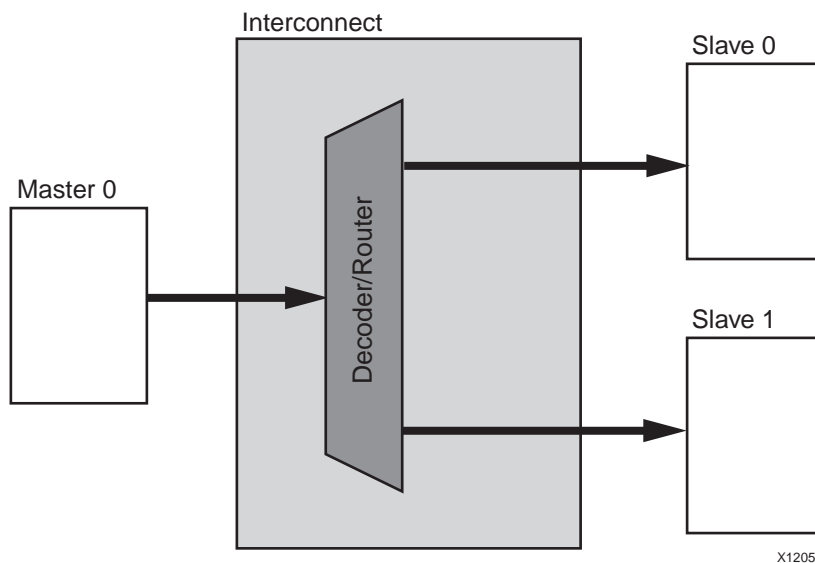


Figure 5: 1-to-N AXI Interconnect Use Case

## N-to-M Interconnect (Crossbar Mode)

The N-to-M use case of the AXI Interconnect core, when in Crossbar mode, features a Shared-Address Multiple-Data (SAMD) topology, consisting of sparse data crossbar connectivity, with single, shared Write and Read address arbitration, as shown in Figure 6 and Figure 7.

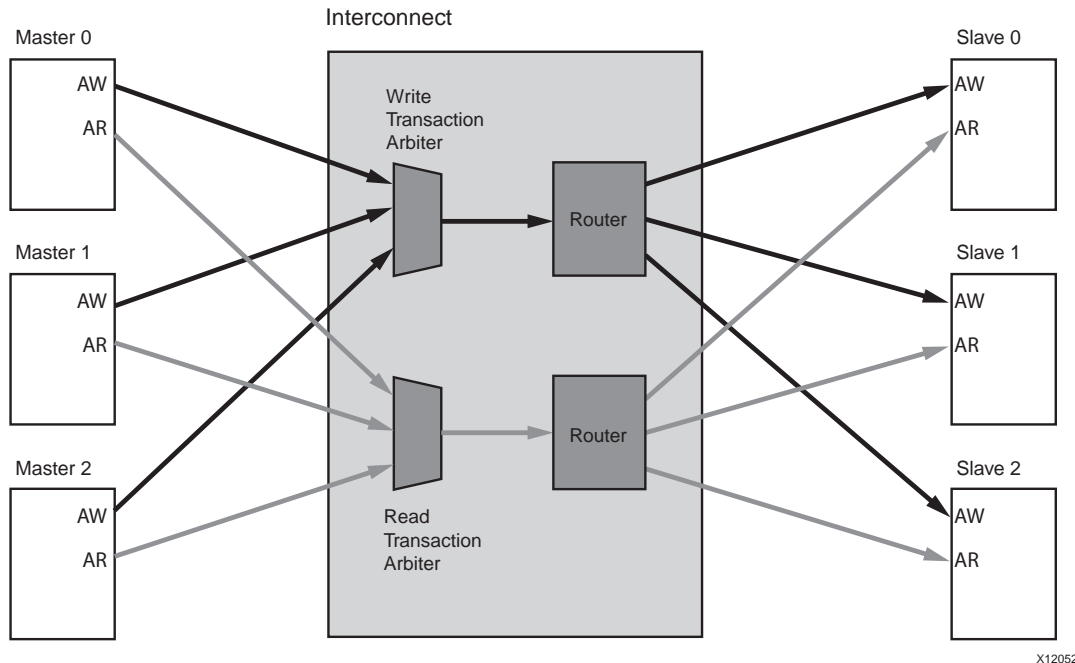


Figure 6: Shared Write and Read Address Arbitration

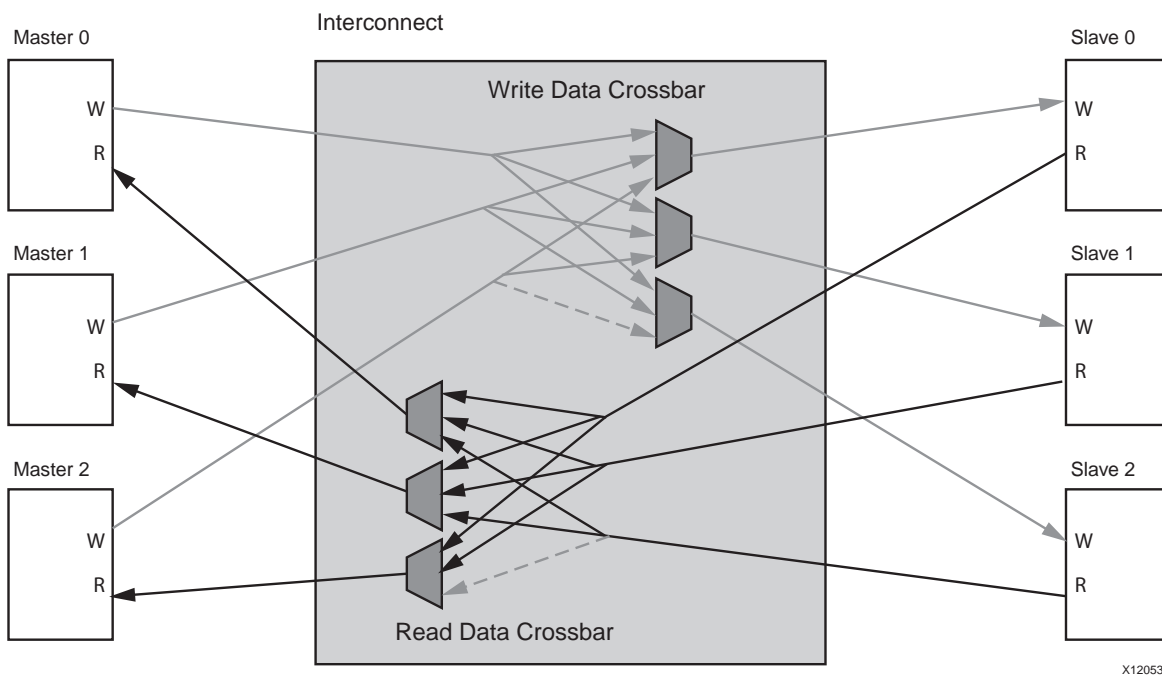


Figure 7: Sparse Crossbar Write and Read Data Pathways

Parallel Write and Read data pathways connect each SI slot to all the MI slots that it can access, according to the configured sparse connectivity map. When more than one source has data to send to different destinations, data transfers can occur independently and concurrently, provided AXI ordering rules are met.

The Write address channels among all SI slots feed into a central address arbiter, which grants access to one SI slot at a time. It is also the case for the Read address channels.

The winner of each arbitration cycle transfers its address information to the targeted MI slot, and pushes an entry into the appropriate command queue(s) that enable various data pathways to route data to the proper destination while enforcing AXI ordering rules.

### N-to-M Interconnect (Shared Access Mode)

When in Shared Access mode, the N-to-M use case of the AXI Interconnect core provides for only one outstanding transaction at a time, as shown in Figure 8. For each connected master, read transaction requests always take priority over writes. The arbiter then selects from among the requesting masters. A write or read data transfer is enabled to the targeted slave device. After the data transfer (including the write response) completes, the next request is arbitrated. Shared Access mode minimizes the resources used to implement the crossbar module of the Interconnect.

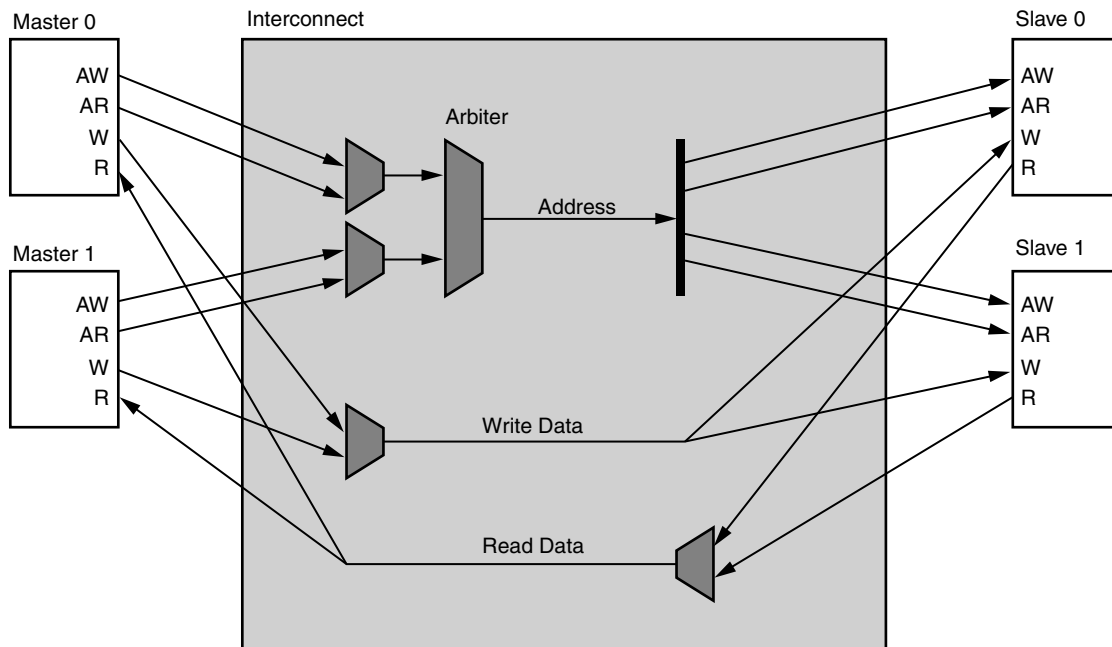


Figure 8: Shared Access Mode

## AXI Interconnect Core Functionality

These subsections describe the functionality within the AXI Interconnect core.

- [Top-Level Slave/Master Interface](#)
- [Width Conversion](#)
- [Width Conversion Transactions](#)
- [Clock Conversion](#)
- [Peripheral Register Slices](#)
- [Datapath FIFOs](#)

- [Use of ID Signals](#)
- [Multiple Address Range Support](#)
- [Cyclic Dependency Avoidance](#)
- [Error Signaling](#)

## Top-Level Slave/Master Interface

When deployed using the XPS flow, the top-level interface consists of a single, vectored AXI SI, plus a single, vectored AXI MI.

Each vectored interface can be configured to connect to between 1 and 16 master/slave devices.

For each signal comprising a vectored AXI interface on the core, its natural width is multiplied by the number of devices to which it is connected. All of the bit slices that connect to a single device are referred to as a *slot* of the interface. For example, the `AWLEN` signal carries an 8-bit value indicating the number of data beats in a Write transaction. If the AXI Interconnect core is configured with two SI slots, then the `S_AXI_AWLEN` signal is a total of 16 bits wide.

The effective widths of the `WDATA`, `WSTRB`, and `RDATA` signals are also configurable per MI/SI. The width of each of these signals on the vectored SI or MI is the maximum configured data width among all the SI and MI slots, as well as the Interconnect's native data width, multiplied by the number of slots. The unused high-order bits for each narrower slot are tied off (for inputs) or left unconnected (for outputs) within the AXI Interconnect core, and are trimmed by the implementation tools. Each AXI interface signal therefore allocates the same physical width across all slots.

For example, if the AXI Interconnect core is configured with two SI slots, one with data width 32 bits and one with data width 128 bits, and there are no larger data widths configured on any MI slot or the Interconnect itself, then each of the `WDATA` and `RDATA` signals on the SI of the core is a total of 256 bits wide, as shown in [Figure 9](#).

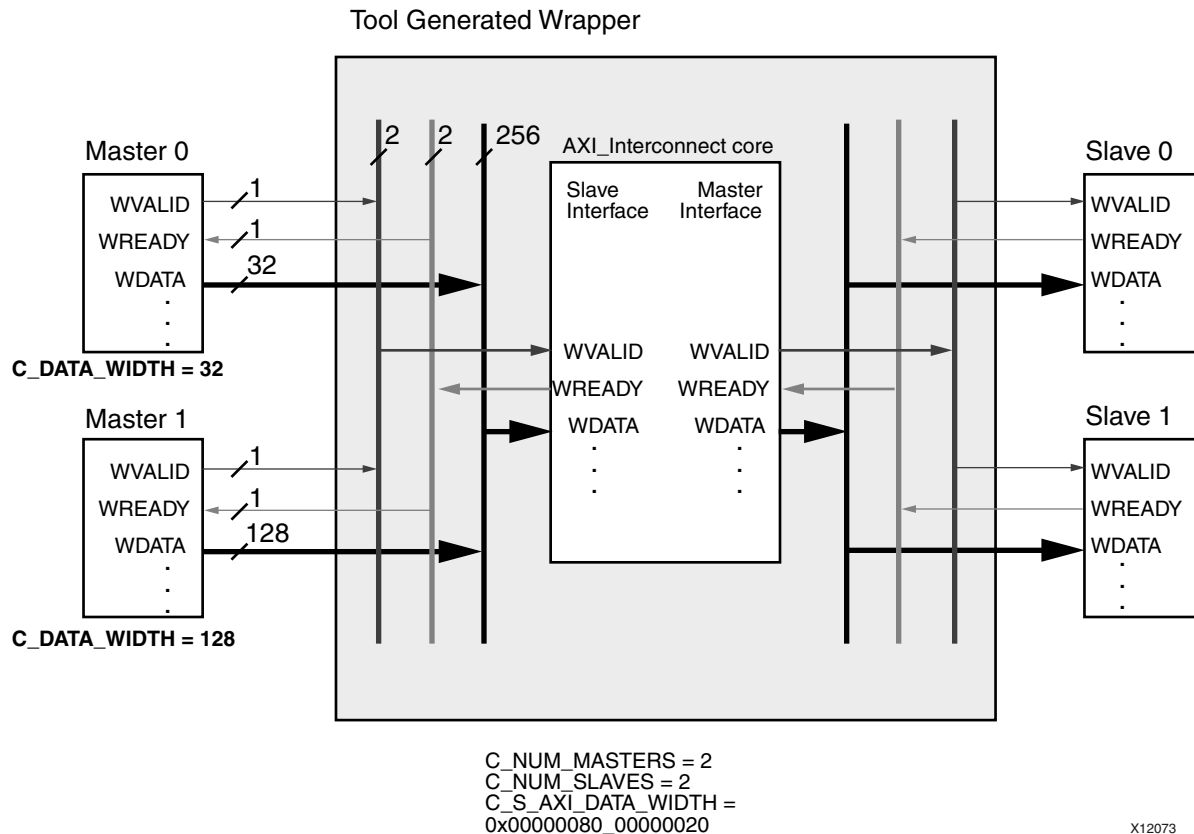


Figure 9: Vectored Slave/Master Interface

Specifically:

- “slot 0” uses WDATA[31:0]
- “slot 1” uses WDATA[255:128], and connects to WDATA[127:0] of the “Master 1” device).
- WDATA[127:32] remains tied off or unconnected inside the AXI Interconnect core.

Similar to I/O signals, many configuration parameters on the AXI Interconnect core are also formatted as vectors across all the SI or MI slots. Vectored parameters are formatted as follows:

- Parameters that define Boolean conditions, such as the TrustZone security indicator (C\_M\_AXI\_SECURE), are formatted as bit vectors with one bit per slot.
- Parameters that define any numerical value, regardless of value range, are formatted as bit vectors with 32 bits per slot.
- Base and high addresses are an exception and are formatted as 64 bits per slot.

In the [Figure 9](#) example, the value of the vectored parameter that defines the effective data widths of the slots on the SI (C\_S\_AXI\_DATA\_WIDTH) would be 0x0000008000000020, where 0x20 represents 32 bits for slot 0, and 0x80 represents 128 bits for slot 1. Parameter values are little-endian, as are I/O signals; consequently, the value corresponding to slot 0 appears at the right-hand Least Significant Bit (LSB) end of the parameter vector.

When deployed using the CORE Generator tool flow, an additional module layer is inserted above the vectored interface that splits up the vectored Slave Interface into individual enumerated interfaces suitable for direct connection to AXI master devices in an HDL design. This top-level module also splits out the individual SI-related parameters accordingly.

## Width Conversion

The AXI Interconnect core has a parametrically defined, internal, native data width, which supports 32, 64, 128, 256, 512, or 1024 bits. The AXI data channels that span the crossbar are sized to the *native* width of the AXI Interconnect core, as specified by the `C_INTERCONNECT_DATA_WIDTH` parameter.

When any SI slots or MI slots are sized differently, the AXI Interconnect core inserts width conversion units to adapt the slot width to the AXI Interconnect native width before transiting the crossbar to the other hemisphere.

The width conversion functions differ depending on whether the datapath width gets wider (upsizing) or narrower (downsizing) when moving in the direction from the SI toward the MI. The width conversion functions are the same in either the SI hemisphere (translating from the SI to the AXI Interconnect native width) or the MI hemisphere (translating from the AXI Interconnect native width to the MI).

MI and SI slots have an associated individual parametric data width value. The AXI Interconnect core adapts each MI and SI slot automatically to the internal native data width as follows:

- When the data width of an SI slot is wider than the internal native data width of the AXI Interconnect core, a downsizing conversion is performed along the pathways of the SI slot.
- When the internal native data width of the AXI Interconnect core is wider than that of an MI slot, a downsizing conversion is performed along the pathways of the MI slot.
- When the data width of an SI slot is narrower than the internal native data width of the AXI Interconnect core, an upsizing conversion is performed along the pathways of the SI slot.
- When the internal native data width of the AXI Interconnect core is narrower than that of an MI slot, an upsizing conversion is performed along the pathways of the MI slot.

The following subsections describe downsizing and upsizing.

### Downsizing

When the data width on the SI side is wider than that on the MI side, and the transfer size of the transaction is also wider than the data width on the MI side, then downsizing is performed and, in the transaction issued to the MI side, the number of data beats is multiplied accordingly.

- For writes, data serialization occurs
- For reads, data merging occurs

The AXI Interconnect core sets the `RRESP` for each output data beat (on the SI) to the worst-case error condition encountered among the input data beats being merged, according to the following descending precedence order: `DECERR`, `SLVERR`, `OKAY`, `EXOKAY`.

When the transfer size of the transaction is equal to or less than the MI side data width, the transaction (address channel values) remains unchanged. Data transfers pass through unchanged except for byte-lane steering. This applies to both writes and reads.

When downsizing, the AXI Interconnect core factors up the length of each burst and detects when the resulting burst length would exceed the maximum burst limit (256 data beats for AXI4). In such cases, the AXI Interconnect core splits the transaction automatically into multiple conforming burst transactions.

- If the `AWLOCK` or `ARLOCK` signal indicates an Exclusive Access write or read transaction, and downsizing results in splitting, then the AXI Interconnect core changes the `LOCK` signal in all output transactions to indicate Normal Access (0).
- When a downsized Write transaction results in splitting, the AXI Interconnect core coalesces the multiple Write responses at the MI and issues one Write response on the SI. The core sets the error response code (`BRESP`) to the worst-case error condition encountered among the multiple input responses, according to the following descending precedence order: `DECERR`, `SLVERR`, `OKAY` (`EXOKAY` cannot occur in a split transaction).

Downsizing, including transaction splitting, is not restricted by values of the AW/ARCACHE signal (specifically the “modifiable” bit). Transaction splitting due to downsizing cannot be restricted by CACHE because there is no other alternative for completing the transaction. See [Table 2, page 17](#) for the various size conversions.

The downsizer module allows multiple outstanding transactions to be propagated. Transaction characteristics from the AW/AR channel transfers are queued while awaiting corresponding response transfers. However, due to the possibility of write response and read data re-ordering, transaction acceptance by the AW and AR channel downsizers is restricted to a single ID thread at a time.

The interconnect does not support downsizing directly from 1024 bits to 32 bits, either between the SI and crossbar or between the crossbar and MI. If any SI is 1024 bits, then C\_INTERCONNECT\_DATA\_WIDTH must be set to a value greater than 32. If the MI is 32 bits, then C\_INTERCONNECT\_DATA\_WIDTH must be set to a value less than 1024.

### Upsizing

When the data width on the MI side is wider than that on the SI side, then upsizing is performed. Data packing is performed (for INCR and WRAP bursts), provided the AW/ARCACHE[1] bit (“Modifiable”) is asserted.

In the resulting transaction issued to the MI side, the number of data beats is reduced accordingly.

- For Writes, data merging occurs.
- For Reads, data serialization occurs.
  - The AXI Interconnect core replicates the RRESP from each input data beat onto the RRESP of each output data beat (on the SI).

When the AW/ARCACHE[1] bit is deasserted, the transaction (address channel values) remains unchanged and data transfers pass through unchanged except for byte-lane steering.

This latter functionality is commonly referred to as an “expander.”

Upsizing never results in transaction splitting. See [Table 2](#) for the various size conversions.

The upsizer module allows multiple outstanding transactions to be propagated. Transaction characteristics from the AW/AR channel transfers are queued while awaiting corresponding response transfers. However, due to the possibility of read data re-ordering, transaction acceptance by the AR-channel upsizer is restricted to a single ID thread at a time. Write transactions, however, are not restricted by ID thread, as B-channel responses require no transformation by the upsizer, and can therefore be propagated in any order as received.

### WIDTH Conversion Transaction Transformations

[Table 2](#) uses these following designators when describing properties, signals, and derived equations:

- si = Slave Interface
- cb = Interconnect (crossbar) core
- mi = Master Interface

[Table 2](#) lists:

- SI hemisphere transformations when relative DWidth compares si.DW to cb.DW
- MI hemisphere transformations when relative DWidth compares cb.DW to mi.DW

### Supporting Equations

These width conversion equations are enumerated in [Table 2](#).

1. When width conversion results in a change in transaction length, the output SIZE is always the same as the output DATA\_WIDTH.



2.  $si.DW = C\_S\_AXI\_DATA\_WIDTH$
3.  $cb.DW = C\_INTERCONNECT\_DATA\_WIDTH$
4.  $mi.DW = C\_M\_AXI\_DATA\_WIDTH$
5.  $si.Bytes = si.DW^{[2]} / 8$
6.  $cb.Bytes = cb.DW^{[3]} / 8$
7.  $mi.Bytes = mi.DW^{[4]} / 8$
8.  $cb.ByteMask = cb.Bytes^{[5]} - 1$
9.  $mi.ByteMask = mi.Bytes^{[6]} - 1$
10.  $si.SIZE = S\_AXI\_AWSIZE$  or  $S\_AXI\_ARSIZE$ , as applicable
11.  $cb.SIZE = si.SIZE$  if  $(cb.LEN=si.LEN)$ , else  $\log_2(cb.Bytes^{[6]})$
12.  $mi.SIZE = cb.SIZE$  if  $(mi.LEN=cb.LEN)$ , else  $\log_2(mi.Bytes^{[7]})$
13.  $si.SizeMask = (2^{**}si.SIZE^{[10]}) - 1$
14.  $cb.SizeMask = (2^{**}cb.SIZE^{[11]}) - 1$
15.  $mi.SizeMask = (2^{**}mi.SIZE^{[12]}) - 1$
16.  $cb.AlignedStart = si.ADDR \& \sim cb.ByteMask^{[8]}$
17.  $cb.AlignedEnd = ((si.ADDR \& \sim si.SizeMask^{[13]}) + (si.LEN * 2^{**}si.SIZE^{[10]})) \& \sim cb.ByteMask^{[9]}$
18.  $cb.upsize\_LEN = (cb.AlignedEnd^{[17]} - cb.AlignedStart^{[16]}) / cb.Bytes^{[6]}$
19.  $mi.AlignedStart = cb.ADDR \& \sim mi.ByteMask^{[9]}$
20.  $mi.AlignedEnd = ((cb.ADDR \& \sim cb.SizeMask^{[13]}) + (cb.LEN * 2^{**}cb.SIZE^{[11]})) \& \sim mi.ByteMask^{[9]}$
21.  $mi.upsize\_LEN = (mi.AlignedEnd^{[20]} - mi.AlignedStart^{[19]}) / mi.Bytes^{[4]}$
22.  $si.conv\_ratio = (2^{**}si.SIZE^{[10]}) / cb.Bytes^{[8]}$
23.  $cb.conv\_ratio = (2^{**}cb.SIZE^{[10]}) / mi.Bytes^{[9]}$
24.  $si.downsize\_LEN = (si.LEN+1) * si.conv\_ratio - 1^{[22]}$
25.  $cb.downsize\_LEN = (cb.LEN+1) * cb.conv\_ratio - 1^{[23]}$
26.  $cb.AlignedAdjustment = (si.ADDR \& si.SizeMask^{[13]} \& \sim cb.ByteMask^{[8]}) / cb.Bytes^{[6]}$
27.  $mi.AlignedAdjustment = (cb.ADDR \& cb.SizeMask^{[14]} \& \sim mi.ByteMask) / mi.Bytes^{[9]}$
28.  $si.burst\_bytes = 2^{**}si.SIZE^{[10]} * (si.LEN+1)$
29.  $cb.burst\_bytes = 2^{**}cb.SIZE^{[11]} * (cb.LEN+1)$
30.  $si.burst\_mask = si.burst\_bytes^{[28]} - 1$
31.  $cb.burst\_mask = cb.burst\_bytes^{[29]} - 1$
32.  $si.wrap\_address = si.ADDR \& \sim si.burst\_mask^{[30]}$
33.  $cb.wrap\_address = cb.ADDR \& \sim cb.burst\_mask^{[31]}$
34.  $si.wrap1\_LEN = (si.burst\_bytes^{[28]} - (si.ADDR \& si.burst\_mask^{[30]})) / cb.Bytes - 1^{[8]}$
35.  $cb.wrap1\_LEN = (cb.burst\_bytes^{[29]} - (cb.ADDR \& cb.burst\_mask^{[31]})) / mi.Bytes - 1^{[7]}$
36.  $si.wrap2\_LEN = (si.ADDR \& si.burst\_mask^{[30]}) / cb.Bytes - 1^{[6]}$
37.  $cb.wrap2\_LEN = (cb.ADDR \& cb.burst\_mask^{[31]}) / mi.Bytes - 1^{[7]}$

**Note:** "x%y" denotes x modulo y.

## Width Conversion Transactions

Table 2: Width Conversion Transactions

Relative DWidth	Conditions	Output Trans	Output LEN	Output ADDR	Output BURST
<b>INCR Bursts</b>					
si.DW[2] = cb.DW[3]	always	1	No change	No change	INCR
cb.DW[3] = mi.DW[4]	always	1	No change	No change	INCR
si.DW[2] > cb.DW[3]	if (2**si.SIZE [10] <= cb.Bytes [6])	1	No change	No change	INCR
	else if (si.downsize_LEN [24] <= 255)	1	si.downsize_LEN [24] - cb.AlignedAdjustment [26]	No change	INCR (1)
	else	ceil ((si.downsize_LEN+1[24]) / 256)	first = 255 - cb.AlignedAdjustment [26] ; last = si.downsize_LEN [24] % 256; others = 255	first = si.ADDR; others = (cb.ADDR[i-1] & ~si.SizeMask [13]) + (256*cb.Bytes [6])	INCR (1)
cb.DW[3] > mi.DW[4]	if (2**cb.SIZE [11] <= mi.Bytes [7])	1	No change	No change	INCR
	else if (cb.downsize_LEN [25] <= 255)	1	cb.downsize_LEN [25] - mi.AlignedAdjustment [27]	No change	INCR (1)
	else	ceil ((cb.downsize_LEN+1[25]) / 256)	first = 255 - mi.AlignedAdjustment [27]; last = cb.downsize_LEN [25] % 256; others = 255	first = cb.ADDR; others = (mi.ADDR[i-1] & ~cb.SizeMask [14]) + (256*mi.Bytes [24])	INCR (1)
si.DW[2] < cb.DW[3]	if si.CACHE[1]	1	cb.upsize_LEN [18]	No change	INCR (1)
	else	1	No change	No change	INCR

1. When width conversion results in a change in transaction length, the output SIZE is always the same as the output DATA\_WIDTH.

Table 2: Width Conversion Transactions (Cont'd)

Relative DWidth	Conditions	Output Trans	Output LEN	Output ADDR	Output BURST
cb.DW[3] < mi.DW[4]	if (si.CACHE [1])	1	mi.upsize_LEN [21]	No change	INCR (1)
	else	1	No change	No change	INCR
<b>WRAP Bursts</b>					
si.DW[2] = cb.DW[3]	always	1	No change	No change	WRAP
cb.DW[3] = mi.DW[4]	always	1	No change	No change	WRAP
si.DW [2] > cb.DW [3]	if (2**si.SIZE [10] <= cb.Bytes)	1	No change	No change	WRAP
	else if (si.downsize_LEN [24] <= 15)	1	si.downsize_LEN [24]	No change	WRAP (1)
	else if ((si.ADDR & si.burst_mask [30]) == 0)	1	si.wrap1_LEN [34]	si.ADDR	INCR (1)
	else	2	first = si.wrap1_LEN [34]; second = si.wrap2_LEN [36]	first = si.ADDR; second = si.wrap_address [32]	INCR (1)
cb.DW[3] > mi.DW[4]	if (2**cb.SIZE [11] <= mi.Bytes)	1	No change	No change	WRAP
	else if (cb.downsize_LEN [25] <= 15)	1	cb.downsize_LEN [25]	No change	WRAP (1)
	else if ((cb.ADDR & cb.burst_mask [30]) == 0)	1	cb.wrap1_LEN [35]	cb.ADDR	INCR (1)
	else	2	first = cb.wrap1_LEN; [35] second = cb.wrap2_LEN [37]	first = cb.ADDR; second = cb.wrap_address [33]	INCR (1)
si.DW[2] < cb.DW[3], Write	if (si.CACHE[1])	1	cell((si.LEN+1) * (2**si.SIZE [10]) / cb.Bytes) - 1	si.wrap_address [32] + (cell((si.ADDR & si.burst_mask [30]) / cb.Bytes) * cb.Bytes) % si.burst_bytes [28]	If (cb.LEN>0) then WRAP; else INCR(1)
	else	1	No change	No change	WRAP
si.DW[2] < cb.DW[3], Read	if (si.CACHE[1])	1	cell((si.LEN+1) * (2**si.SIZE [10]) / cb.Bytes [6]) - 1	si.wrap_address [33] + (int((si.ADDR & si.burst_mask [30]) / cb.Bytes [6]) * cb.Bytes [6])	If (cb.LEN>0) then WRAP; else INCR(1)
	else	1	No change	No change	WRAP

1. When width conversion results in a change in transaction length, the output SIZE is always the same as the output DATA\_WIDTH.

Table 2: Width Conversion Transactions (Cont'd)

Relative DWidth	Conditions	Output Trans	Output LEN	Output ADDR	Output BURST
cb.DW[3] < mi.DW[4], Write	if (si.CACHE[1])	1	$\text{cell}((\text{cb.LEN}+1) * (2^{**}\text{cb.SIZE}^{[1:1]}) / \text{mi.Bytes}^{[7]}) - 1$	$\text{cb.wrap\_address}^{[33]} + (\text{cell}((\text{cb.ADDR} + \text{cb.burst\_mask}^{[31]}) / \text{mi.Bytes}^{[7]}) * \text{mi.Bytes}^{[7]}) \% \text{cb.burst\_bytes}^{[29]}$	If (mi.LEN > 0) then WRAP, else INCR <sup>(1)</sup>
	else	1	No change	No change	WRAP
cb.DW[3] < mi.DW[4], Read	if (si.CACHE[1])	1	$\text{cell}((\text{cb.LEN}+1) * (2^{**}\text{cb.SIZE}^{[1:1]}) / \text{mi.Bytes}^{[7]}) - 1$	$\text{cb.wrap\_address}^{[33]} + (\text{int}((\text{cb.ADDR} + \text{cb.burst\_mask}^{[31]}) / \text{mi.Bytes}^{[7]}) * \text{mi.Bytes}^{[7]})$	If (mi.LEN > 0) then WRAP, else INCR <sup>(1)</sup>
	else	1	No change	No change	WRAP
<b>FIXED Bursts</b>					
si.DW[2] = cb.DW[3]	always	1	No change	No change	FIXED
cb.DW[3] = mi.DW[4]	always	1	No change	No change	FIXED
si.DW[2] > cb.DW[3]	if (2**si.SIZE <sup>[1:1]</sup> <= cb.Bytes <sup>[6]</sup> )	1	No change	No change	FIXED
	else	si.LEN+1	$\text{all} = \max(\text{si.conv\_ratio}^{[22]} - \text{cb.AlignedAdjustment}^{[26]} - 1, 0)$	all = si.ADDR	INCR <sup>(1)</sup>
cb.DW[3] > mi.DW[4]	if (2**cb.SIZE <sup>[1:0]</sup> <= mi.Bytes <sup>[7]</sup> )	1	No change	No change	FIXED
	else	cb.LEN+1	$\text{all} = \max(\text{cb.conv\_ratio}^{[23]} - \text{mi.AlignedAdjustment}^{[27]} - 1, 0)$	all = cb.ADDR	INCR <sup>(1)</sup>
si.DW[2] < cb.DW[3]	always	1	No change	No change	FIXED
cb.DW[3] < mi.DW[4]	always	1	No change	No change	FIXED

1. When width conversion results in a change in transaction length, the output SIZE is always the same as the output DATA\_WIDTH.

## Clock Conversion

Clock conversion comprises the following:

- 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.
- An asynchronous clock conversion module performs either reduction or acceleration of clock-rates by passing the channel signals through an asynchronous FIFO.

For both the reduction and the acceleration modules, the sample cycle for the faster clock domain is determined automatically. Each module applies to all five AXI channels.

The MI and SI each have a vector of clock inputs in which each bit synchronizes all the signals of the corresponding interface slot. The AXI Interconnect core has its own native clock input. The AXI Interconnect core adapts the clock rate of each MI and SI slot automatically to the native clock rate of the core.

Typically, the native clock input of the AXI Interconnect core is tied to the same clock source as used by the highest frequency SI or MI slot in the system design, such as the MI slot connecting to the main memory controller.

## Peripheral Register Slices

You can optionally insert a two-deep register slice (skid buffer) on each of the five AXI channels at each SI or MI slot to help improve system timing closure. At the outer-most periphery of both the SI and MI, each channel of each interface slot can be optionally buffered by a register slice. These are provided mainly to improve system timing at the expense of one latency cycle.

Peripheral register slices are always synchronized to the SI or MI slot clock.

## Datapath FIFOs

Under some circumstances, AXI Interconnect throughput is improved by buffering data bursts. This is commonly the case when the data rate at an SI or MI slot differs from the native data rate of the AXI Interconnect core due to data width or clock rate conversion. To accommodate the various rate change combinations, you can optionally insert data burst buffers at these locations:

- The SI-side Write data FIFO is located before the crossbar module, after any SI-side width or clock conversion.
- The MI-side Write data FIFO is located after the crossbar module, before any MI-side width, clock, or protocol conversion.
- The MI-side Read data FIFO is located before (on the MI side of) the crossbar module, after any MI-side width, clock, or protocol conversion.
- The SI-side Read data FIFO is located after (on the SI side of) the crossbar module, before any SI-side width or clock conversion.

Data FIFOs are synchronized to the AXI Interconnect native clock. The width of each data FIFO matches the AXI Interconnect native data width.

## Use of ID Signals

The transaction ID signals that propagate from SI to MI (`AWID` and `ARID`) and back again (`BID` and `RID`) identify the original source of each transaction, and therefore, how responses received on the MI are to be routed back to the originating SI slot across the interconnect topology of the system.

Endpoint master devices can optionally output `AWID` and `ARID` signals that the master device can use to select among multiple “threads” of transactions, as though the master IP was comprised of multiple master devices internally. The “reordering depth” is the total number of ID values that can be generated by a master, and is assumed to be  $2^{*idwidth}$ , where `idwidth` is specified by the `THREAD_ID_WIDTH` parameter of each SI slot. Master devices with a reordering depth of one need not have any ID signals on their interface. Transaction ordering is as follows:

- Transactions belonging to the same thread must be returned in order.
- Transactions among different threads can be returned out-of-order.

ID values among all the SI slots must be made unique before propagating to any MI slot. The AXI Interconnect core prefixes a constant unique “master ID” value to the `AWID` and `ARID` signals sampled at each SI slot (if any).

A `BASE_ID` parameter associated with each SI slot allows the AXI Interconnect core to assign master IDs at compile time. Because endpoint master devices are not required to drive their assigned master ID on their own ID outputs, master devices do not need to be aware of their own assigned master ID values.

When two Interconnect instances are cascaded so that an MI slot of one instance connects to an SI slot of another instance, all ID signals produced by the upstream AXI Interconnect core are treated as though they are the thread ID bits of a connected master device. As with other master devices, the downstream AXI Interconnect core prefixes the ID signals sampled from a cascaded SI slot with a unique master ID. This causes the ID width to grow as it propagates forward across a cascaded AXI Interconnect topology. All responses matching that master ID are routed back to the upstream AXI Interconnect instance.

The `M_AXI_WID` signal is provided for any connected AXI3 slave devices. The `M_AXI_WID` signal is normally generated based on the `M_AXI_AWID` value issued for the corresponding AW-channel transfer. The `S_AXI_WID` signal is normally ignored except when the interconnect is configured in pass-through mode (1 SI slot, 1 MI slot) and both the connected master and slave devices are AXI3 (and no other conversions). In that case, the `S_AXI_WID` signal is directly propagated to `M_AXI_WID`, along with all other AXI interface signals.

Figure 10 shows an example of cascading two AXI Interconnect instances.

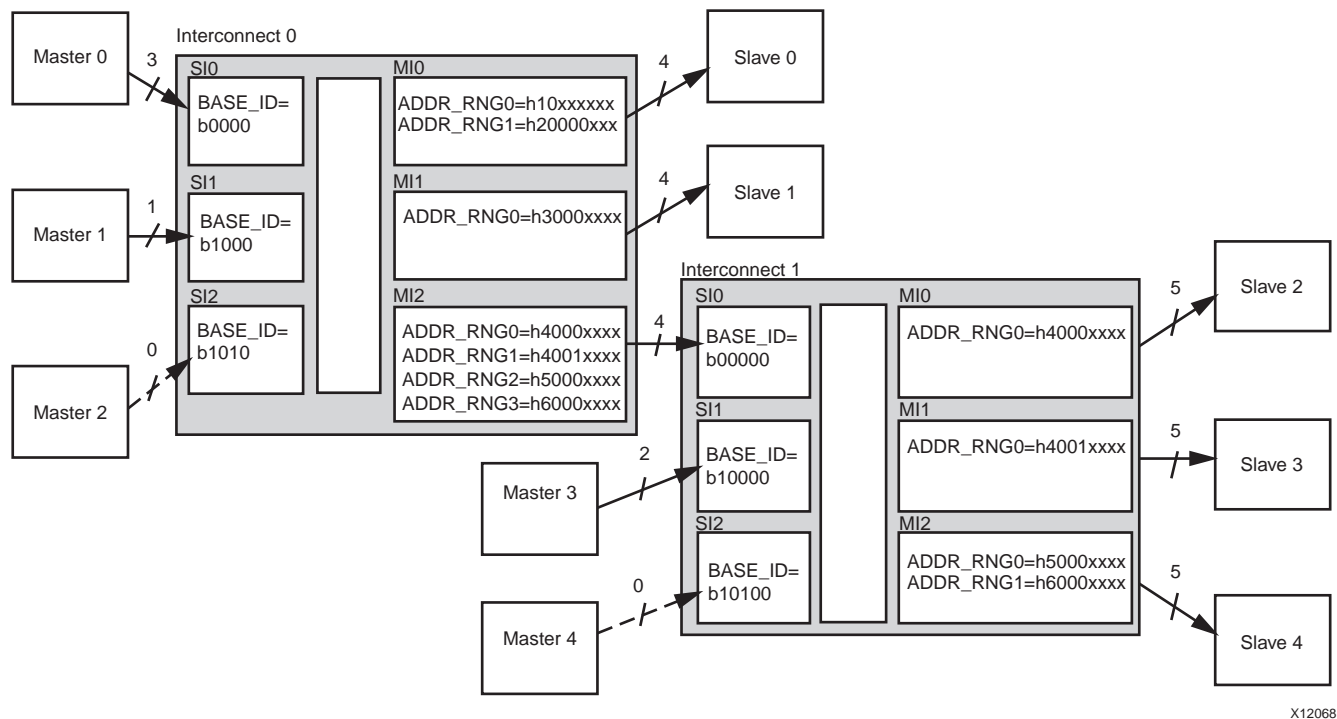


Figure 10: Cascading AXI Interconnect Cores

Figure 10 shows the following:

- MI slot #2 of AXI Interconnect 0 connects to SI slot 0 of AXI Interconnect 1. The endpoint slave devices #2-4 have address ranges as defined for MI slots #0-2 of AXI Interconnect 1.
- Note:** For conciseness, BASEADDR-HIGHADDR pairs are represented as ADDR ranges with don't cares.
- The complete set of all address ranges accessible by Interconnect 1 are listed as multiple address ranges for MI slot #2 of Interconnect 0.
- The arrows represent the ID signals propagated from each master device. AXI Interconnect 0 produces 4 bits of ID output, which is the smallest width possible to keep the master IDs unique. For example, when Master 0 issues a transaction, the output ID is its master ID (1'b0) followed by 3 bits of ID sampled from the master device.
- All transactions from Master 2 have ID value 4'b1010 (no variable thread bits from the master device).
- When a transaction from Master 0-2 targets Slaves 2-4, the AXI Interconnect 0 passes a 4-bit ID value to Interconnect 1. Interconnect 1 then prefixes it with 1'b0 (the Master ID of its SI slot #0) to produce a 5-bit ID that it passes to any of the connected slave devices.

## Multiple Address Range Support

The AXI Interconnect core must determine which MI slot is the target of each transaction by decoding the address of each AW and AR channel transaction from the SI slots. This address decode involves only those upper-order address bits needed to distinguish between MI slots and ignores lower-order bits that might be used to distinguish locations within the connected slave device. The entire address value received from the SI is presented to the MI and made available to the slave device. It is visible to any connected monitors, even though the high-order address bits are typically not reused by the slave device.



In some cases, there might be multiple, possibly disjoint, address ranges that define when a single slave device (MI slot) is accessed. The address decode logic in the AXI Interconnect core includes the multiple ranges that determine selection of each MI slot. Differentiation between the multiple address ranges is also typically required by the functionality of the connected slave device.

That typically means some of the decode logic implemented by the AXI Interconnect core is replicated in the slave device. The AMBA 4 specification introduces AXI signals `AWREGION` and `ARREGION` that can be used to encode the results of which address range is being decoded by the AXI Interconnect core. The AXI Interconnect core generates these `REGION` outputs for use by slave devices with multiple address decode ranges so that the range decode logic does not need to be replicated in the slave device.

The 4-bit value produced on each `REGION` signal corresponds to the position within the `C_M_AXI_BASE_ADDR` and `C_M_AXI_HIGH_ADDR` parameters, within each MI slot, that matches the transaction address. These address ranges are often represented using multiple parameters on the connected slave device, of a form similar to `C_busif_RNGnn_BASEADDR` and `C_busif_RNGnn_HIGHADDR`. See [Figure 10](#) for an example of how multiple address ranges can be assigned to various MI slots.

Whenever a transaction address received on the SI does not match any of the ranges being decoded by the AXI Interconnect core, the transaction is trapped and handled by a decode error module within the AXI Interconnect core. An exception occurs when the AXI Interconnect core has only one MI slot and only one address range. In that case, the `C_RANGE_CHECK` parameter determines whether address decoding and associated decode error traps are implemented or whether all transactions are propagated to the MI slot.

## Cyclic Dependency Avoidance

Any time there is more than one transaction ID (issued by one or more master devices) on which multiple outstanding transactions can be issued, and there is more than one connected slave device that can queue multiple transactions, and any of the slave devices can respond out-of-order on either the R or B channel, there is a potential cyclic dependency (deadlock) risk. Because the AXI Interconnect core is fully AXI-compliant, the AXI Interconnect core is equipped to handle slave devices that support out-of-order response.

### How Deadlock Occurs

The following example shows how a sequence of Read transactions can result in deadlock. A similar situation also applies to a sequence of Write transactions when a slave device can reorder its Write response. This example shows a case where there are two master devices (M0 and M1) and two slave devices (S0 and S1) connected using the AXI Interconnect core:

1. Master device M0 reads from Slave device S0.
2. Master device M0 then reads from Slave device S1 (using the same ID thread).
3. Master device M1 then reads from Slave device S1.
4. Master device M1 then reads from Slave device S0 (using the same ID thread).
5. Slave device S0 responds to Master device M1 first. It re-orders the Read response, which is allowable because the received transaction IDs are different. However, the AXI Interconnect core cannot pass the response to Master device M1 because Master device M1 must first receive its response from Slave device S1.
6. Slave device S1 responds to Master device M0 (it does not re-order). But the AXI Interconnect core cannot pass the response to Master device M0 because Master device M0 must first receive its response from Slave device S0.

This results in deadlock.

## Avoiding Deadlock Using Single Slave Per ID

The method used in the AXI Interconnect core to avoid deadlock is “Single Slave per ID.” This method does not impact the performance of the transactions of most critical concern. These are the pipelining of multiple Reads and Writes, and by multiple master devices to a performance-critical slave device, such as a memory controller.

The “Single Slave per ID” method imposes the restriction that each ID thread received at each SI slot (from each master device) can have outstanding transactions (of each type) to only one MI slot at a time.

However, MI slots are still permitted to issue multiple outstanding transactions from multiple SI slots.

By imposing this rule in the example shown in the previous section, the Read transaction from M0 to S1 in step 2 is stalled until S0 completes its response to M0. Similarly, the transaction from M1 to S0 in step 4 is stalled until S1 completes its response to M1. Whatever ways the transactions proceed forward under these conditions would avoid the interdependencies that could cause deadlock.

The “Single Slave per ID” restriction applies to all transaction threads whenever the AXI Interconnect core is configured as anything more complex than a 1-to-1 pass-through. Besides preventing deadlock, this restriction also guarantees in-order completion of all Write transactions at the SIs, even if different MI slots are targeted by a transaction thread in successive transactions.

For example, a master device writes to a DMA descriptor in memory, then writes to a control register in a DMA engine that subsequently reads that descriptor. Because the AXI Interconnect core does not allow the second Write to propagate to the DMA slave device until the first Write completes (Write response received from the memory controller), there is no risk that the DMA reads stale descriptor data from memory. Each master device is therefore guaranteed in-order completion of transactions to various slave devices, in the same direction, and on the same thread. Therefore, under those conditions, master devices do not need to condition subsequent Write transactions on receiving Write responses for prior transactions.

**Note:** AXI protocol provides no means to ensure in-order completion between Write and Read transactions other than waiting for the B-Channel responses of all earlier writes to complete.

## Error Signaling

The error conditions detected in the AXI Interconnect core are:

- Address decode error: No eligible MI slot mapped to the address of the transaction, according to the connectivity map and applicable Write-only/Read-only parameters. The AXI Interconnect core returns a DECERR and the transaction is not propagated to any MI slot. However, address decode errors are not trapped when the C\_RANGE\_CHECK parameter is set to 0. By default, C\_RANGE\_CHECK is enabled whenever there are multiple MI slots or if there are multiple address ranges. If the C\_RANGE\_CHECK parameter is forced to OFF (0) when there are multiple MI slots, any access to an illegal address might result in unpredictable and non-compliant transaction propagation.
- AXI4-Lite access violation: Either of the following conditions:
  - Burst length violation: Transaction length >1 data beat when targeting an AXI4-Lite MI slot.
  - Data size violation: Transaction data transfer size wider than 4 bytes when targeting an AXI4-Lite MI slot.

The AXI Interconnect core returns a DECERR, and the transaction is not propagated to the MI slot. AXI4-Lite access violations are disabled when C\_RANGE\_CHECK = 0. By default, C\_RANGE\_CHECK is enabled when any MI slots are configured as AXI4-Lite and any SI slots are configured as any protocol other than AXI4-Lite. If C\_RANGE\_CHECK is OFF (0), and an invalid transaction targets an AXI4-Lite MI slot, the results are unpredictable and the transaction will probably fail.

- An MI slot with `C_M_AXI_SECURE` set is targeted by a transaction in which `AWPROT[1]` or `ARPROT[1]` is set (unsecure).

**Note:** It is illegal to disable `C_RANGE_CHECK` if any MI slots are configured as SECURE.

The AXI Interconnect core does not detect the following error conditions:

- If the response ID received on an MI slot does not map to any SI slot, no READY response is issued from the AXI Interconnect core on the MI slot. The entire response (Write response or Read data burst) is permanently blocked by the AXI Interconnect core. This can cause the problematic slave device and any master device expecting to receive the response to hang.
- The AXI Interconnect core does not trap AXI4 protocol violations, which are the responsibility of the endpoint IP.
- The AXI Interconnect core neither supports nor traps Write data interleaving (all Write data is routed according to write transaction order; `WID` is not sampled at the SI).
- The AXI Interconnect core does not trap narrow burst violations. This occurs when an SI slot is configured with `C_S_AXI_SUPPORTS_NARROW_BURST = 0` and it receives a transaction where there is a length > 1 data beat and the data transfer size is less than the SI slot data width, or any transaction in which `AWCACHE[1]` or `ARCACHE[1]` is deasserted. This is the responsibility of the endpoint master IP.
- Xilinx Platform Studio (XPS) enforces design rules that prevent erroneous configurations at compile time. Therefore, no error detection is provided by the AXI Interconnect core for these configuration errors:
  - Non-integer clock ratios when not configured for asynchronous clocking
  - Parameter value range violations
  - Address or ID range overlap, non-binary size or base value misalignment.

## AXI Protocol Converters

These subsections describe AXI protocol converters:

- [AXI4-Lite Slave Conversion](#)
- [AXI3 Slave Converter](#)

### AXI4-Lite Slave Conversion

Each MI slot connected to an AXI4-Lite slave device routes through an AXI4-Lite conversion block. The conversion block single-threads all transactions, including single-threaded Round-Robin arbitration between Write and Read transactions. In most cases, Write and Read addresses are multiplexed onto a single bus, which is then duplicated onto the `AWADDR` and `ARADDR` signals of the MI slot. In most cases, because these duplicated signals are trimmed during back-end design implementation, the resources used by AXI4-Lite slaves are approximately the same as if there were only one address bus.

The transaction ID (`AWID` or `ARID`) is stripped and stored in the conversion block, and retrieved during response transfers as `BID` or `RID`.

**Note:** The AXI4-Lite protocol conversion does not convert AXI4/AXI3 bursts into a sequence of single-beat transactions for AXI4-Lite slaves to minimize the resource utilization of the AXI4-Lite solution. Master devices accessing AXI4-Lite slave devices are expected to issue only single-beat transactions in which the `SIZE` of the data transfer is not greater than 32 bits.

Figure 11 illustrates the AXI4-Lite conversion logic.

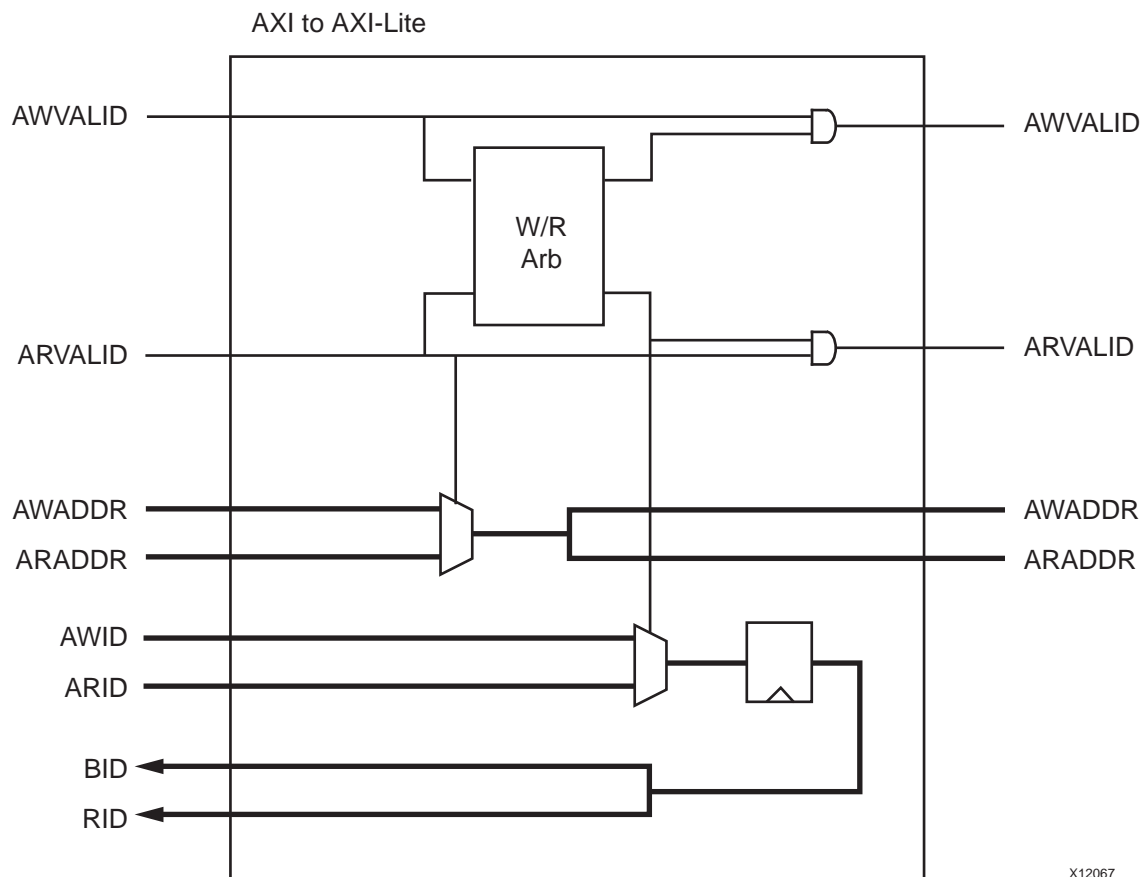


Figure 11: AXI4-Lite Conversion Logic

## AXI3 Slave Converter

A set of AXI3 slave conversion modules is instantiated at each MI slot connected to an AXI3 slave device, if the MI slot is accessible by one or more AXI4 SI slots.

This module receives an AW or AR transfer (command) on its slave interface and produces one or more commands on its MI, similar to the address channel downsizer module. The data transfer `SIZE` is never modified by the AXI3 converter. Any time a burst longer than 16 data beats is received, the command is split into a number of shorter burst transactions.

The AXI3 converter module normally allows multiple outstanding transactions to be propagated. Transaction characteristics from the AW/AR channel transfers are queued while awaiting corresponding response transfers. However, due to the possibility of write response and read data re-ordering, transaction acceptance by the AW and AR channel converter is restricted to a single outstanding transaction at a time (for each direction) whenever the transaction requires splitting.

## I/O Signals

This section lists the AXI Interconnect core signals.

In [Table 3](#), [Table 4](#), [page 29](#), [Table 5](#), [page 31](#), [Table 6](#), [page 32](#), [Table 7](#), [page 33](#), and [Table 8](#), [page 36](#), the “Default” column shows whether the input signal is required (REQ) or, if not, its default value if left unconnected. Signal connections are required only for the SI and MI slots that are used. Values in the Default column also provide the protocol mode of the slot: AXI4 and AXI3, or “Lite” for AXI4-Lite. Input signals that are not sampled (do not care) for AXI4-Lite are indicated by “d/c”.

## Slave Interface I/O Signals

[Table 3](#) lists the Slave Interface signals. In the Width column “N” refers to the total number of SI slots, which is the number of master devices connected to the AXI Interconnect core.

When deployed using the CORE Generator tool flow, each of the signal names listed in [Table 3](#) takes the form “S<sub>nn</sub>\_AXI\_signalname”, where *nn* is the 2-digit index number (with leading zero) of each Slave Interface. In the Width column, N = 1 for all signals on the CORE Generator core interface.

**Table 3: Slave I/O Signals**

Signal Name	Direction	Default	Width	Description (Range)
S_AXI_ARESET_OUT_N	Output		N*1	Reset output (active-Low) resynchronized to each slot's clock. (This is not a signal defined by AXI protocol)
S_AXI_ACLK	Input	REQ	N*1	Clock
S_AXI_AWID	Input	AXI3, AXI4: 0 Lite: d/c	N*C_AXI_ID_WIDTH	Write Address Channel Transaction ID
S_AXI_AWADDR	Input	REQ	N*C_AXI_ADDR_WIDTH	Write Address Channel Address
S_AXI_AWLEN	Input	AXI3, AXI4: 0 Lite: d/c	N*8	Write Address Channel Burst Length (0-255)
S_AXI_AWSIZE	Input	AXI3, AXI4: REQ <sup>(1)</sup> Lite: d/c	N*3	Write Address Channel Transfer Size code (0-7)
S_AXI_AWBURST	Input	AXI3, AXI4: REQ <sup>(1)</sup> Lite: d/c	N*2	Write Address Channel Burst Type code (0-2).

Table 3: Slave I/O Signals (Cont'd)

Signal Name	Direction	Default	Width	Description (Range)
S_AXI_AWLOCK	Input	AXI3, AXI4: 0 Lite: d/c	N*2	Write Address Channel Atomic Access Type (0, 1)
S_AXI_AWCACHE	Input	AXI3, AXI4: 0 <sup>(2)</sup> Lite: d/c	N*4	Write Address Channel Cache Characteristics
S_AXI_AWPROT	Input	0b000 <sup>(3)</sup>	N*3	Write Address Channel Protection Bits
S_AXI_AWQOS <sup>(4)</sup>	Input	AXI4: 0 Lite: d/c	N*4	AXI4 Write Address Channel Quality of Service
S_AXI_AWUSER <sup>(5)</sup>	Input	AXI3, AXI4: 0 Lite: d/c	N*C_AXI_AWUSER_WIDTH	User-defined AW Channel signals
S_AXI_AWVALID	Input	REQ	N*1	Write Address Channel Valid
S_AXI_AWREADY	Output		N*1	Write Address Channel Ready
S_AXI_WID <sup>(5)</sup>	Input	0	N*C_AXI_ID_WIDTH	Write Data Channel Transaction ID for AXI3 masters (sampled and propagated only when the interconnect is in pass-through mode between AXI3 endpoints).
S_AXI_WDATA	Input	REQ	N*C_S_AXI_DATA_WIDTH	Write Data Channel Data
S_AXI_WSTRB	Input	all ones	N*C_S_AXI_DATA_WIDTH/8	Write Data Channel Byte Strobes
S_AXI_WLAST	Input	AXI3, AXI4: 0 Lite: d/c	N*1	Write Data Channel Last Data Beat
S_AXI_WUSER <sup>(5)</sup>	Input	AXI3, AXI4: 0 Lite: d/c	N*C_AXI_WUSER_WIDTH	User-defined W Channel signals
S_AXI_WVALID	Input	REQ	N*1	Write Data Channel Valid.
S_AXI_WREADY	Output		N*1	Write Data Channel Ready.
S_AXI_BID	Output		N*C_AXI_ID_WIDTH	Write Response Channel Transaction ID.
S_AXI_BRESP	Output		N*2	Write Response Channel Response Code (0-3).
S_AXI_BUSER <sup>(5)</sup>	Output		N*C_AXI_BUSER_WIDTH	User-defined B Channel signals.
S_AXI_BVALID	Output		N*1	Write Response Channel Valid.
S_AXI_BREADY	Input	REQ	N*1	Write Response Channel Ready.
S_AXI_ARID	Input	AXI3, AXI4: 0 Lite: d/c	N*C_AXI_ID_WIDTH	Read Address Channel Transaction ID.
S_AXI_ARADDR	Input	REQ	N*C_AXI_ADDR_WIDTH	Read Address Channel Address.
S_AXI_ARLEN	Input	AXI3, AXI4: 0 Lite: d/c	N*8	Read Address Channel Burst Length code (0-255).
S_AXI_ARSIZE	Input	AXI3, AXI4: REQ <sup>(1)</sup> Lite: d/c	N*3	Read Address Channel Transfer Size code (0-7).
S_AXI_ARBURST	Input	AXI3, AXI4: REQ <sup>(1)</sup> Lite: d/c	N*2	Read Address Channel Burst Type (0-2).
S_AXI_ARLOCK	Input	AXI3, AXI4: 0 Lite: d/c	N*2	Read Address Channel Atomic Access Type (0, 1).
S_AXI_ARCACHE	Input	AXI3, AXI4: 0 <sup>(2)</sup> Lite: d/c	N*4	Read Address Channel Cache Characteristics.

Table 3: Slave I/O Signals (Cont'd)

Signal Name	Direction	Default	Width	Description (Range)
S_AXI_ARPROT	Input	0b000 <sup>(3)</sup>	N*3	Read Address Channel Protection Bits.
S_AXI_ARQOS <sup>(4)</sup>	Input	AXI4: 0 Lite: d/c	N*4	AXI4 Read Address Channel Quality of Service.
S_AXI_ARUSER <sup>(5)</sup>	Input	AXI3, AXI4: 0 Lite: d/c	N*C_AXI_ARUSER_WIDTH	User-defined AR Channel signals.
S_AXI_ARVALID	Input	REQ	N*1	Read Address Channel Valid.
S_AXI_ARREADY	Output		N*1	Read Address Channel Ready.
S_AXI_RID	Output		N*C_AXI_ID_WIDTH	Read Data Channel Transaction ID.
S_AXI_RDATA	Output		N*C_S_AXI_DATA_WIDTH	Read Data Channel Data.
S_AXI_RRESP	Output		N*2	Read Data Channel Response Code (0-3).
S_AXI_RLAST	Output		N*1	Read Data Channel Last Data Beat.
S_AXI_RUSER <sup>(5)</sup>	Output		N*C_AXI_RUSER_WIDTH	User-defined R Channel signals.
S_AXI_RVALID	Output		N*1	Read Data Channel Valid.
S_AXI_RREADY	Input	REQ	N*1	Read Data Channel Ready.

1. Xilinx recommends that AXI4 master devices drive their *AW/R*SIZE and *AW/R*BURST outputs. Typically, a master device drives an *AW/R*SIZE value that corresponds to its interface data width, unless application requirements dictate otherwise. Typically, a master device drives its *AW/R*BURST output to 0b01, which indicates an incremental (INCR) burst.
2. Xilinx recommends that master devices drive their *AW/R*CACHE outputs to 0b0011 to allow the AXI Interconnect core to pack data while performing width conversion and to allow store-and-forward in datapath FIFOs.
3. AXI protocol requires master devices to drive their *AW/R*PROT output. If the *AW/R*PROT signals are left undriven, it would default to all zeros and the transaction would be interpreted as secure.
4. Although the QOS signals are defined only by the AXI4 protocol specification, this interconnect IP also propagates QOS signals for any SI slot configured as AXI3.
5. Not applicable when deployed using the CORE Generator tool flow.

## Master Interface I/O Signals

In the Width column of Table 4, “M” refers to the total number of Master Interface (MI) slots, which is the number of slave devices connected to the AXI Interconnect core.

When deployed using the CORE Generator tool flow, each of the signal names listed in Table 4 takes the form “Mmm\_AXI\_signalname”, where *mm* is currently always “00”.

Table 4: Master I/O Signals

Signal Name	Direction	Default	Width	Description (Range)
M_AXI_ARESET_OUT_N	Output		M*1	Reset output (active-low) re-synchronized to each slot's clock. (This is not a signal defined by AXI protocol.)
M_AXI_ACLK	Input	REQ	M*1	Clock.
M_AXI_AWID	Output		M*C_AXI_ID_WIDTH	Write Address Channel Transaction ID.
M_AXI_AWADDR	Output		M*C_AXI_ADDR_WIDTH	Write Address Channel Address.
M_AXI_AWLEN	Output		M*8	Write Address Channel Burst Length code. (0-255).
M_AXI_AWSIZE	Output		M*3	Write Address Channel Transfer Size code (0-7).
M_AXI_AWBURST	Output		M*2	Write Address Channel Burst Type (0-2).
M_AXI_AWLOCK	Output		M*2	Write Address Channel Atomic Access Type (0, 1).
M_AXI_AWCACHE	Output		M*4	Write Address Channel Cache Characteristics.
M_AXI_AWPROT	Output		M*3	Write Address Channel Protection Bits



Table 4: Master I/O Signals (Cont'd)

Signal Name	Direction	Default	Width	Description (Range)
M_AXI_AWREGION <sup>(1)</sup>	Output		M*4	AXI4 Write Address Channel address region index.
M_AXI_AWQOS <sup>(2)</sup>	Output		M*4	Write Address Channel Quality of Service.
M_AXI_AWUSER <sup>(1)</sup>	Output		M*C_AXI_AWUSER_WIDTH	User-defined AW Channel signals.
M_AXI_AWVALID	Output		M*1	Write Address Channel Valid.
M_AXI_AWREADY	Input	REQ	M*1	Write Address Channel Ready.
M_AXI_WID <sup>(1)</sup>	Output		M*C_AXI_ID_WIDTH	Write Data Channel Transaction ID for AXI3 slaves (normally copied from M_AXI_AWID).
M_AXI_WDATA	Output		M*C_M_AXI_DATA_WIDTH	Write Data Channel Data.
M_AXI_WSTRB	Output		M*C_M_AXI_DATA_WIDTH/8	Write Data Channel Data Byte Strobes.
M_AXI_WLAST	Output		1	Write Data Channel Last Data Beat.
M_AXI_WUSER <sup>(1)</sup>	Output		M*C_AXI_WUSER_WIDTH	User-defined W Channel signals.
M_AXI_WVALID	Output		M*1	Write Data Channel Valid.
M_AXI_WREADY	Input	REQ	M*1	Write Data Channel Ready.
M_AXI_BID	Input	AXI3, AXI4: REQ Lite: d/c	M*C_AXI_ID_WIDTH	Write Response Channel Transaction ID.
M_AXI_BRESP	Input	0b00	M*2	Write Response Channel Response Code (0-3).
M_AXI_BUSER <sup>(1)</sup>	Input	AXI3, AXI4: 0 Lite: d/c	M*C_AXI_BUSER_WIDTH	User-defined B Channel signals.
M_AXI_BVALID	Input	REQ	M*1	Write Response Channel Valid.
M_AXI_BREADY	Output		M*1	Write Response Channel Ready.
M_AXI_ARID	Output		M*C_AXI_ID_WIDTH	Read Address Channel Transaction ID.
M_AXI_ARADDR	Output		M*C_AXI_ADDR_WIDTH	Read Address Channel Address.
M_AXI_ARLEN	Output		M*8	Read Address Channel Burst Length code (0-255).
M_AXI_ARSIZE	Output		M*3	Read Address Channel Transfer Size code (0-7).
M_AXI_ARBURST	Output		M*2	Read Address Channel Burst Type (0-2).
M_AXI_ARLOCK	Output		M*2	Read Address Channel Atomic Access Type (0,1).
M_AXI_ARCACHE	Output		M*4	Read Address Channel Cache Characteristics.
M_AXI_ARPROT	Output		M*3	Read Address Channel Protection Bits.
M_AXI_ARREGION <sup>(1)</sup>	Output		M*4	AXI4 Read Address Channel address region index.
M_AXI_ARQOS <sup>(2)</sup>	Output		M*4	AXI4 Read Address Channel Quality of Service.
M_AXI_ARUSER <sup>(1)</sup>	Output		M*C_AXI_ARUSER_WIDTH	User-defined AR Channel signals.
M_AXI_ARVALID	Output		M*1	Read Address Channel Valid.
M_AXI_ARREADY	Input	REQ	M*1	Read Address Channel Ready.
M_AXI_RID	Input	AXI3, AXI4: REQ Lite: d/c	M*C_AXI_ID_WIDTH	Read Data Channel Transaction ID.
M_AXI_RDATA	Input	REQ	M*C_M_AXI_DATA_WIDTH	Read Data Channel Data.
M_AXI_RRESP	Input	0b00	M*2	Read Data Channel Response Code (0-3).
M_AXI_RLAST	Input	AXI3, AXI4: REQ Lite: d/c	M*1	Read Data Channel Last Data Beat.

Table 4: Master I/O Signals (Cont'd)

Signal Name	Direction	Default	Width	Description (Range)
M_AXI_RUSER <sup>(1)</sup>	Input	AXI3, AXI4: 0 Lite: d/c	M*C_AXI_RUSER_WIDTH	User-defined R Channel signals.
M_AXI_RVALID	Input	REQ	M*1	Read Data Channel Valid.
M_AXI_RREADY	Output		M*1	Read Data Channel Ready.

1. Not applicable when deployed using the CORE Generator tool flow.

2. Although the QOS signals are defined only by the AXI4 protocol specification, this interconnect IP also propagates QOS signals for any MI slot configured as AXI3.

## Global Ports

Table 5: Global Port Signals

Port Signal Name	Direction	Default	Width	Description (Range)
INTERCONNECT_ACLK	Input	REQ	1	Interconnect native clock input.
INTERCONNECT_ARESETN	Input	REQ	1	Global Reset (Active Low). See <a href="#">Reset Requirements</a> .

## Reset Requirements

The INTERCONNECT\_ARESETN input must be held active (Low) for a minimum of 16 clock cycles to ensure complete resetting of all internal logic. When multiple clock frequencies are used, INTERCONNECT\_ARESETN must be held active for 16 cycles of the lowest frequency clock connected to the AXI Interconnect core (including the INTERCONNECT\_ACLK frequency). This requirement can be satisfied by driving INTERCONNECT\_ARESETN with the similarly named output port of a proc\_sys\_reset core.

## Design Parameters

The following subsections list the design parameters and conventions used to describe those parameters.

### Conventions in the Parameter Summary Tables

The following conventions are used in the Core, Inherent Slave, and Inherent Master Parameters table ([Table 6](#), [Table 7](#), and [Table 8](#)):

In the Format (Range) column:

- "N" represents the value of C\_NUM\_SLAVE\_SLOTS.
- "M" represents the value of C\_NUM\_MASTER\_SLOTS.
- Braces {} denote a replication count for the value that follows.
- "Bit1" represents a 1-bit value, "Bit32" represents a 32-bit value, and "Bit64" a 64-bit value.  
For example, "{N} Bit32" is used to indicate a parameter consisting of the concatenation of 32-bit values for each SI slot.
- Core parameters influence HDL compilation, unless indicated otherwise by footnote N.

## Global Core Parameters (XPS Flow)

Table 6: Global Core Parameters (XPS Flow)

Parameter Name	Default Value	Format/Range	Description
C_NUM_SLAVE_SLOTS <sup>(T)</sup>	1	Integer (1-16)	Number of SI slots.
C_NUM_MASTER_SLOTS <sup>(T)</sup>	1	Integer (1-16)	Number of MI slots.
C_FAMILY <sup>(T)</sup>	REQ	string	FPGA Family.
C_AXI_ID_WIDTH <sup>(T)</sup>	1	Integer (1-16)	Width of all ID signals propagated by the AXI Interconnect core.
C_AXI_ADDR_WIDTH <sup>(C)</sup>	32	Integer (32)	Width of all ADDR signals for all SI slots and MI slots.
C_S_AXI_IS_INTERCONNECT <sup>(T)</sup>	{N}0b0	{N} Bit1	Used to determine whether CDAM logic is implemented: 0 = connects to an endpoint master device, 1 = connects to another AXI Interconnect core.
C_INTERCONNECT_DATA_WIDTH <sup>(O)</sup>	Same as widest connected MI slot	Integer (32, 64, 128, 256, 512, 1024)	Data width of the internal interconnect Write and Read datapaths.
C_INTERCONNECT_ACLK_RATIO <sup>(T)</sup>	1	Integer (1-2147483647)	Clock frequency ratio of the internal AXI Interconnect core with regard to all SI and MI slots. (Tools set this value to the Interconnect clock frequency in Hz.)
C_AXI_SUPPORTS_USER_SIGNALS <sup>(O)</sup>	0	Integer	Indicates whether or not to propagate the USER signals (all 5 channels) across the AXI Interconnect core. 0 = not propagate 1 = propagate
C_AXI_AWUSER_WIDTH <sup>(O)</sup>	1	Integer (1-256)	Width of AWUSER signals for all AXI4 SI slots and MI slots.
C_AXI_ARUSER_WIDTH <sup>(O)</sup>	1	Integer (1-256)	Width of ARUSER signals for all AXI4 SI slots and MI slots.
C_AXI_WUSER_WIDTH <sup>(O)</sup>	1	Integer (1-256)	Width of WUSER signals for all AXI4 SI slots and MI slots.
C_AXI_RUSER_WIDTH <sup>(O)</sup>	1	Integer (1-256)	Width of RUSER signals for all AXI4 SI slots and MI slots.
C_AXI_BUSER_WIDTH <sup>(O)</sup>	1	Integer (1-256)	Width of BUSER signals for all AXI4 SI slots and MI slots.
C_AXI_CONNECTIVITY <sup>(T)</sup>	all ones	{M} Bit32 {N} Bit1	Sparse crossbar connectivity from each SI slot (N) to each MI slot (M). (Used only when Interconnect is in Crossbar mode.) 0 = no pathway required 1 = pathway required

### Notes:

I = Inherent parameter available on all connected master devices.  
 U = User specified.  
 T = Tool generated (EDK originates the information and sets the value.)  
 C = Constant  
 O = Tool generated or TCL automated with user override.  
 N = Not used by the HDL of the core.

Table 6: Global Core Parameters (XPS Flow) (Cont'd)

Parameter Name	Default Value	Format/Range	Description
C_INTERCONNECT_CONNECTIVITY_MODE <sup>(U)</sup>	1	Integer (0,1)	Defines the interconnect architecture: 0 = Shared Access (Area optimized) 1 = Crossbar (Performance optimized)
C_RANGE_CHECK <sup>(O)</sup>	ON (1) if C_NUM_MASTER_SLOTS > 1, or if C_M_AXI_BASE/HIGH_ADDR defines more than 1 range, or if any MI slots are AXI4-Lite while any SI slots are non-AXI4-Lite, or if C_M_AXI_SECURE is set for any MI slot; otherwise OFF (0).	Integer (0,1)	Determines whether the AXI Interconnect core detects various transaction error conditions: 0 (OFF) = Do not detect any DECERR conditions. See <a href="#">Decode Error Detection, page 51</a> . 1 (ON) = Trap transaction errors and produce DECERR response.

**Notes:**

I = Inherent parameter available on all connected master devices.  
 U = User specified.  
 T = Tool generated (EDK originates the information and sets the value.)  
 C = Constant  
 O = Tool generated or TCL automated with user override.  
 N = Not used by the HDL of the core.

## Slave Interface Parameters (XPS Flow)

Table 7: Slave Interface-Related Parameters (XPS Flow)

Parameter Name	Default Value	Format/Range	Description
C_S_AXI_PROTOCOL <sup>(M)</sup>	{N} 0x00000000	{N} Bit32	AXI protocol of connected master device: 0 = SI slot is AXI4 1 = SI slot is AXI3 2 = SI slot is AXI4-Lite
C_S_AXI_DATA_WIDTH <sup>(M)</sup>	{N} 0x00000020	{N} Bit32 (0x00000020, 0x00000040, 0x00000080, 0x00000100, 0x00000200, 0x00000400)	Effective width of S_AXI_WDATA and S_AXI_RDATA for each SI slot. (Must be 0x20 for AXI4-Lite SI slots)
C_S_AXI_BASE_ID <sup>(I,O)</sup>	{N} 0x00000000	{N} Bit32 (0-0xFFFF)	Base ID of each SI slot (N-1:0).
C_S_AXI_THREAD_ID_WIDTH <sup>(M)</sup>	{N} 0x00000000	{N} Bit32 (0-0x10)	Number of variable low-order ID bits of each SI slot (N-1:0). Each value must be <= C_AXI_ID_WIDTH.
C_S_AXI_SINGLE_THREAD <sup>(I,U)</sup>	{N} 0b0	{N} Bit1	ID-thread support by SI slot: 0 = Accept multiple outstanding thread ID values (performance optimized). 1 = Accept only one outstanding thread ID value at a time (area optimized).

**Notes:**

I = Inherent parameter available on all connected master devices.  
 M = Value is copied from a parameter that exists on the connected master device.  
 T = Tool generated (EDK originates the information and sets the value.)  
 C = Constant  
 O = Tool generated with user override (EDK originates the information and sets the value; user can override.)  
 N = Not used by the HDL of the core.  
 U = User specified.

Table 7: Slave Interface-Related Parameters (XPS Flow) (Cont'd)

Parameter Name	Default Value	Format/Range	Description
C_S_AXI_ACLK_RATIO <sup>(I,T)</sup>	{N}0x00000001	{N} Bit32 (0x1-0x7FFFFFFF)	Clock frequency ratio of each SI slot with regard to internal interconnect, if synchronous. (Tools set this value to the SI clock frequency in Hz.)
C_S_AXI_IS_ACLK_ASYNC <sup>(I,O)</sup>	{N}d, where the default, d, for each SI slot is 0 if the ratio (C_S_AXI_ACLK_RATIO[slot] : C_INTERCONNECT_ACLK_RATIO) is 1:k or k:1, where k is an integer 1-16; otherwise d = 1.	{N} Bit1	Indicates if the SI slot clock is synchronous or asynchronous to AXI Interconnect native clock. 0 = SI slot clock is synchronous to AXI Interconnect native clock 1 = SI slot clock is asynchronous to interconnect native clock
C_S_AXI_ARB_PRIORITY <sup>(I,U)</sup>	{N}0x00000000	{N} Bit32 (0x00000000 -0x0000000f)	Arbitration priority among each SI slot. Higher values indicate higher priority. All slots with value 0 participate in round-robin arbitration.
C_S_AXI_WRITE_ACCEPTANCE <sup>(I,U)</sup>	{M}0x00000001	{M} Bit32 (0x1, 0x2, 0x4, 0x8, 0x10, 0x20)	Number of data-active Write transactions that each AXI SI slot can generate
C_S_AXI_READ_ACCEPTANCE <sup>(I,U)</sup>	{M}0x00000001	{M} Bit32 (0x1, 0x2, 0x4, 0x8, 0x10, 0x20)	Number of active Read transactions that each AXI SI slot can generate
C_S_AXI_SUPPORTS_WRITE <sup>(M)</sup>	{N}0b1	{N}Bit1	Indicates whether each SI slot uses Write-related channels. 0 = Read-only 1 = Use AW, W, and B channels
C_S_AXI_SUPPORTS_READ <sup>(M)</sup>	{N}0b1	{N}Bit1	Indicates whether each SI slot uses Read-related channels. 0 = Write-only 1 = Use AR and R channels
C_S_AXI_SUPPORTS_NARROW_BURST <sup>(M, N)</sup>	{N}0b1	{N} Bit1	Indicates whether the connected master device can produce narrow bursts. 0 = all bursts are same size as data width and A*CACHE[1]=1 always (does not apply to single-beat transfers). 1 = can produce narrow bursts or deassert A*CACHE[1].
C_S_AXI_WRITE_FIFO_DEPTH <sup>(I,U)</sup>	{N}0x00000000	{N} Bit32 (0x00000000, 0x00000020, 0x00000200)	Depth of SI-side Write data FIFO (before W channel arbitration) for each SI slot.
C_S_AXI_WRITE_FIFO_DELAY <sup>(I,U)</sup>	{N}0b0	{N} Bit1	Delay issuing AWVALID to the arbiter until the complete burst is stored in the SI-side write data FIFO for each SI slot. (Requires C_S_AXI_WRITE_FIFO_DEPTH = 0x200 for corresponding slot.)

**Notes:**

I = Inherent parameter available on all connected master devices.  
M = Value is copied from a parameter that exists on the connected master device.  
T = Tool generated (EDK originates the information and sets the value.)  
C = Constant  
O = Tool generated with user override (EDK originates the information and sets the value; user can override.)  
N = Not used by the HDL of the core.  
U = User specified.

Table 7: Slave Interface-Related Parameters (XPS Flow) (Cont'd)

Parameter Name	Default Value	Format/Range	Description
C_S_AXI_READ_FIFO_DEPTH (I,U)	{N}0x00000000	{N} Bit32 (0x00000000, 0x00000020, 0x00000200)	Depth of SI-side Read data FIFO (after R channel routing) for each SI slot.
C_S_AXI_READ_FIFO_DELAY (I,U)	{N}0b0	{N} Bit1	Delay issuing ARVALID to the arbiter until the SI-side read data FIFO has enough vacancy to store the entire burst length for each SI slot. (Requires C_S_AXI_READ_FIFO_DEPTH = 0x200 for corresponding slot.)
C_S_AXI_AW_REGISTER (I,U)	{N}0x00000000	{N} Bit32	Insert register slice on AW channel at each SI slot interface. 0 = Bypass 1 = Fully_Registered 7 = Light_Weight 8 = Automatic
C_S_AXI_AR_REGISTER (I,U)	{N}0x00000000	{N} Bit32	Insert register slice on AR channel at each SI slot interface. 0 = Bypass 1 = Fully_Registered 7 = Light_Weight 8 = Automatic
C_S_AXI_W_REGISTER (I,U)	{N}0x00000000	{N} Bit32	Insert register slice on W channel at each SI slot interface. 0 = Bypass 1 = Fully_Registered 7 = Light_Weight 8 = Automatic
C_S_AXI_R_REGISTER (I,U)	{N}0x00000000	{N} Bit32	Insert register slice on R channel at each SI slot interface. 0 = Bypass 1 = Fully_Registered 7 = Light-Weight 8 = Automatic
C_S_AXI_B_REGISTER (I,U)	{N}0x00000000	{N} Bit32	Insert register slice on B channel at each SI slot interface. 0 = Bypass 1 = Fully_Registered 7 = Light-Weight 8 = Automatic

**Notes:**

I = Inherent parameter available on all connected master devices.  
M = Value is copied from a parameter that exists on the connected master device.  
T = Tool generated (EDK originates the information and sets the value.)  
C = Constant  
O = Tool generated with user override (EDK originates the information and sets the value; user can override.)  
N = Not used by the HDL of the core.  
U = User specified.

## Master Interface Parameters (XPS Flow)

Table 8: Master Interface-Related Parameters (XPS Flow)

Parameter Name	Default Value	Format/Range	Description
C_M_AXI_PROTOCOL (S)	{M} 0x00000000	{M} Bit32	AXI protocol of connected slave device: 0 = MI slot is AXI4 1 = MI slot is AXI3 2 = MI slot is AXI4-Lite
C_M_AXI_DATA_WIDTH (S)	{M}0x00000020	{M} Bit32 (0x00000020, 0x00000040, 0x00000080, 0x00000100, 0x00000200, 0x00000400)	Effective width of M_AXI_WDATA and M_AXI_RDATA for each MI slot. (Must be 0x20 for AXI4-Lite MI slots.)
C_M_AXI_BASE_ADDR (I,U)	{M} (16)0xffffffff	{M} (16) Bit64	Base address of each range (15:0) of each MI slot (M-1:0). For unused ranges, set base address to 0xffffffff.
C_M_AXI_HIGH_ADDR (I,U)	{M} (16) 0x00000000_00000000	{M} (16) Bit64	High address of each range (15:0) of each MI slot (M-1:0). For unused ranges, set high address to 0x00000000_00000000.
C_M_AXI_ACLK_RATIO (I,T)	{M}0x00000001	{M} Bit32 (0x1-0x7FFFFFFF)	Clock frequency ratio of each MI slot with regard to internal AXI Interconnect core, if synchronous. (Tools set this value to the MI clock frequency in Hz.)
C_M_AXI_IS_ACLK_ASYNC (I,O)	{M}d, where the default, d, for each MI slot is 0 if the ratio (C_M_AXI_ACLK_RATIO[slot] : C_INTERCONNECT_ACLK_RATIO) is 1:k or k:1, where k is an integer 1-16; otherwise d = 1.	{M} Bit1	Indicates if the MI slot clock is synchronous or asynchronous to AXI Interconnect native clock. 0 = MI slot clock is synchronous 1 = MI slot clock is asynchronous
C_M_AXI_SUPPORTS_WRITE (S)	{M}0b1	{M}Bit1	Indicates whether each MI slot uses Write-related channels. 0 = Read-only 1 = Use AW, W, and B channels
C_M_AXI_SUPPORTS_READ (S)	{M}0b1	{M}Bit1	Indicates whether each MI slot Read-related channels. 0 = Write-only 1 = AR and R channels
C_M_AXI_WRITE_ISSUING (I,U)	{M}0x00000001	{M} Bit32 (0x1, 0x2, 0x4, 0x8, 0x10, 0x20)	Number of data-active Write transactions that each AXI4 MI slot can generate
C_M_AXI_READ_ISSUING (I,U)	{M}0x00000001	{M} Bit32 (0x1, 0x2, 0x4, 0x8, 0x10, 0x20)	Number of active Read transactions that each AXI4 MI slot can generate

### Notes:

- I = Inherent parameter available on all connected master devices.
- S = Value is copied from a parameter that exists on the connected slave device.
- U = User specified.
- T = Tool generated (EDK originates the information and sets the value.)
- C = Constant
- N = Not used by the HDL of the core.
- O = Tool generated with user override (EDK originates the information and sets the value; user can override.)



Table 8: Master Interface-Related Parameters (XPS Flow) (Cont'd)

Parameter Name	Default Value	Format/Range	Description
C_M_AXI_SECURE (I,U)	{M}0b0	{M} Bit1	Indicates whether each MI slot connects to a secure slave device (allows TrustZone secure access). 0 = non-secure slave device 1 = secure slave device
C_M_AXI_SUPPORTS_NARROW_BURST (S, N)	{M}0b1	{M} Bit1	Indicates whether the connected slave device is configured to support bursts where the transfer size is narrower than the data width. 0 = connected slave device does not tolerate bursts that are a different SIZE than the MI-slot data-width (does not apply to single-beat transfers). 1 = connected slave device supports narrow bursts.
C_M_AXI_WRITE_FIFO_DEPTH (I,U)	{M}0x00000000	{M} Bit32 (0x00000000, 0x00000020, 0x00000200)	Depth of MI-side Write data FIFO (after W channel routing) for each MI slot 0x0 = no FIFO 0x20 = 32-deep LUT-RAM based FIFO 0x200 = 512-deep block RAM based FIFO
C_M_AXI_WRITE_FIFO_DELAY (I,U)	{M}0b0	{M} Bit1	Delay issuing M_AXI_AWVALID until the complete burst is stored in the MI-side write data FIFO for each MI slot. (Requires C_M_AXI_WRITE_FIFO_DEPTH = 0x200 for corresponding slot.)
C_M_AXI_READ_FIFO_DEPTH (I,U)	{M}0x00000000	{M} Bit32 (0x00000000, 0x00000020, 0x00000200)	Depth of MI-side Read data FIFO (before R channel arbitration) for each MI slot 0x0 = no FIFO 0x20 = 32-deep LUT-RAM based FIFO 0x200 = 512-deep block RAM based FIFO
C_M_AXI_READ_FIFO_DELAY (I,U)	{M}0b0	{M} Bit1	Delay issuing M_AXI_ARVALID until the MI-side read data FIFO has enough vacancy to store the entire burst length for each MI slot. (Requires C_M_AXI_READ_FIFO_DEPTH = 0x200 for corresponding slot.)
C_M_AXI_AW_REGISTER (I,U)	{M}0x00000000	{M} Bit32	Insert register slice on AW channel at each MI slot interface. 0 = Bypass, 1 = Fully_registered 7 = Light-weight, 8 = Automatic
C_M_AXI_AR_REGISTER (I,U)	{M}0x00000000	{M} Bit32	Insert register slice on AR channel at each MI slot interface. 0 = Bypass, 1 = Fully_registered 7 = Light-weight, 8 = Automatic
C_M_AXI_W_REGISTER (I,U)	{M}0x00000000	{M} Bit32	Insert register slice on W channel at each MI slot interface: 0 = Bypass, 1 = Fully_Registered 7 = Light_Weight, 8 = Automatic

**Notes:**

I = Inherent parameter available on all connected master devices.  
S = Value is copied from a parameter that exists on the connected slave device.  
U = User specified.  
T = Tool generated (EDK originates the information and sets the value.)  
C = Constant  
N = Not used by the HDL of the core.  
O = Tool generated with user override (EDK originates the information and sets the value; user can override.)

**Table 8: Master Interface-Related Parameters (XPS Flow) (Cont'd)**

Parameter Name	Default Value	Format/Range	Description
C_M_AXI_R_REGISTER (I,U)	{M}0x00000000	{M} Bit32	Insert register slice on R channel at each MI slot interface: 0 = Bypass, 1 = Fully_Registered 7 = Light_Weight, 8 = Automatic
C_M_AXI_B_REGISTER (I,U)	{M}0x00000000	{M} Bit32	Insert register slice on B channel at each MI slot interface: 0 = Bypass, 1 = Fully_Registered 7 = Light_Weight, 8 = Automatic

**Notes:**

- I = Inherent parameter available on all connected master devices.
- S = Value is copied from a parameter that exists on the connected slave device.
- U = User specified.
- T = Tool generated (EDK originates the information and sets the value.)
- C = Constant
- N = Not used by the HDL of the core.
- O = Tool generated with user override (EDK originates the information and sets the value; user can override.)

## Global Parameters (CORE Generator Flow)

Table 9: Global Parameters (CORE Generator Flow)

Parameter Name	Default Value	Format/Range	Description
C_NUM_SLAVE_PORTS	2	Integer (1-16)	Number of Slave Interfaces.
C_FAMILY	REQ	string	FPGA Family.
C_THREAD_ID_WIDTH	0	Integer (0-8)	Number of ID bit to be sampled (if any) on all Slave Interfaces.
C_THREAD_ID_PORT_WIDTH	1	Integer (1-8)	Width of ID signals on all Slave Interfaces. (Same as C_THREAD_ID_WIDTH except it cannot be zero.)
C_AXI_ADDR_WIDTH	32	Integer (12-64)	Width of all ADDR signals for all SIs and MIs.
C_INTERCONNECT_DATA_WIDTH	Same as C_M00_AXI_DATA_WIDTH	Integer (32, 64, 128, 256, 512, 1024)	Data width of the internal interconnect Write and Read datapaths.

## Slave Interface Parameters (CORE Generator Flow)

Table 10: Slave Interface-Related Parameters (CORE Generator Flow)

Parameter Name	Default Value	Format/Range	Description
C_Snn_AXI_DATA_WIDTH	32	Integer (32, 64, 128, 256, 512, 1024)	Width of Snn_AXI_WDATA and Snn_AXI_RDATA signals.
C_Snn_AXI_ACLK_RATIO	"1:1"	String ("1:16"..."1:1"..."16:1")	Clock frequency ratio of each SI with regard to internal interconnect, if synchronous.
C_Snn_AXI_IS_ACLK_ASYNC	0	Bit1	Indicates if the SI clock is synchronous or asynchronous to AXI Interconnect native clock. 0 = SI clock is synchronous to AXI Interconnect native clock 1 = SI clock is asynchronous to interconnect native clock
C_Snn_AXI_ARB_PRIORITY	0	Integer (0-15)	Arbitration priority among each SI. Higher values indicate higher priority. All SIs with value 0 participate in round-robin arbitration.
C_Snn_AXI_WRITE_ACCEPTANCE	1	Integer (1, 2, 4, 8, 16, 32)	Number of data-active Write transactions that each AXI SI slot can generate.
C_Snn_AXI_READ_ACCEPTANCE	1	Integer (1, 2, 4, 8, 16, 32)	Number of active Read transactions that each AXI SI slot can generate.
C_Snn_AXI_READ_WRITE_SUPPORT	"READ/WRITE"	String ("READ/WRITE", "READ-ONLY", "WRITE-ONLY")	Indicates whether each SI uses Write-related and/or Read-related channels.
C_Snn_AXI_WRITE_FIFO_DEPTH	0	Integer (0, 32, 512)	Depth of SI-side Write data FIFO (before W channel multiplexing). 0 = No FIFO 32 = 32-deep LUT-RAM based FIFO 512 = 512-deep block RAM based FIFO
C_Snn_AXI_WRITE_FIFO_DELAY	0	Bit1	Delay issuing AWVALID to the arbiter until the complete burst is stored in the SI-side write data FIFO. (Requires C_Snn_AXI_WRITE_FIFO_DEPTH = 512.)
C_Snn_AXI_READ_FIFO_DEPTH	0	Integer (0, 32, 512)	Depth of SI-side Read data FIFO (after R channel routing). 0 = no FIFO 32 = 32-deep LUT-RAM based FIFO 512 = 512-deep block RAM based FIFO

Table 10: Slave Interface-Related Parameters (CORE Generator Flow) (Cont'd)

Parameter Name	Default Value	Format/Range	Description
C_Snn_AXI_READ_FIFO_DELAY	0	Bit1	Delay issuing ARVALID to the arbiter until the SI-side read data FIFO has enough vacancy to store the entire burst length. (Requires C_Snn_AXI_READ_FIFO_DEPTH = 512.)
C_Snn_AXI_REGISTER	0	Bit1	Insert register slices on all SI channels. Fully registered for W and R channels, Light-weight for AW, AR, and B channels

## Master Interface Parameters (CORE Generator Flow)

Table 11: Master Interface-Related Parameters (CORE Generator Flow)

Parameter Name	Default Value	Format/Range	Description
C_M00_AXI_DATA_WIDTH	32	Integer (32, 64, 128, 256, 512, 1024)	Width of M00_AXI_WDATA and M00_AXI_RDATA signals.
C_M00_AXI_ACLK_RATIO	"1:1"	String ("1:16"..."1:1"..."16:1")	Clock frequency ratio of MI with regard to internal interconnect, if synchronous.
C_M00_AXI_IS_ACLK_ASYNC	0	Bit1	Indicates if the MI clock is synchronous or asynchronous to AXI Interconnect native clock. 0 = MI clock is synchronous to AXI Interconnect native clock 1 = MI clock is asynchronous to interconnect native clock
C_Mnn_AXI_READ_WRITE_SUPPORT	"READ/WRITE"	String ("READ/WRITE", "READ-ONLY", "WRITE-ONLY")	Indicates whether the MI uses Write-related and/or Read-related channels.
C_M00_AXI_WRITE_ISSUING	1	Integer (1, 2, 4, 8, 16, 32)	Number of data-active Write transactions that the MI can issue.
C_M00_AXI_READ_ISSUING	1	Integer (1, 2, 4, 8, 16, 32)	Number of active Read transactions that the MI can issue.
C_M00_AXI_WRITE_FIFO_DEPTH	0	Integer (0, 32, 512)	Depth of MI-side Write data FIFO (after W channel multiplexing) 0 = no FIFO 32 = 32-deep LUT-RAM based FIFO 512 = 512-deep block RAM based FIFO
C_M00_AXI_WRITE_FIFO_DELAY	0	Bit1	Delay issuing M00_AXI_AWVALID until the complete burst is stored in the MI-side write data FIFO. (Requires C_M00_AXI_WRITE_FIFO_DEPTH = 512.)
C_M00_AXI_READ_FIFO_DEPTH	0	Integer (0, 32, 512)	Depth of MI-side Read data FIFO (before R channel routing) 0 = no FIFO 32 = 32-deep LUT-RAM based FIFO 512 = 512-deep block RAM based FIFO
C_M00_AXI_READ_FIFO_DELAY	0	Bit1	Delay issuing M00_AXI_ARVALID until the MI-side read data FIFO has enough vacancy to store the entire burst length. (Requires C_M00_AXI_READ_FIFO_DEPTH = 512.)
C_M00_AXI_REGISTER	0	Bit1	Insert register slices on all MI channels. Fully registered for W and R channels, Light-weight for AW, AR, and B channels

## AXI Interconnect Parameter Usage

The following tables provide further definition and usage details on the various design parameters, their values, effects, and interactions with other parameters. The parameter descriptions also indicate the circumstances in which parameters on the AXI Interconnect core are copied or derived from parameters found on connected master and slave devices.

### Interface Protocol

Interconnect	Connected Master	Connected Slave
C_S_AXI_PROTOCOL	C_busif_PROTOCOL	
C_M_AXI_PROTOCOL		C_busif_PROTOCOL

The \*\_**PROTOCOL** parameter identifies the interface sub-protocol (AXI4, AXI3, or AXI4-Lite) of the AMBA AXI specification. Typically, the parameter is specified as a constant in the MPD of the connected master and slave IP. However, some IP supports configurable protocol (typically AXI4 vs. AXI4-Lite), which is then user-selectable.

The AXI Interconnect core uses the **Protocol** parameter to:

- Insert optional protocol conversion modules
- Impose error detection logic
- In the case of AXI4-Lite, conserve logic resources associated with unused AXI4 interface features

The tools copy these values from the connected masters and slaves onto the AXI Interconnect core.

### Data Width

Interconnect	Connected Master	Connected Slave
C_S_AXI_DATA_WIDTH	C_busif_DATA_WIDTH	
C_M_AXI_DATA_WIDTH		C_busif_DATA_WIDTH
C_INTERCONNECT_DATA_WIDTH		

The **C\_S\_AXI\_DATA\_WIDTH** and **C\_M\_AXI\_DATA\_WIDTH** parameters indicate the widths of the WDATA and RDATA signals on the connected master and slave devices, respectively. The tools copy these values from the connected masters and slaves onto the AXI Interconnect core.

The **C\_INTERCONNECT\_DATA\_WIDTH** parameter specifies the native data width of the internal crossbar. By default, the tools set this to match the widest connected MI slot. Users can override this with any supported value (regardless of connected device widths).

When the value of:

- **C\_S\_AXI\_DATA\_WIDTH** for an SI slot is less than **C\_INTERCONNECT\_DATA\_WIDTH**, an upsizer module is inserted in the pathways from that SI slot in the SI hemisphere of the AXI Interconnect core (between the SI and the crossbar).
- **C\_S\_AXI\_DATA\_WIDTH** is greater than **C\_INTERCONNECT\_DATA\_WIDTH**, a downsizer module is inserted in the SI hemisphere.
- **C\_M\_AXI\_DATA\_WIDTH** differs from **C\_INTERCONNECT\_DATA\_WIDTH**, the appropriate width converter is inserted in the MI hemisphere (between the crossbar and the MI).

Data width converters perform the packing and serialization of data required to adapt the differing data widths, and consequently affect the data bandwidth along the pathways between the SI, crossbar and MI.

Selecting a sufficiently high value of **C\_INTERCONNECT\_DATA\_WIDTH** can avoid loss of data bandwidth.

For example, you could elect to set the `C_INTERCONNECT_DATA_WIDTH` to match the width of a speed-critical slave, such as a memory controller, even though all the masters that access that slave have narrower data widths. With that setting, the AXI Interconnect core can perform data packing (Write transactions) or serialization (Read transactions) concurrently along multiple SI slot pathways in the SI hemisphere while the wide slave device and crossbar periodically maintain a data throughput rate higher than any one master device can sustain.

In contrast, selecting a lower value of `C_INTERCONNECT_DATA_WIDTH` can reduce logic resource utilization for less speed-critical designs. When seeking to minimize resource utilization, set the `C_INTERCONNECT_DATA_WIDTH` so that it minimizes the total number of width converters (upsizers and downsizers).

The interconnect does not support downsizing directly from 1024 bits to 32 bits, either between the SI and crossbar or between the crossbar and MI. If any SI is 1024 bits wide, then `C_INTERCONNECT_DATA_WIDTH` must be set to a value greater than 32. If the MI is 32 bits wide, then `C_INTERCONNECT_DATA_WIDTH` must be set to a value less than 1024.

## Clock Frequency

Interconnect	Connected Master	Connected Slave
<code>C_S_AXI_ACLK_RATIO</code>	<code>C_busif_ACLK_RATIO</code> (also <code>C_busif_ACLK_FREQ_HZ</code> )	
<code>C_M_AXI_ACLK_RATIO</code>		<code>C_busif_ACLK_RATIO</code> (also <code>C_busif_ACLK_FREQ_HZ</code> )
<code>C_S_AXI_IS_ACLK_ASYNC</code>	<code>C_busif_IS_ACLK_ASYNC</code>	
<code>C_M_AXI_IS_ACLK_ASYNC</code>		<code>C_busif_IS_ACLK_ASYNC</code>
<code>C_INTERCONNECT_ACLK_RATIO</code>		

The XPS tools keep track of the various clock signals in an embedded system using the `CLK_FREQ_HZ` property (sometimes represented by a parameter `C_busif_ACLK_FREQ_HZ`) associated with the clock port (ACLK) which is associated with each bus interface.

The AXI Interconnect core also has a global clock port, `INTERCONNECT_ACLK`, that synchronizes the crossbar and other internal modules. The tools survey the `CLK_FREQ_HZ` values among all the connected masters and slaves, compare them to the frequency of the `INTERCONNECT_ACLK` port, and determine the clock frequency relationships between each of the SI and MI slots and the crossbar.

The clock frequency relationships are determined in the tools as follows:

- When the relationship is an integer ratio (faster or slower) within the range 1:16 to 16:1, the tools set the corresponding `IS_ACLK_ASYNC` parameter to zero (synchronous); otherwise, the slot is tagged as asynchronous.
- The tools assign the `C_INTERCONNECT_ACLK_RATIO` parameter the value of the `CLK_FREQ_HZ` property of the `INTERCONNECT_ACLK` port. They also assign `C_S_AXI_ACLK_RATIO` and `C_M_AXI_ACLK_RATIO` the value of `CLK_FREQ_HZ` for the SI and MI ACLK ports, respectively. In the core, only the ratio between these parameters is significant, and only where the corresponding `C_S/M_AXI_IS_ACLK_ASYNC` = 0.
- When an SI slot is asynchronous (`C_S_AXI_IS_ACLK_ASYNC` = 1) or its clock ratio is different than that of the `INTERCONNECT_ACLK` (`C_S_AXI_ACLK_RATIO` != `C_INTERCONNECT_ACLK_RATIO`), then a clock converter module is inserted in the pathways from that SI slot in the SI hemisphere of the AXI Interconnect core (between the SI and the crossbar).
- When an MI slot is asynchronous, or has a clock ratio different than the AXI Interconnect core, a clock converter module is inserted in the MI hemisphere (between the crossbar and the MI).

- When  $C\_S/M\_AXI\_IS\_ACLK\_ASYN = 0$ , a synchronous clock converter is used to resolve differing ratios, which minimizes latency and resources, and enforces the correct timing constraints for pathways that cross between the clock domains.
- When  $C\_S/M\_AXI\_IS\_ACLK\_ASYN = 1$ , a clock converter based on an asynchronous FIFO is used, which eliminates timing relationships between signals in each clock domain.

Clock converters always introduce additional latency. Passing through a clock converter in both the SI and MI hemispheres when traversing any pathway between an SI and MI is wasteful; select clock frequencies to avoid passing through a clock converter in both hemispheres whenever possible.

To reduce the number of clock converters in the system, it can be advantageous to cascade AXI Interconnect instances, grouping together similarly clocked devices. For example, connecting a group of low-frequency AXI4-Lite slaves to a separate AXI Interconnect core clocked at the same low frequency could consolidate the clock domain crossing onto a single converter in the pathway between the cascaded AXI Interconnect instances.

When speed-critical devices, such as a memory controller, are connected to an AXI Interconnect core, best data throughput is usually achieved by clocking the `INTERCONNECT_ACLK` port with the same clock source as the speed-critical slave.

## Address Ranges

Interconnect	Connected Master	Connected Slave
<code>C_M_AXI_BASE_ADDR</code>		<code>C_busif_BASEADDR</code> or <code>C_busif_RNGnn_BASEADDR</code>
<code>C_M_AXI_HIGH_ADDR</code>		<code>C_busif_HIGHADDR</code> or <code>C_busif_RNGnn_HIGHADDR</code>
<code>C_AXI_ADDR_WIDTH</code>	<code>C_busif_ADDR_WIDTH</code>	<code>C_busif_ADDR_WIDTH</code>

For all unused address ranges, the corresponding values in `C_M_AXI_BASE_ADDR` are set to all ones (for as many low-order bits as specified by `C_AXI_ADDR_WIDTH`, or more), and the corresponding values in `C_M_AXI_HIGH_ADDR` are set to all zeros.

For each used address range:

- The range size ( $HIGH\_ADDR - BASE\_ADDR + 1$ ) must be a minimum of 4k.
- The range size must be a power of 2.
- `BASE_ADDR` must be a multiple of the range size (aligned).
- No overlapping address ranges across the entire address decode table (all MI slots).

All address range restrictions are enforced by the tools used to configure the AXI Interconnect core, and are therefore not enforced by error checking provided by the core itself.

When two Interconnect instances are cascaded, the address decoder in the upstream Interconnect uses multiple address ranges for the cascaded MI slot to represent the collection of address ranges for all the downstream slave devices that are accessible through the downstream Interconnect instance.

It is important to use separate address regions to represent different endpoint slave devices, and avoid combining more than one slave device into the same address decoder region, even if they are mapped to adjacent or nearby regions of the system address map. This allows the Cyclic Dependency Avoidance Method (CDAM) to enforce the policy that each ID thread received at each SI slot can produce outstanding Write or Read transactions to only one endpoint slave device at a time. See [Multiple Address Range Support, page 22](#) and [Figure 10, page 22](#) for an example of how multiple address ranges should be defined for cascaded MI slots.



Embedded hardware systems support 32-bit addresses only; therefore, the value of `C_AXI_ADDR_WIDTH` is the constant 32.

**Note:** The AXI Interconnect core does not support address remapping. Therefore, if any multi-ported slave devices (such as multi-ported memory controllers) are used in the system, for which multiple bus interfaces share the same or overlapping address ranges, those multiple bus interfaces must be connected to different instances of AXI Interconnect, and those Interconnect instances *must not* be cascaded to each other. The tools enforce the rule that multiple address ranges within the same Interconnect must not overlap.

## ID Ranges

Interconnect	Connected Master	Connected Slave
<code>C_S_AXI_BASE_ID</code>	<code>C_INTERCONNECT_busif_BASE_ID</code>	
<code>C_S_AXI_THREAD_ID_WIDTH</code>	<code>C_busif_SUPPORTS_THREADS</code> , <code>C_busif_THREAD_ID_WIDTH</code>	
<code>C_AXI_ID_WIDTH</code>		<code>C_busif_ID_WIDTH</code>

The ID values route response transfers back to the correct SI slot and to the connected master device.

Master devices supporting a transaction reordering depth > 1 can issue multiple “threads” of transactions by driving different values on their `AWID` and `ARID` outputs.

The various ID parameters are related as follows:

- The `C_S_AXI_THREAD_ID_WIDTH` vector parameter specifies the number of ID bits produced by each master device. Masters with reordering depth of 1 produce no ID signals and the corresponding value of `C_S_AXI_THREAD_ID_WIDTH` is set to zero. Any thread ID bits sampled at an SI slot are used as the low-order bits of the complete transaction ID signal used throughout the Interconnect core and propagated to the MI.
- The `C_AXI_ID_WIDTH` global parameter specifies the width of the complete transaction ID signal used throughout the Interconnect and propagated by all MI slots, and must include enough high-order bits (Master ID) to uniquely distinguish between all the SI slots.
- The `C_S_AXI_BASE_ID` parameter defines the base (lowest) ID value corresponding to each SI slot.
- In the `C_S_AXI_BASE_ID` parameter values, the number of low-order bit positions indicated by `C_S_AXI_THREAD_ID_WIDTH` must all be zero.
- The remaining high-order bits of `C_S_AXI_BASE_ID` are considered to be the “Master ID”, and remain constant for all transactions propagated from each SI slot.
- The complete transaction ID signal used throughout the Interconnect and propagated by all MI slots is formed by ORing the thread ID bits (if any) sampled by an SI slot with the SI slot’s `C_S_AXI_BASE_ID` value.

The EDK tools determine the required value of the AXI Interconnect `C_AXI_ID_WIDTH` parameter and assign unique Master ID values among all the SI slots based upon the values of the `C_S_AXI_BASE_ID` and `C_S_AXI_THREAD_ID_WIDTH` parameters found on each SI slot.

The AXI Interconnect core uses the `C_S_AXI_BASE_ID` and `C_S_AXI_THREAD_ID_WIDTH` parameters to implement the decode logic for routing responses on the R and B channels back to the proper SI slot.

When an SI slot connects to an MI slot of another (upstream) AXI Interconnect instance, the range of ID values that can be received on the ID signals of that slot consists of all the ID values on all the SI slots of the upstream AXI Interconnect core (those that have connection pathways to the connected MI slot). ID signals produced by the upstream AXI Interconnect core are treated as thread ID bits of a connected master device.

As with other master devices, the downstream AXI Interconnect prefixes the ID signals sampled from a cascaded SI slot with a unique master ID, causing the ID width to grow as it propagates forward across a cascaded AXI Interconnect topology. All responses matching that master ID are routed back to the upstream AXI Interconnect.



Similar to address ranges, each ID range comprises an aligned, binary-sized range, as all the low-order bits of the `BASE_ID` parameter (as defined by `THREAD_ID_WIDTH`) must be zero.

There must be no overlap among any ID ranges across the entire ID decode table (all SI slots). Unlike address ranges, ID ranges have a minimum size of 1.

Specifying thread ID width parameters on endpoint masters vs. interconnect:

- When the `C_S_AXI_THREAD_ID_WIDTH` value for an SI slot of the AXI Interconnect core is zero, the size of the corresponding ID range is a single ID value and no ID signals are sampled at the SI slot.
- On the connected master device, the corresponding `C_busif_THREAD_ID_WIDTH` parameter is often used to determine the physical bit-width of the ID signals on the IP interface, and therefore must never be set to a value less than 1.
- The `C_busif_SUPPORTS_THREADS` parameter is used by the endpoint master to indicate when the effective thread width of the master is zero.
- When `C_busif_SUPPORTS_THREADS = 0` on a connected master, `C_S_AXI_THREAD_ID_WIDTH` is set to zero for the corresponding SI slot on the AXI Interconnect core.

## Transaction Acceptance and Issuing Limits

Interconnect	Connected Master	Connected Slave
<code>C_S_AXI_WRITE_ACCEPTANCE</code>	<code>C_INTERCONNECT_busif_WRITE_ISSUING</code>	
<code>C_S_AXI_READ_ACCEPTANCE</code>	<code>C_INTERCONNECT_busif_READ_ISSUING</code>	
<code>C_M_AXI_WRITE_ISSUING</code>		<code>C_INTERCONNECT_busif_WRITE_ACCEPTANCE</code>
<code>C_M_AXI_READ_ISSUING</code>		<code>C_INTERCONNECT_busif_READ_ACCEPTANCE</code>

The `C_S_AXI_WRITE_ACCEPTANCE` and `C_S_AXI_READ_ACCEPTANCE` parameters limit the number of current outstanding transactions of each type that the crossbar will accept, per ID thread. For each ID value that can be accepted at each SI slot (based on the values of `C_S_AXI_THREAD_ID_WIDTH` and `C_S_AXI_SINGLE_THREAD`), the crossbar maintains a separate Write and Read transaction counter. There is no limit on the total number of transactions (across all threads) that the crossbar will accept. Also, the ACCEPTANCE limit parameters do not take into account the number of address transfers that might be accepted and stored in buffer modules, such as register slices and clock converters, that could be implemented along the address channel in the SI hemisphere, prior to arriving at the crossbar.

The `C_M_AXI_WRITE_ISSUING` and `C_M_AXI_READ_ISSUING` parameters limit the total number of current outstanding transactions of each type that the crossbar issues (with any ID value). The ISSUING limit parameters do not take into account the number of address transfers that might be accepted and stored in buffer modules, such as register slices and clock converters, that could be implemented along the address channel in the MI hemisphere, after being issued by the crossbar.

Regarding acceptance and issuing counters:

- A Write transaction is considered to be complete (counter decremented) when a `BVALID/BREADY` handshake completes at the crossbar
- A Read transaction is considered to be complete when an `RVALID/RREADY` handshake completes at the crossbar in which `RLAST` is asserted

Write or Read transactions received at SI slots that have reached their acceptance limit, or that target an MI slot that has reached its issuing limit, are disqualified from arbitration so that the Write or Read arbiter, respectively, can continue to grant arbitration to other qualified SI slots, instead of stalling.

Increasing the value of an ACCEPTANCE or ISSUING parameter can increase data throughput by allowing Write and Read commands to be pipelined in connected slave devices, thus avoiding idle cycles on the Write and Read data channels. However, increasing the parameter values too much could lead to head-of-line blocking when multiple master devices access a shared slave.

When the tools copy the parameter values from the connected master and slave devices to the AXI Interconnect core:

- ISSUING parameters on connected master devices map to ACCEPTANCE parameters on the SI of the AXI Interconnect core
- ACCEPTANCE parameters on connected slave devices map to ISSUING parameters on the MI of the AXI Interconnect core

**Note:** For AXI4-Lite SI slots and MI slots, the ACCEPTANCE and ISSUING parameters, respectively, are ignored and only one outstanding transaction at a time is allowed.

### Arbitration Priority

Interconnect	Connected Master	Connected Slave
C_S_AXI_ARB_PRIORITY	C_INTERCONNECT_busif_ARB_PRIORITY	

Primarily, arbitration is granted based on the relative priority of the associated SI slot. The C\_S\_AXI\_ARB\_PRIORITY parameter can be set to a static priority value between 0 and 15; higher values have higher priority. In case of a tie:

- When the priority level of all qualified requesting SI slots have level 0, arbitration among them is decided by round-robin.
- When the highest priority value among the requesting SI slots is greater than 0, priority among slots sharing that priority value is based on their slot number; lower slot numbers have higher priority.

Write or Read transactions received at SI slots that have reached their acceptance limit or target an MI slot that has reached its issuing limit are temporarily disqualified from arbitration so that the Write or Read arbiter, respectively, can continue to grant arbitration to other qualified SI slots, rather than stalling (see [Transaction Acceptance and Issuing Limits](#), page 45.)

Also, any SI slot that attempts to access an MI slot in a manner that risks deadlock, is temporarily disqualified from arbitration (see [Cyclic Dependency Avoidance](#), page 23.)

### Crossbar Connectivity

Interconnect	Connected Master	Connected Slave
C_AXI_CONNECTIVITY		C_INTERCONNECT_busif_MASTERS
C_INTERCONNECT_CONNECTIVITY_MODE		
C_S_AXI_SINGLE_THREAD	C_INTERCONNECT_busif_SINGLE_THREAD	
C_S_AXI_IS_INTERCONNECT		

The AXI Interconnect core supports two connectivity architectures, as selected by the C\_INTERCONNECT\_CONNECTIVITY\_MODE parameter.

When set to Shared Access mode (C\_INTERCONNECT\_CONNECTIVITY\_MODE = 0), the AXI Interconnect core provides for only one outstanding transaction at a time. Shared Access mode minimizes the resources used to implement the crossbar module of the Interconnect in applications that do not require the performance of concurrent data transfers.

When set to Crossbar mode (`C_INTERCONNECT_CONNECTIVITY_MODE = 1`), the AXI Interconnect core supports “sparse crossbar” connectivity, which means that the user can specify which SI slots need to access which MI slots. By disabling unused paths, datapath multiplexing logic and address decoding logic can be reduced, resulting in reduced FPGA resource utilization and faster timing paths.

When configured in Crossbar mode, the remaining connectivity parameters are set as follows:

- Sparse connectivity information is represented by the `C_AXI_CONNECTIVITY` parameter on the AXI Interconnect core.
- The `C_AXI_CONNECTIVITY` value is derived by accumulating the `C_INTERCONNECT_busif_MASTERS` parameters found on all the connected slave devices.
- The `C_INTERCONNECT_busif_MASTERS` parameter on each slave device lists all the master devices that need to access that slave, according to the instance name of each master device and the name of the bus interface on each master instance.

When using the XPS GUI to establish the bus connectivity of a system, the tools automatically insert `C_INTERCONNECT_busif_MASTERS` parameters in the Microprocessor Hardware System (MHS) design file.

Also, the `C_S_AXI_IS_INTERCONNECT` parameter affects crossbar connectivity. This parameter indicates whether any of the slave interface (SI) slots are being driven by another Interconnect.

Two AXI Interconnect cores can be cascaded by using the `axi2axi_connector` core to bridge an MI slot of an upstream AXI Interconnect to an SI slot of a downstream Interconnect.

Under certain conditions, the tools set the value of the `C_S_AXI_IS_INTERCONNECT` parameter automatically when the tools detect an `axi2axi_connector` on an SI slot; no user intervention is required. This parameter is used by the downstream AXI Interconnect to avoid duplication of transaction thread control logic that is performed by the upstream AXI Interconnect, when possible.

The `C_S_AXI_SINGLE_THREAD` parameter simplifies thread control logic on a per-SI-slot basis by allowing one or more outstanding transactions from only one thread ID at a time. This could save resources, especially if the SI slot connects to a master device or upstream Interconnect that produces very wide ID signals.

The various connectivity parameters are interrelated as shown in [Table 12](#).

**Table 12: Relationship among Connectivity Parameters**

<code>C_INTERCONNECT_CONNECTIVITY_MODE</code>	<code>C_S_AXI_SINGLE_THREAD</code>	<code>C_S_AXI_IS_INTERCONNECT</code>	Description
0	x	x	Shared Access architecture. Interconnect accepts one transaction at a time.
1	1	x	SI slot is single-threaded. SI slot accepts multiple transactions, but only one thread ID value at a time.
1	0	0	SI slot is multi-threaded. SI slot accepts transactions on multiple ID threads at a time, but only one target MI slot per thread.
1	0	1	SI slot is multi-threaded. SI slot accepts transactions on multiple ID threads at a time, and assumes an upstream interconnect accepts only one targeted endpoint slave device per thread.

**Note:** XPS tools assign consecutive SI slot numbers to connected masters automatically, and consecutive MI slot numbers to connected slaves, according to the order in which the master and slave instances and their bus interfaces appear in the MHS file. There is no mechanism to allow users to explicitly control the assignment of slot numbers or to preserve slot number assignments across design iterations when master/slave instances and/or their bus-interface connections are added, deleted, or replaced. The AXI Interconnect core does not support “holes” in the SI or MI.

## Read-Only and Write-Only Interfaces

Interconnect	Connected Master	Connected Slave
C_S_AXI_SUPPORTS_WRITE	C_busif_SUPPORTS_WRITE	
C_S_AXI_SUPPORTS_READ	C_busif_SUPPORTS_READ	
C_M_AXI_SUPPORTS_WRITE		C_busif_SUPPORTS_WRITE
C_M_AXI_SUPPORTS_READ		C_busif_SUPPORTS_READ

Typically, the Read/Write-only parameters are specified as a constant in the MPD of the connected master/slave IP. However, some IP might support configurable read-only or write-only operation, which is then selectable by the user. The following rules apply to disabling the support for Reads and Writes:

- When the `SUPPORTS_WRITE` parameter is disabled, the connectivity pathways for the AW, W, and B channels of the corresponding SI or MI slot are disabled, similar to the way the `C_AXI_CONNECTIVITY` parameter disables pathways between specified pairs of SI slots and MI slots.
- When the `SUPPORTS_READ` parameter is disabled, the connectivity pathways for the AR and R channels of the corresponding SI or MI slot are disabled.
- When unused Write and Read channels are disabled, the datapath multiplexing logic and address decoding logic can be reduced, resulting in reduced FPGA resource utilization and faster timing paths.
- For each SI slot and each MI slot, `SUPPORTS_WRITE` or `SUPPORTS_READ` (or both) must be enabled.

For slave devices that can be configured to support Read and Write transactions, the tools set the corresponding parameter values based on the values found on the connected master devices that access those slave devices.

## Register Slices

Interconnect	Connected Master	Connected Slave
C_S_AXI_AW_REGISTER	C_INTERCONNECT_busif_AW_REGISTER	
C_S_AXI_AR_REGISTER	C_INTERCONNECT_busif_AR_REGISTER	
C_S_AXI_W_REGISTER	C_INTERCONNECT_busif_W_REGISTER	
C_S_AXI_R_REGISTER	C_INTERCONNECT_busif_R_REGISTER	
C_S_AXI_B_REGISTER	C_INTERCONNECT_busif_B_REGISTER	
C_M_AXI_AW_REGISTER		C_INTERCONNECT_busif_AW_REGISTER
C_M_AXI_AR_REGISTER		C_INTERCONNECT_busif_AR_REGISTER
C_M_AXI_W_REGISTER		C_INTERCONNECT_busif_W_REGISTER
C_M_AXI_R_REGISTER		C_INTERCONNECT_busif_R_REGISTER
C_M_AXI_B_REGISTER		C_INTERCONNECT_busif_B_REGISTER

Setting a `REGISTER` parameter enables a register slice at the outer-most periphery of the SI or MI. Each channel of each interface slot can be enabled independently. Register slices are provided mainly to improve system timing at the expense of one latency cycle.

The modes selections are as follows:

- FULLY\_REGISTERED:** Implemented as a 2-deep FIFO buffer, this mode supports throttling by the channel source and/or channel destination as well as back-to-back transfers without incurring bubble cycles.
- LIGHT\_WEIGHT:** Implemented as a simple 1-stage pipeline register, this mode minimizes resources while isolating timing paths, but always incurs 1 bubble cycle following each transfer.
- AUTOMATIC:** For AXI4-Lite slots, use `LIGHT_WEIGHT` for all channels. For other protocols, use `FULLY_REGISTERED` on W and R channels, and use `LIGHT_WEIGHT` on AW, AR, and B channels.

## Datapath FIFOs

Interconnect	Connected Master	Connected Slave
C_S_AXI_WRITE_FIFO_DEPTH	C_INTERCONNECT_busif_WRITE_FIFO_DEPTH	
C_S_AXI_READ_FIFO_DEPTH	C_INTERCONNECT_busif_READ_FIFO_DEPTH	
C_M_AXI_WRITE_FIFO_DEPTH		C_INTERCONNECT_busif_WRITE_FIFO_DEPTH
C_M_AXI_READ_FIFO_DEPTH		C_INTERCONNECT_busif_READ_FIFO_DEPTH
C_S_AXI_WRITE_FIFO_DELAY	C_INTERCONNECT_busif_WRITE_FIFO_DELAY	
C_S_AXI_READ_FIFO_DELAY	C_INTERCONNECT_busif_READ_FIFO_DELAY	
C_M_AXI_WRITE_FIFO_DELAY		C_INTERCONNECT_busif_WRITE_FIFO_DELAY
C_M_AXI_READ_FIFO_DELAY		C_INTERCONNECT_busif_READ_FIFO_DELAY

Setting a `*_FIFO_DEPTH` parameter inserts a FIFO buffer on the data channel in either the SI or MI hemisphere, directly adjacent to the crossbar. The value of the parameter sets the depth of the FIFO (0 = no FIFO). The width of the data stored in each FIFO always matches the native data width of the crossbar (`C_INTERCONNECT_DATA_WIDTH`).

By default, FIFO buffers are only inserted in the W and/or R data channels. When a `*_FIFO_DELAY` parameter is enabled, a 32-deep FIFO buffer (based on distributed RAM) is also inserted into the corresponding AW or AR address channel. (The write response channel has no FIFO capability.)

When a `*_WRITE_FIFO_DELAY` parameter is enabled, an AW channel transfer is read from the AW FIFO only after `WLAST` is received by the W channel FIFO. That delays the issuing of the write transaction until the entire write burst is stored in the FIFO and is available for transfer, thus avoiding stalling due to a slow write data source.

When a `*_READ_FIFO_DELAY` parameter is enabled, an AR channel transfer is read from the AR FIFO only when the vacancy in the R channel FIFO is at least the length of the entire burst, according to `ARLEN`. (The “vacancy” is the amount of free space in the R channel FIFO that has not already been committed by previously issued AR commands.) That delays the issuing of the read transaction until it can be guaranteed that the entire read burst can be stored in the FIFO, thus avoiding stalling due to a slow read data destination.

## TrustZone Security

Interconnect	Connected Master	Connected Slave
C_M_AXI_SECURE		C_INTERCONNECT_busif_SECURE

Setting the `C_M_AXI_SECURE` parameter enforces TrustZone security for each designated MI slot (connected slave) as a whole. If configured as a secure slot, only secure AXI accesses are permitted (`S_AXI_AWPROT[1]` or `M_AXI_AWPROT[1]` must be zero). Any non-secure accesses are blocked and the AXI Interconnect core generates a `DECERR` response on the SI. If the connected slave device provides its own TrustZone security feature, then you would generally not set the `C_M_AXI_SECURE` parameter on the AXI Interconnect.

## Narrow Burst Support

Interconnect	Connected Master	Connected Slave
C_S_AXI_SUPPORTS_NARROW_BURST	C_busif_SUPPORTS_NARROW_BURST	
C_M_AXI_SUPPORTS_NARROW_BURST		C_busif_SUPPORTS_NARROW_BURST

Setting the `C_S_AXI_SUPPORTS_NARROW_BURST` parameter to zero for an SI slot indicates that the connected master device is “well-behaved” and never issues any multi-beat transactions in which the `SIZE` of the data transfers is narrower than the interface data width (“narrow bursts”). For example, a “well-behaved” master that has 64-bit wide `WDATA/RDATA` signals always drives its `AWSIZE/ARSIZE` to 0x011 (8 bytes) when issuing transactions in which `AWLEN/ARLEN > 0` (single-beat transactions can be of any `SIZE`.) For all connected masters that are not “well-behaved”, the `C_S_AXI_SUPPORTS_NARROW_BURST` parameter must remain set to one (default).

Also, when `C_S_AXI_SUPPORTS_NARROW_BURST = 0`, the master device must never deassert `AWCACHE[1]` or `ARCACHE[1]` (“Modifiable” bit) for any transaction.

This allows any upsizers in the pathways driven by the SI slot to fully pack data so that the AXI Interconnect core never produces narrow bursts on the MI for any transactions that originate from the “well-behaved” SI slot.

The AXI Interconnect core does not detect or provide a `DECERR` response if a narrow burst is received on any SI slot for which `C_S_AXI_SUPPORTS_NARROW_BURST = 0`, as it is a design requirement of the master IP to avoid narrow bursts.

Setting the `C_M_AXI_SUPPORTS_NARROW_BURST` parameter to zero for an MI slot indicates that the connected slave device can expect to receive no narrow bursts and can therefore omit the associated data packing logic.

For slave devices that feature configurable support for narrow burst, the XPS tools set their `C_busif_SUPPORTS_NARROW_BURST` parameter to zero automatically when all master devices that can access the slave are well-behaved (masters have their `C_busif_SUPPORTS_NARROW_BURST` set to zero).

The AXI Interconnect core itself does not use either of the `SUPPORTS_NARROW_BURST` parameters. All upsizers (in this version of Interconnect) always pack multi-beat bursts (provided the Modifiable bit is set, as required) so that narrow bursts are never created by the interconnect.

## User Signals

Interconnect	Connected Master	Connected Slave
<code>C_AXI_SUPPORTS_USER_SIGNALS</code>	<code>C_busif_SUPPORTS_USER_SIGNALS</code>	<code>C_busif_SUPPORTS_USER_SIGNALS</code>
<code>C_AXI_AWUSER_WIDTH</code>	<code>C_busif_AWUSER_WIDTH</code>	<code>C_busif_AWUSER_WIDTH</code>
<code>C_AXI_ARUSER_WIDTH</code>	<code>C_busif_ARUSER_WIDTH</code>	<code>C_busif_ARUSER_WIDTH</code>
<code>C_AXI_WUSER_WIDTH</code>	<code>C_busif_WUSER_WIDTH</code>	<code>C_busif_WUSER_WIDTH</code>
<code>C_AXI_RUSER_WIDTH</code>	<code>C_busif_RUSER_WIDTH</code>	<code>C_busif_RUSER_WIDTH</code>
<code>C_AXI_BUSER_WIDTH</code>	<code>C_busif_BUSER_WIDTH</code>	<code>C_busif_BUSER_WIDTH</code>

When these parameters are specified on connected master and slave devices, they indicate whether those device interfaces include `USER` signals (on any channels) and, if so, the width of the `USER` signal on each channel.

By surveying the parameter values among connected master and slave devices, the tools determine if the AXI Interconnect core needs to propagate `USER` signals between the SI and MI (on any channels) and, if so, the maximum required width of the `USER` signals to propagate on each channel. The tools therefore set the values of the `USER` parameters automatically on the AXI Interconnect core based on the values found on the connected masters and slaves.

Generally, the various `USER_WIDTH` parameter are used to determine the physical bit-width of the `USER` signals on the IP interface, and therefore must never be set to a value less than 1. When `C_AXI_SUPPORTS_USER_SIGNALS = 1`, at least 1 bit of `USER` signal is propagated on each of the five AXI channels.



Propagation of WUSER and RUSER signals is supported only for SI slots and MI slots for which the interface data width matches the native data width of the interconnect (C\_INTERCONNECT\_DATA\_WIDTH). RUSER and WUSER signals are always blocked (forced to all zeros) by upszers and downszers. Due to the possibility of transaction splitting, the BUSER signal is not propagated by downszers or AXI3 protocol converters. Propagation of AWUSER and ARUSER signals is supported for all SI and MI slots, regardless of data width.

## Decode Error Detection

AXI Interconnect Parameter	Description
C_RANGE_CHECK	<p>Determines whether the AXI Interconnect core detects various transaction error conditions as follows:</p> <ul style="list-style-type: none"> <li>When ON (1), causes transactions received at the SI to be examined for various error conditions that might result in the AXI Interconnect core responding with a DECERR code on either the BRESP or RRESP signal.</li> <li>When OFF (0), the logic that produces DECERR responses is omitted from the AXI Interconnect implementation, saving resources.</li> </ul>

By default, C\_RANGE\_CHECK is turned ON (1) when:

- There are multiple MI slots
- There are multiple address ranges defined for any MI slot
- There are any AXI4-Lite MI slots (connected slaves) and there are any non-AXI4-Lite SI slots (connected masters)
- The C\_M\_AXI\_SECURE parameter is set for any MI slot

The above default conditions ensure that any protocol-compliant transaction received at the SI result in either a protocol-compliant completion or a proper DECERR response. When none of the above conditions are true, DECERR logic is omitted (by default), resulting in logic reduction. This allows an Interconnect in a “pass-through” configuration to be implemented as wires, consuming no resources.

The C\_RANGE\_CHECK parameter can be forced OFF to save logic resources when the system designer is sure that the connected masters will never issue certain transactions that would normally trigger a DECERR response. This condition is satisfied when all of the following are true:

- No master accesses a non-existent slave. This is true when either:
  - There is only one MI slot (connected slave), regardless of its address range
  - All transactions have addresses that are covered by the configured address map, taking into account the sparse-crossbar connectivity map and any read-only/write-only slaves
  - The address map covers the entire address space, as defined by C\_AXI\_ADDR\_WIDTH
- No master accesses an AXI4-Lite slave with either of the following inappropriate transactions:
  - Transaction length >1 data beat
  - Data transfer size wider than 4 bytes
- No MI slots are configured with C\_M\_AXI\_SECURE enabled, regardless of the behavior of the masters. It is illegal to turn C\_RANGE\_CHECK OFF if any MI slots are configured as SECURE, and any attempt to do so results in a compile time error.

The C\_RANGE\_CHECK parameter can be forced ON to trap invalid transaction addresses when there is only one MI slot with only one address range.

The following conditions are not detected by the AXI Interconnect core, and are therefore unaffected by C\_RANGE\_CHECK:

- Unrecognized response ID errors
- AXI4 protocol violations (including AXI3 atomic LOCK transactions)
- Write data interleaving
- Narrow burst violations (see C\_S\_AXI\_SUPPORTS\_NARROW\_BURST)
- Non-integer clock ratios when not configured for asynchronous clocking
- Parameter value range violations
- Address or ID range overlap, non-binary size or base value misalignment

### Parameter Rules Summary

In addition to the value ranges described in the parameter tables, these specific rules apply:

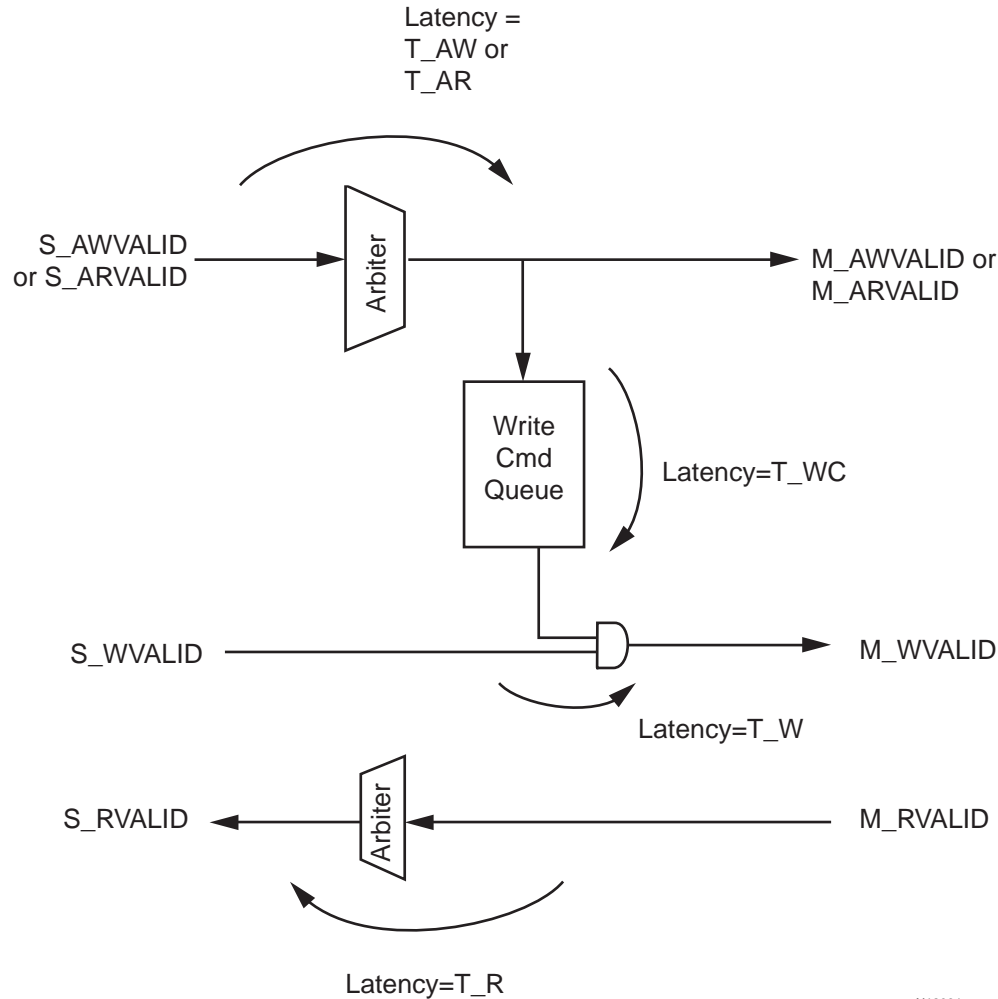
- If any SI is 1024 bits wide, C\_INTERCONNECT\_DATA\_WIDTH must be set to a value greater than 32. If the MI is 32 bits wide, C\_INTERCONNECT\_DATA\_WIDTH must be set to a value less than 1024.
- The C\_S\_AXI\_DATA\_WIDTH of each slot must be 32 when C\_S\_AXI\_PROTOCOL denotes AXI4-Lite. Each C\_M\_AXI\_DATA\_WIDTH must be 32 when C\_M\_AXI\_PROTOCOL denotes AXI4-Lite.
- For each used MI slot, at least one address range must be defined (non-null).
- For each address range, the range size must be a power of 2 and at least 4096. BASE\_ADDR must be a multiple of the range size (all low-order bits that select locations within the range must be zero). The low-order bits (within the range) of HIGH\_ADDR must be all ones.
- There must be no overlap among any of the address ranges across all MI slots.
- For each used SI slot, parameters C\_S\_AXI\_BASE\_ID and C\_S\_AXI\_THREAD\_ID\_WIDTH must be defined.
- For each SI slot configured as AXI4-Lite, C\_S\_AXI\_IS\_INTERCONNECT must be 0 (endpoint master device) and C\_S\_AXI\_THREAD\_ID\_WIDTH must be 0. (AXI4-Lite interfaces use no ID signals.)
- For each ID range, all low-order bits of the BASE\_ID parameter (as defined by THREAD\_ID\_WIDTH), if any, must be zero. In other words, BASE\_ID must be a multiple of the range size ( $2^{**} \text{THREAD\_ID\_WIDTH}$ ).
- There must be no overlap among any of the ID ranges across all SI slots.
- The upper bound of each ID range ( $\text{BASE\_ID} + 2^{**} \text{THREAD\_ID\_WIDTH} - 1$ ) must not exceed the maximum ID value ( $2^{**} \text{C\_AXI\_ID\_WIDTH} - 1$ ) of the AXI Interconnect.
- For each SI slot and each MI slot, SUPPORTS\_WRITE or SUPPORTS\_READ (or both) must be enabled.



## Performance

### Latency

Figure 12 shows the baseline latency model for the crossbar module (when not in bypass mode).



X12091

Figure 12: Crossbar Model Baseline Latency

In Figure 12, the baseline latency is as follows:

- $T_{AW} = T_{AR} = 2$  cycles of `INTERCONNECT_ACLK`, for the forward propagation of AW/ARVALID, provided there are no pending conditions that would inhibit granting arbitration (such as a higher-priority request). Each arbitration also causes 2 bubble cycles, resulting in 3 cycles (minimum) between successive arbitrations by the same SI slot.
- $T_{WC} = 1$  cycle of `INTERCONNECT_ACLK`.
- For each MI slot, if `C_M_AXI_PROTOCOL != AXI4Lite` and `C_M_AXI_DATA_WIDTH != C_INTERCONNECT_DATA_WIDTH`, then  $T_W = 1$  cycle of `INTERCONNECT_ACLK`, with no bubble cycles (supports continuous back-to-back data transmission); otherwise,  $T_W = 0$  cycles (combinatorial through crossbar).
- $T_R = 1$  or 2 cycles of `INTERCONNECT_ACLK`, with no bubble cycles (supports continuous back-to-back data transmission). The second latency cycle occurs when there is a re-arbitration (when the requesting MI slot is

different than the last granted MI slot) following an idle cycle. While the same MI slot propagates back-to-back data, or while multiple MI slots continuously interleave data, latency through the R-channel arbiter is 1 cycle.

- T\_B (B-channel latency, not shown) = 1 or 2 cycles of INTERCONNECT\_ACLK. (Same as for T\_R.)

Additional latency cycles are added for various optional modules outside the crossbar, including:

- “FULLY\_REGISTERED” register slices (each applicable channel): 1 latency cycle of S\_AXI\_ACLK or M\_AXI\_ACLK, with no bubble cycles (best-case 100% channel bandwidth).
- “LIGHT\_WEIGHT” register slices (each applicable channel): 1 latency cycle of S\_AXI\_ACLK or M\_AXI\_ACLK, with one bubble cycle (best-case 50% channel bandwidth), which is appropriate for AW, AR, and B channel transfers, and all transfers involving AXI4-Lite endpoints.
- Data FIFOs:
  - W and R channels: 3 latency cycles of INTERCONNECT\_ACLK, with no bubble cycles.
  - AW, AR, and B channels: no latency.
- Clock conversion: latency varies.
- Upsizers:
  - AW and AR channels: 1 latency cycle.
  - W channel: 1 latency cycle (for each cycle in which packing completes), with no bubble cycles on the SI-side (narrow) interface.
  - R channel: 1 latency cycle.
  - B channel: no latency.
  - Clocking:
    - Upsizers in the SI hemisphere are clocked by INTERCONNECT\_ACLK.
    - Upsizers in the MI hemisphere are clocked by M\_AXI\_ACLK.
- Downsizers:
  - AW and AR channels: 1 latency cycle.
  - R channel: no latency (for each cycle in which packing completes), with no bubble cycles on the MI-side (narrow) interface.
  - W channel: no latency.
  - B channel: no latency.
  - Clocking:
    - Downsizers in the SI-hemisphere are clocked by S\_AXI\_ACLK.
    - Downsizers in the MI-hemisphere are clocked by INTERCONNECT\_ACLK.
- AXI4-Lite conversion: no latency on any channels.
- AXI3 conversion:
  - AW and AR channels: 1 latency cycle of M\_AXI\_ACLK.
  - W, R, and B channels: no latency.

## Clock Frequency

This section describes the expected clock frequencies of the AXI Interconnect for the Spartan-6 and Virtex-6 families. The actual achievable clock frequency can vary due to FPGA logic utilization, physical placement constraints (such as I/Os), tool options, and other factors.

## Nx1 Configurations

These configuration options represent applications providing access by multiple master devices to a single high-performance slave device, such as a memory controller:

- Up to four SI slots and one MI slot
- 64-bit MI data width
- 64-bit internal data width
- Upsizers to handle 32-bit wide SI slots, as needed
- Clock converters, as needed
- Datapath FIFOs, as needed

The realizable in-system clock frequencies of the AXI Interconnect in such applications are:

- Virtex-6 device with -1 speed grade: 200 MHz
- Spartan-6 device with -3 speed grade: 133 MHz

## Other Configurations

For larger configurations, including NxM crossbar with up to eight SI slots, the Interconnect typically supports these clock frequencies:

- Virtex-6 FPGA with -1 speed grade: 167 MHz
- Spartan-6 FPGA with -3 speed grade: 111 MHz

## Resource Utilization

Table 13 and Table 14 indicate the estimated FPGA resource utilization for various modules within the AXI Interconnect core. Some typical configurations of each module are listed. The overall area of a given instance of AXI Interconnect can be estimated by accumulating the utilizations of all constituent modules.

Table 13: Virtex-6 FPGA Resource Utilization

Module	Configuration	FFs	LUTs	Block RAM
Crossbar (SAMD)	4 SI x 1 MI, Data width = 32 bits	300	740	
Crossbar (SAMD)	4 SI x 1 MI, Data width = 64 bits	390	940	
Crossbar (SAMD)	1 SI x 4 MI, Data width = 32 bits	340	400	
Shared-Access Crossbar (SASD)	1 SI x 4 MI, Data width = 32 bits	35	150	
Crossbar (SAMD)	4 SI x 4 MI, Data width = 32 bits	560	1530	
Upsizer	Data width = 32 to 64 bits	420	560	
Downsizer	Data width = 64 to 32 bits	560	620	
Sync Clock Converter	Data width = 32 bits	180	70	
Sync Clock Converter	Data width = 64 bits	330	110	
Async Clock Converter	Data width = 32 bits	720	350	
Async Clock Converter	Data width = 64 bits	870	400	
Datapath FIFO	32-deep (LUT RAM), Data width = 32 bits	110	60	
Datapath FIFO	32-deep (LUT RAM), Data width = 64 bits	180	90	
Datapath FIFO	512-deep (Block RAM), Data width = 32 bits	90	60	1
Datapath FIFO	512-deep (Block RAM), Data width = 64 bits	130	60	2
Register slice	AW/AR channel, Light-weight (type 7)	30	5	

Table 13: Virtex-6 FPGA Resource Utilization (Cont'd)

Module	Configuration	FFs	LUTs	Block RAM
Register slice	B channel, Light-weight (type 7)	5	5	
Register slice	W/R channel, Light-weight (type 7), Data width = 32 bits	40	10	
Register slice	W/R channel, Light-weight (type 7), Data width = 64 bits	80	10	
Register slice	W/R channel, Fully registered (type 1), Data width = 32 bits	80	50	
Register slice	W/R channel, Fully registered (type 1), Data width = 64 bits	150	80	
AXI4 to AXI3 converter	Data width = 32 bits	240	250	
AXI4 to AXI4-Lite converter	Data width = 32 bits	5	15	

Table 14: Spartan-6 FPGA Resource Utilization

Module	Configuration	FFs	LUTs	Block RAM
Crossbar (SAMD)	4 SI x 1 MI, Data width = 32 bits	300	740	
Crossbar (SAMD)	4 SI x 1 MI, Data width = 64 bits	390	970	
Crossbar (SAMD)	1 SI x 4 MI, Data width = 32 bits	340	400	
Shared-Access Crossbar (SASD)	1 SI x 4 MI, Data width = 32 bits	40	150	
Crossbar (SAMD)	4 SI x 4 MI, Data width = 32 bits	560	1430	
Upsizer	Data width = 32 to 64 bits	420	560	
Downsizer	Data width = 64 to 32 bits	560	630	
Sync clock converter	Data width = 32 bits	190	75	
Sync clock converter	Data width = 64 bits	330	120	
Async clock converter	Data width = 32 bits	720	330	
Async clock converter	Data width = 64 bits	870	400	
Datapath FIFO	32-deep (LUT RAM), Data width = 32 bits	110	50	
Datapath FIFO	32-deep (LUT RAM), Data width = 64 bits	180	70	
Datapath FIFO	512-deep (Block RAM), Data width = 32 bits	90	50	2
Datapath FIFO	512-deep (Block RAM), Data width = 64 bits	130	70	3
Register slice	AW/AR channel, Light-weight (type 7)	30	5	
Register slice	B channel, Light-weight (type 7)	5	5	
Register slice	W/R channel, Light-weight (type 7), Data width = 32 bits	40	10	
Register slice	W/R channel, Light-weight (type 7), Data width = 64 bits	80	10	
Register slice	W/R channel, Fully registered (type 1), Data width = 32 bits	80	50	
Register slice	W/R channel, Fully registered (type 1), Data width = 64 bits	150	80	
AXI4 to AXI3 converter	Data width = 32 bits	200	180	
AXI4 to AXI4-Lite converter	Data width = 32 bits	5	15	

## Support

Xilinx provides technical support for this LogiCORE IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

## Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx ISE Design Suite Embedded Edition software under the terms of the Xilinx [End User License](#). The core is generated using the Xilinx ISE Embedded Edition software (EDK). More information about this module is available at the [AXI Interconnect](#) page.

See the [Xilinx Intellectual Property](#) page for information on other Xilinx LogiCORE IP modules. For information on pricing and availability of other Xilinx LogiCORE modules and software, contact your local Xilinx [sales representative](#).

## References

- *ARM AMBA AXI Protocol v2.0* (document number ARM IHI 0022C)  
<http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ih0022c/index.html>

These Xilinx documents can be located from the [Xilinx Support website](#):

- *Platform Specification Format Reference Manual* (UG642)
- *Xilinx AXI Reference Guide* (UG761)
- *LogiCORE IP AXI-to-AXI Connector* (DS803)

## Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
09/21/2010	1.0	Initial Xilinx release in IDS release 12.3.
12/14/10	2.0	Xilinx release in IDS release 12.4.
03/01/11	3.0	Xilinx release in IDS release 13.1.
06/22/11	4.0	Xilinx release in IDS release 13.2.
10/19/11	4.1	<p>ISE 13.3 software release for core version 1.04.a.</p> <ul style="list-style-type: none"> <li>Removed SI slot references from the C_INTERCONNECT_DATA_WIDTH parameter.</li> <li>In <a href="#">XPS Supported Features</a> on <a href="#">page 2</a>, added bulleted item about not converting AXI4/AXI3 bursts.</li> <li>In <a href="#">CORE Generator Tool Supported Features</a>, replaced bulleted item about the number of Slave and Master interfaces supported.</li> <li>Added paragraph about the M_AXI_WID and S_AXI_WID signals to <a href="#">Use of ID Signals</a>, <a href="#">page 21</a>.</li> <li>Added note to <a href="#">AXI4-Lite Slave Conversion</a>, <a href="#">page 26</a>.</li> <li>Added S_AXI_WID and note 4 to <a href="#">Table 3</a>.</li> <li>Added note 2 to <a href="#">Table 4</a>.</li> <li>Added C_S_AXI_WRITE_FIFO_DELAY and C_S_AXI_READ_FIFO_DELAY to <a href="#">Table 7</a>.</li> <li>Added C_M_AXI_WRITE_FIFO_DELAY and C_M_AXI_READ_FIFO_DELAY to <a href="#">Table 8</a>.</li> <li>Added C_Snn_AXI_WRITE_FIFO_DELAY and C_Snn_AXI_READ_FIFO_DELAY to <a href="#">Table 10</a>.</li> <li>Added C_M00_AXI_WRITE_FIFO_DELAY and C_M00_AXI_READ_FIFO_DELAY to <a href="#">Table 11</a>.</li> <li>Revised <a href="#">Datapath FIFOs</a>, <a href="#">page 49</a>.</li> </ul>

## Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critappls>.