HB0766
Handbook
CoreAXI4Interconnect v2.8

Microsemi
a Microchip company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

Contents

Figures

Tables

1 Revision History

The revision history describes the edits done in this document. The changes are listed as follows.

1.1 Revision 8.0

Published in March 2020. The core was updated to version 2.8. Following are the edits done in this revision of the document.

- Updated Table 1, page 7, Table 2, page 7, Table 3, page 8, and Table 4, page 8
- Removed parameters MASTERx_DEF_BURST_LEN.
- Added more details in section User Testbench, page 51.

1.2 Revision 7.0

Published in September 2019. The core was updated to version 2.7.

1.3 Revision 6.0

Published in May 2019. Following are the edits done in this revision of the document.

- The core was updated to version 2.6.
- Removed parameter AHB_MASTERx_BRESP_CHECK_MODE. Added parameters SLAVEy_START_ADDR_UPPER, SLAVEy_END_ADDR_UPPER, MASTERx_READ_INTERLEAVE,SLAVEy_READ_INTERLEAVE, OPTIMIZATION. Updated valid value for ADDR_WIDTH and NUM_MASTERS parameters. For more information, see Table 8, page 35.
- Updated Device Utilization and Performance table.
- Added new section Clocking and Reset. For more information, see Clocking and Reset, page 42
- Added new section Design Constraints. For more information, see Design Constraints, page 52

1.4 Revision 5.0

Published in Apr 2018. Following are the edits done in this revision of the document.

- The core is updated to version 2.5.
- Removed parameters UPPER_COMPARE_BIT,LOWER_COMPARE_BIT,SLOTy_BASE_VEC, SLOTy_MIN_VEC and SLOTy_MAX_VEC, added parameters SLAVEy_START_ADDR and SLAVEy_END_ADDR, and updated valid value for ADDR_WIDTH parameter. For more information, see Table 8, page 35.
- Updated address decoding example. For more information, see Address Decoding, page 21.
- A new chapter, System Integration, page 53, is added.
- A new section, Smart Design, page 49 is added in Tool Flows, page 49.

1.5 Revision 4.0

Published in Nov 2017. The core was updated to version 2.4.

1.6 Revision 3.0

- Revision 3.0 was published in Jul 2017. Following are the edits done in the revision 3.0 of this document.The core was updated to v2.3.

Support to number of slaves is upgraded to 32. For more information, see Introduction, page 4.

- Description of M_CLKx and S_CLKy is updated. For more information, see Table 7, page 29.

Parameter description and valid values are updated. For more information, see Table 8, page 35.

1.7 Revision 2.0

Published in Apr 2017. The core was updated to version 2.2.

1.8 Revision 1.0

Published in Mar 2017. It was the first publication of this document. Created for version 2.0.

2 Terminology

This section describes terms and definitions used in this document.

2.1 Abbreviations

- **AMBA**: ARM Advanced Microcontroller Bus Architecture
- **AXI4**: Advanced eXtensible Interface v4
- **AXI4-Lite**: Advanced eXtensible Interface v4 Lite
- **AXI3**: Advanced eXtensible Interface v3
- **AHB-Lite**: AMBA High-performance Bus Lite
- **DWC**: Data Width Converter
- **CDC**: Clock Domain Crossings
- **FIFO**: First In First Out
- **DMA**: Direct Memory Access
- **UART**: Universal Asynchronous Receiver/Transmitter
- **DDR3**: Double Data Rate 3
- **LSRAM**: Large Synchronous Random Accessible Memory
- **SASD**: Shared Address Shared Data
- **SAMD**: Shared Address Multiple Data

2.2 Terms and Definitions

- **x**: Represents a range of 0 to 15 for master signals and 0 to 31 for slave signals unless stated otherwise.
- **y**: Represents a range of 0 to 31 for slave signals unless stated otherwise.
- **Parameters**: In section 4.2 **Core Parameters**, **Parameter Name** column shows actual parameter name used in RTL and **Description** column starts with parameter name appear in the AXI4 Interconnect Configurator (GUI interface). These two names are used interchangeably throughout the document.

3 Introduction

The AMBA AXI4 Interconnect core connects one or more AXI memory-mapped master devices to one or more memory-mapped slave devices. The AMBA AXI protocol supports high-performance, high-frequency system designs. CoreAXI4Interconnect is a configurable core with the following features:

- Supports high-bandwidth and low-latency designs.
- Meets the interface requirements of a wide-range of components.
- Suitable for memory controllers with high-initial access latency.
- Provides flexibility in the implementation of interconnect architectures.

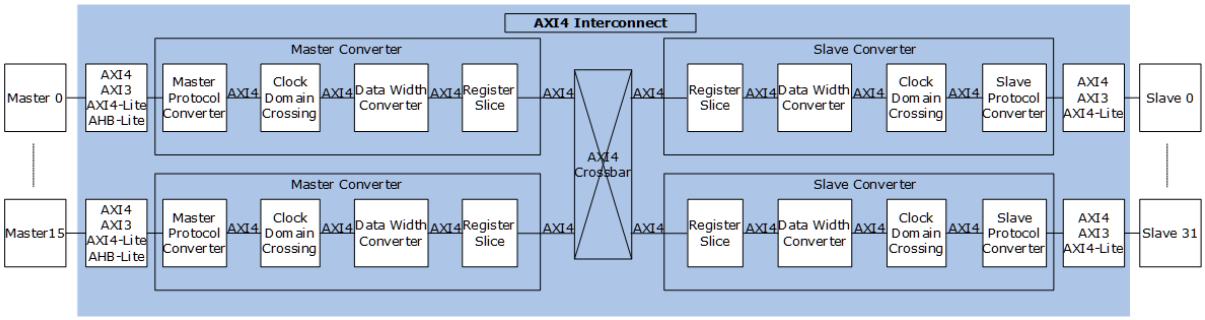
The key features of the AXI protocol are:

- Separate address/control and data phases.
- Supports unaligned data transfers, using byte strobes.
- Uses burst-based transactions with only the start address issued.
- Separate read and write data channels, which provides low-cost Direct Memory Access (DMA).
- Supports multiple outstanding read/write transactions.
- Supports out-of-order read transaction completion.
- Permits easy addition of register stages to provide timing closure.

The following figure shows the top-most AXI4 Interconnect core block diagram. Inside the AXI Interconnect core, theAXI4 Crossbar core routes the traffic between the Slave Ports (SP) and Master Ports (MP). The convention followed here is that the master-side ports connect to external upstream masters, while the slave-side ports connect to external downstream slaves. Along each pathway connecting the master ports or the slave ports to the Crossbar, an optional series of AXI Infrastructure cores perform various conversion and buffering functions. The Infrastructure cores include **Register Slice, Data Width Converter, Protocol Converter,** and **Clock-Domain Crossing Converter.** The Protocol Converter includes AHB-Lite and AXI options, when placed at Master Ports.

The AXI Interconnect core can be configured to have up to 32 slave-side ports and up to 16 master ports. The crossbar core, at the center, routes traffic on all the AXI channels between the slave-side ports and master ports. Along each of the pathways between slave-side ports and the crossbar, or between the crossbar and master ports. There are one or more infrastructure cores that perform various conversion and storage functions. The crossbar effectively splits the AXI Interconnect core down the middle between the master ports-related functional units and the slave-side ports-related units.

Figure 1 • **CoreAXI4Interconnect**



3.1 AXI4 Infrastructure Cores

The following components are included within each instance of the AXI4 Interconnect core, depending on the configuration of AXI4 Interconnect core:

1. **AXI4 Crossbar** connects one or more similar AXI4 memory-mapped masters to one or more similar memory-mapped slaves.
2. **AXI4 Data-Width Converter** connects one AXI4 memory-mapped interface to one AXI4 interface memory-mapped having a wider or narrower datapath.
3. **Master Protocol Converter** converts an AXI3, AXI4-Lite, or AHB-Lite master to an AXI4 master.
4. **Slave Protocol Converter** converts an AXI4 slave to an AXI3 or AXI4-Lite slave.
5. **AXI4 Register Slice** connects one AXI4 memory-mapped master to one AXI4 memory-mapped slave through a set of pipeline registers, to break a critical timing path.
6. **Clock Domain Crossing (CDC)** components provide AXI4 to AXI4 clock domain crossing functions, which can be enabled on a per-port basis.

The internal design of the CoreAXI4Interconnect is AXI4, only at the ingress and egress is AXI3, AXI4Lite and AHB-Lite (master ports only) handles – where, they are converted to AXI4 format. The crossbar sets the native data width for the core. An individual port can defined to have a different data width, which defines the width for that ports Protocol Converter and Data Width Converter modules.

3.1.1 Key Features

The following list describes the key features of CoreAXI4Interconnect:

- AXI protocol compliant The AXI4 Interconnect core can be configured to support AXI4, AXI3, and AXI4-Lite protocols on all master or slave ports, and the AHB-Lite protocol on master ports.
- The AXI4 Interconnect core breaks-up, burst transactions of more than 16 data beats from AXI4 masters into multiple transactions of no more than 16 beats when addressed to an AXI3 slave.
- The AXI4 Interconnect core breaks-up burst transactions of more than 1 data beats from AXI4 or AXI3 masters into multiple transactions of 1 beats when addressed to an AXI4Lite slave.
- Interface data widths:
  - AXI4/AXI3/AHB-Lite: 32, 64, 128, 256, or 512 bits
  - AXI4-Lite: 32 or 64 bits
- Address width: Up to 64 bits
- USER width (per channel): Up to 64 bits
- ID width: Up to 8 bits
- Supports Read-only and Write-only masters and slaves, to reduce resource utilization.
- Supports up to 16 masters and 32 slaves
- Supports AXI3/AXI4 read interleaving.

3.1.2 Limitations

The following list gives the limitations of CoreAXI4Interconnect in this version:

- The AXI4 Interconnect core doesn't support AXI4 streaming interface.
- The AXI4 Interconnect core doesn't support Trust Zone security.
- The AXI4 Interconnect core doesn't support Region for slave devices with multiple address decode ranges.
- AXI4 QoS signals do not effect arbitration priority in crossbar.
- The AXI4 Interconnect core doesn't support low-power mode or propagate the AXI C channel signals.
- The AXI4 Interconnect core doesn't support time out in case of response is not received from the destination.
- The AXI4 Interconnect core doesn't support AXI3 write interleaving.

3.1.3 Core Version

This handbook supports CoreAXI4Interconnect version 2.8.

3.1.4 Supported Families

The following list of families support CoreAXI4Interconnect v2.8:

- PolarFire SoC
- PolarFire®
- SmartFusion®2
- IGLOO®2
- RTG4™

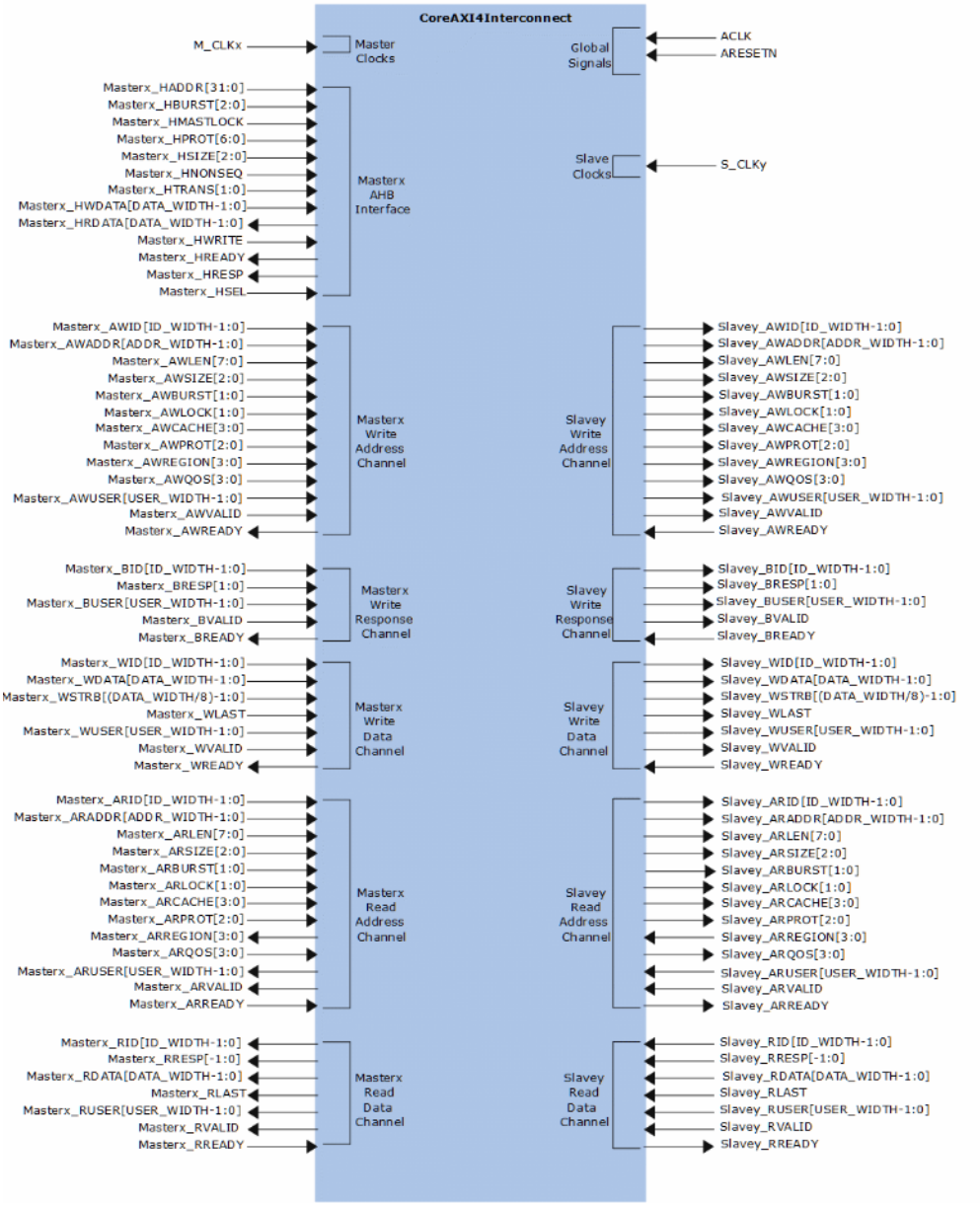
3.1.5 Supported Interfaces

CoreAXI4Interconnect is available with the following interfaces:

- AXI3, AXI4, AXI4-Lite, or AHB-Lite Masters
- AXI3, AXI4, or AXI4-Lite Slaves

The following figure shows detailed block diagram of CoreAxi4interconnect system:

Figure 2 • **Core AXI4 Interconnect System**



3.1.6 Device Utilization and Performance

The following tables summarize various parameters settings data. The PolarFire MPF300TS_ES-1FCG484E device and the SmartFusion2 M2S150TS-1FC1152 device are used in deriving the following data. The PolarFire SoC device data is the same as the PolarFire device, and the IGLOO2 device data is the same as the SmartFusion2 device.

Table 1 • **CoreAXI4Interconnect Device Utilization and Performance (8 × 8, Crossbar Data Width, 32-Bit)**

Family	Device	Logic Elements		Performance	RAM
		Sequential	Combinational		
SmartFusion2	M2S150TS-1FC1152	8201	12665	122MHz – ACLK (Crossbar Clock) 145MHz – S_CLK (Slave Clock)	63 Blocks of uSRAM
PolarFire	MPF300TS_ES-1FCG484E	6141	9892	156MHz – ACLK (Crossbar Clock) 293MHz – S_CLK (Slave Clock)	78 Blocks of uSRAM
RTG4	RT4G150L-FCG1657M	8347	11776	72 MHz – ACLK (Crossbar Clock) 68 MHz – S_CLK (Slave Clock)	63 Blocks of uSRAM

Note: NUM_MASTERS = 8, NUM_SLAVES= 8, DATA_WIDTH = 32, ADDR_WIDTH = 32, ID_WIDTH = 4, OPEN_TRANS_MAX=2, CROSSBAR_MODE=SAMD, SLAVEy_RS-1, MASTER_RSx=1, RD_ARB_EN=0, MASTERx_CLOCK_DOMAIN_CROSSING=0, SLAVEy_CLOCK_DOMAIN_CROSSING=1.

Table 2 • **CoreAXI4Interconnect Device Utilization and Performance (8 × 8, Crossbar Data Width, 64-Bit)**

Family	Device	Logic Elements		Performance	RAM
		Sequential	Combinational		
SmartFusion2	M2S150TS-1FC1152	9500	14100	121MHz – ACLK (Crossbar Clock) 150MHz – S_CLK (Slave Clock)	70 Blocks of uSRAM
PolarFire	MPF300TS_ES-1FCG484E	7339	11371	170MHz – ACLK (Crossbar Clock) 281MHz – S_CLK (Slave Clock)	92 Blocks of uSRAM
RTG4	RT4G150L-FCG1657M	9592	13812	61 MHz – ACLK (Crossbar Clock) 65 MHz – S_CLK (Slave Clock)	70 Blocks of uSRAM

Note: NUM_MASTERS = 8, NUM_SLAVES= 8, DATA_WIDTH = 64, ADDR_WIDTH = 32, ID_WIDTH = 4, OPEN_TRANS_MAX=2, CROSSBAR_MODE=SAMD, SLAVEy_RS-1, MASTER_RSx=1,

RD_ARB_EN=0, MASTERx_CLOCK_DOMAIN_CROSSING=0, SLAVEy_CLOCK_DOMAIN_CROSSING=1

Table 3 • **CoreAXI4Interconnect Device Utilization and Performance (4 ×4, 64-Bit Crossbar Data Width, Data Width Converters and Clock Domain Crossings)**

Family	Device	Logic Elements		Performance	RAM
		Sequential	Combinational		
SmartFusion2	M2S150TS-1FC1152	10938	14755	125 MHz – ACLK (Crossbar Clock) 190 MHz – S_CLK (Slave Clock) 169 MHz – M_CLK (Master Clock)	117 Blocks of uSRAM
PolarFire	MPF300TS_ES-1FCG484E	9042	13124	203 MHz – ACLK (Crossbar Clock) 336 MHz – S_CLK (Slave Clock) 219 MHz – M_CLK (Master Clock)	158 Blocks of uSRAM
RTG4	RT4G150L-FCG1657M	9592	13812	54 MHz – ACLK (Crossbar Clock) 69 MHz – S_CLK (Slave Clock) 44 MHz – M_CLK (Master Clock)	112 Blocks of uSRAM

Note: NUM_MASTERS = 4, NUM_SLAVES= 4, DATA_WIDTH = 64, MASTER0_DATA_WIDTH=32, MASTER1_DATA_WIDTH=128 (all other master's DATA_WIDTH= 64) ADDR_WIDTH = 32, SLAVE0_DATA_WIDTH=32, SLAVE1_DATA_WIDTH=128 (all other slave's DATA_WIDTH= 64), ID_WIDTH = 4, OPEN_TRANS_MAX=2, CROSSBAR_MODE=SAMD, SLAVEy_RS-1, MASTER_RSx=1, RD_ARB_EN=0, MASTERx_CLOCK_DOMAIN_CROSSING=1, SLAVEy_CLOCK_DOMAIN_CROSSING=1

Table 4 • **CoreAXI4Interconnect Device Utilization and Performance (4 × 4, 64-Bit Crossbar Data Width, Data Width Converters, Clock Domain Crossings and Read Interleaving)**

Family	Device	Logic Elements		Performance	RAM
		Sequential	Combinational		
PolarFire	MPF300TS_ES-1FCG484E	19242	30713	183 MHz – ACLK (Crossbar Clock) 274 MHz – S_CLK (Slave Clock) 231 MHz – M_CLK (Master Clock)	489 Blocks of uSRAM

Note: NUM_MASTERS = 4, NUM_SLAVES = 4, DATA_WIDTH = 64, MASTER0_DATA_WIDTH = 32, MASTER1_DATA_WIDTH = 128 (all other master's DATA_WIDTH = 64) ADDR_WIDTH = 32, SLAVE0_DATA_WIDTH = 32, SLAVE1_DATA_WIDTH = 128 (all other slave's DATA_WIDTH = 64), ID_WIDTH = 2, OPEN_TRANS_MAX = 2, CROSSBAR_MODE = SAMD, SLAVEy_RS-1, MASTER_RSx = 1, RD_ARB_EN = 0, NUM_THREADS = 4, MASTERx_CLOCK_DOMAIN_CROSSING = 1, SLAVEy_CLOCK_DOMAIN_CROSSING = 1, MASTERx_READ_INTERLEAVE = 1, SLAVEy_READ_INTERLEAVE = 1

4 Functional Descriptions

The CoreAXI4Interconnect soft IP provides a user interface for the AXI4, AXI3, AXI4-Lite, or AHB- Lite master interfaces to AXI4, AXI3, or AXI4-Lite slave interfaces.

4.1 AXI4 Crossbar

Each instance of the AXI4 Interconnect core contains one AXI4 Crossbar instance, if it is configured with more than one SIave and one Master. The slaveports of the AXI4 Crossbar core can be configured to comprise 1 to 32 slaveports slots, to accept transactions from up to 16 connected master ports. The master ports can be configured to comprise 1 to 16 master ports slots to issue transactions up to 32 connected slave devices.

The AXI4 Cross bar key features are as follows:

- Selectable Crossbar Architecture – Shared Address Multiple Data (SAMD) or Shared Address Shared Data (SASD)
- Shared Address Multiple Data mode (Performance optimized)
    - Shared-Address (one for Write addresses and one for Read addresses), Multiple-Data (SAMD) crossbar architecture.
    - Parallel crossbar pathways for Write data and Read data channels. This enables independent and concurrent Write data transactions up to 16 write data paths, as well as independent and concurrent Read data transactions up to 16 read data paths. This allows multiple data transactions in parallel, AXI4 provided ordering rules are met, which the crossbar imposes.
    - It allows programmable number of outstanding transactions at a time.
    - Sparse crossbar data paths according to configured connectivity map, resulting in reduced resource utilization.
    - One shared Write address bus, plus one shared Read address bus.
    - One shared Response bus.
    - Arbitration latencies typically do not impact to data throughput, when transactions average is at least two data beats.
    - Crossbar always operates with AXI4 Interface.
- Shared Address Shared Data mode (Area optimized)
    - Shared write data, shared read data, shared write address, and shared read-address buses (SASD).
    - It has one shared response bus.
    - It minimizes resource utilization.
- Supports connected masters with multiple reordering depths (ID threads).
- Supports up to 8-bit wide ID signals with varying ID width per connected master.
- Supports to configure maximum outstanding read/write transactions.
- 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.
- Supports "Single Slave per ID" method to avoid cyclic dependency (deadlock). For each ID thread, issued by a connected master, the Crossbar allows one or more outstanding transactions to only one slave device for Writes and one slave device for Reads, at a time.
- Round-robin arbitration
    - Round-robin arbitration is used among all connected masters.
    - Any slave ports slot which reached its outstanding limit, or is targeting an master ports slot which reached its issuing limit, or is trying to access master ports slot in a manner that risks deadlock, is temporarily removed from arbitration, so that, the other slave ports slots can be granted.

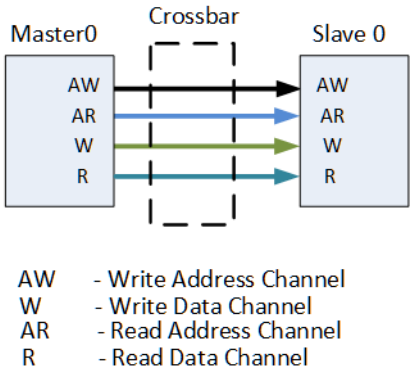
The AXI4Interconnect core can be configured in one of several modes as shown in the following sections:

4.1.1 Pass-Through

In this case, only one master needs to connect to one slave, the AXI4Interconnect can be configured in 1-to-1 or pass-through mode. In this case, the only function of the core is to convert, such as, connecting an AXI4 Master to and AXI4/3 Slave. The crossbar is bypassed, which maximizes the performance and minimizes the resource utilization.

Note: If read interleaving for any master or slave is enabled, then the crossbar will not be bypassed. User should configure **Mx Read Interleave** and **Sy Read Interleave** parameters to zero to minimize the resource utilization.

Figure 3 • **Pass-Through**



To configure the AXI4 Interconnect core in pass through mode, parameters configuration should be as follows:

- Number of Masters - 1
- Number of Slaves - 1
- M0 Read Interleaving - Disabled
- S0 Read Interleaving - Disabled

Other parameters can be configured as per the requirement. Figure 4, Figure 5, and Figure 6 illustrates the pass through mode configuration.

Figure 4 • **Pass-through Mode Configuration**

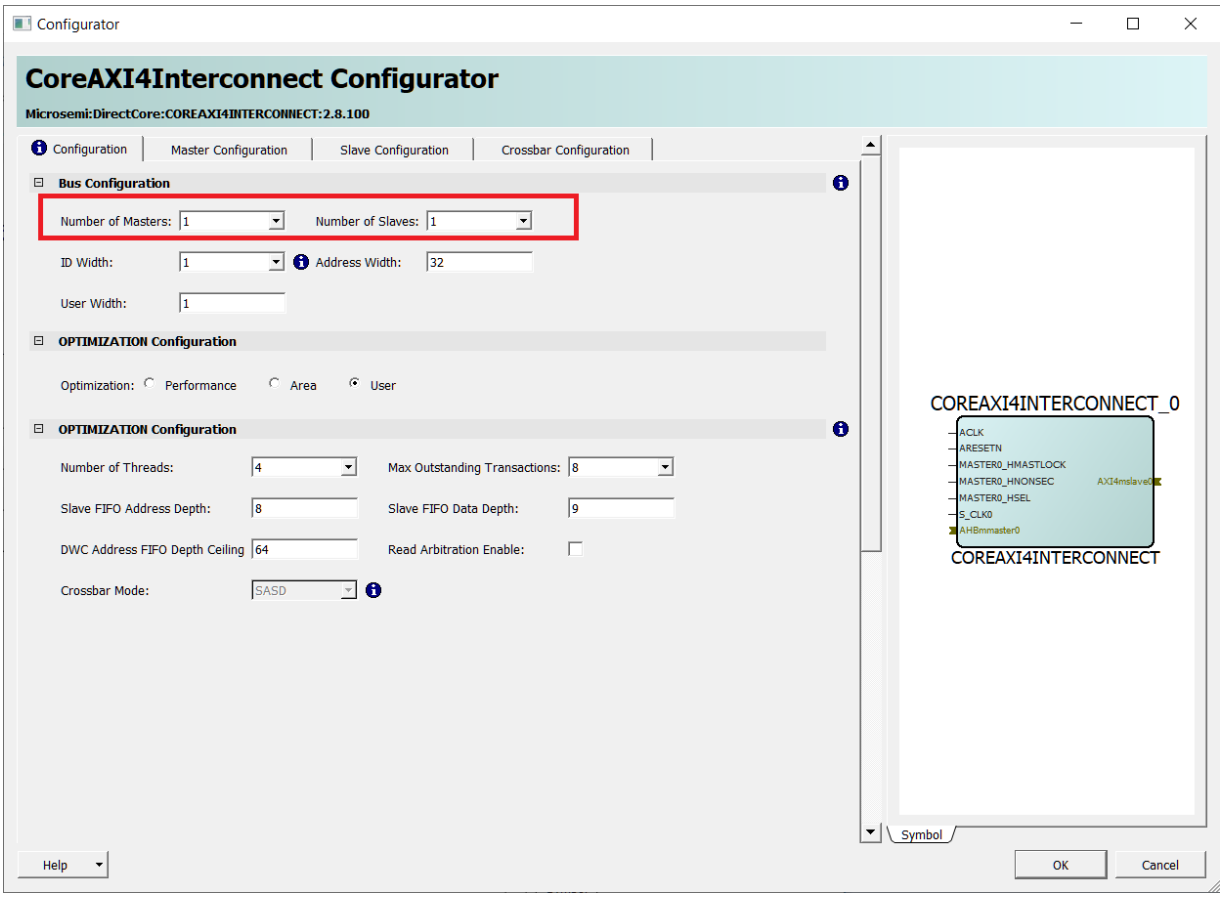


Figure 5 • **Pass-through Mode Configuration continue..**

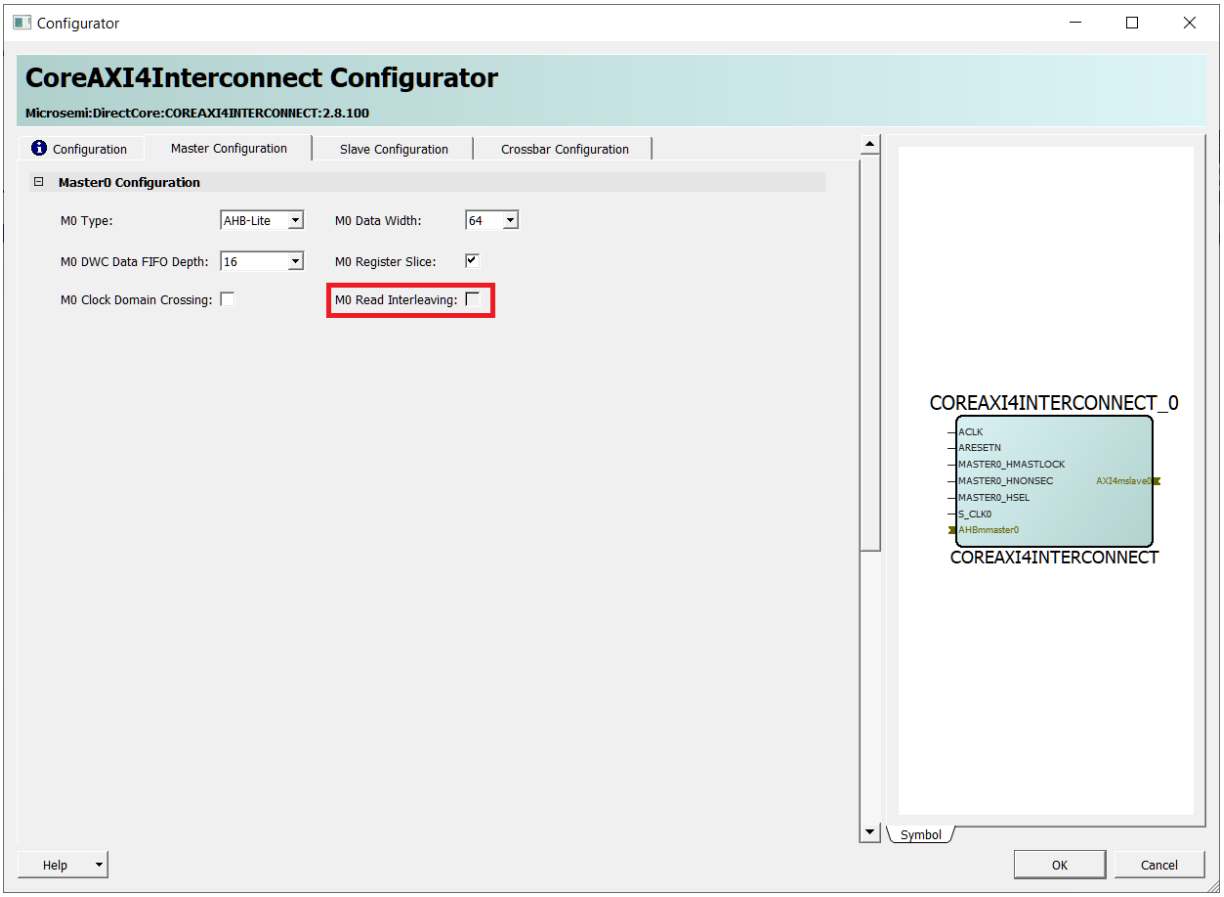
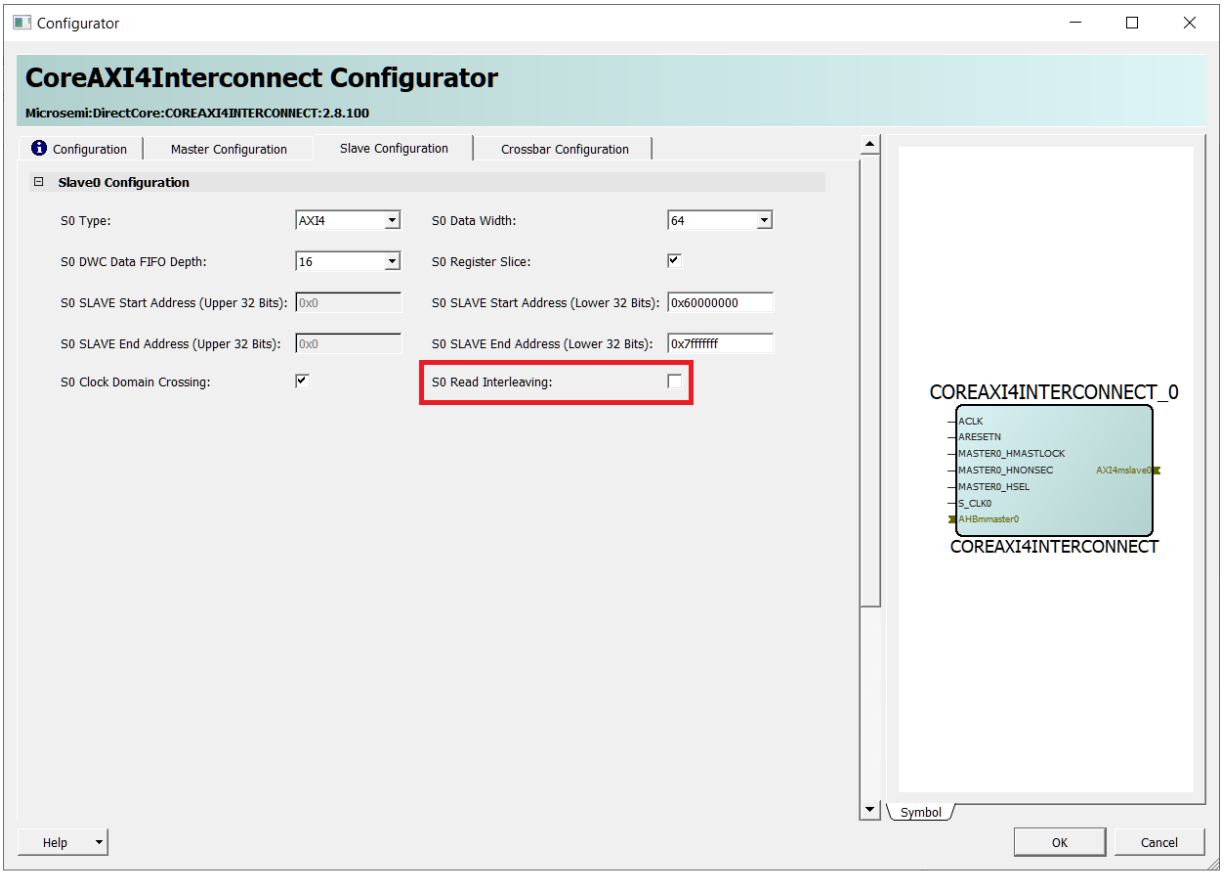


Figure 6 • **Pass-through Mode Configuration continue...**



4.1.2 N-to-1 Interconnect or 1-to-M Interconnect

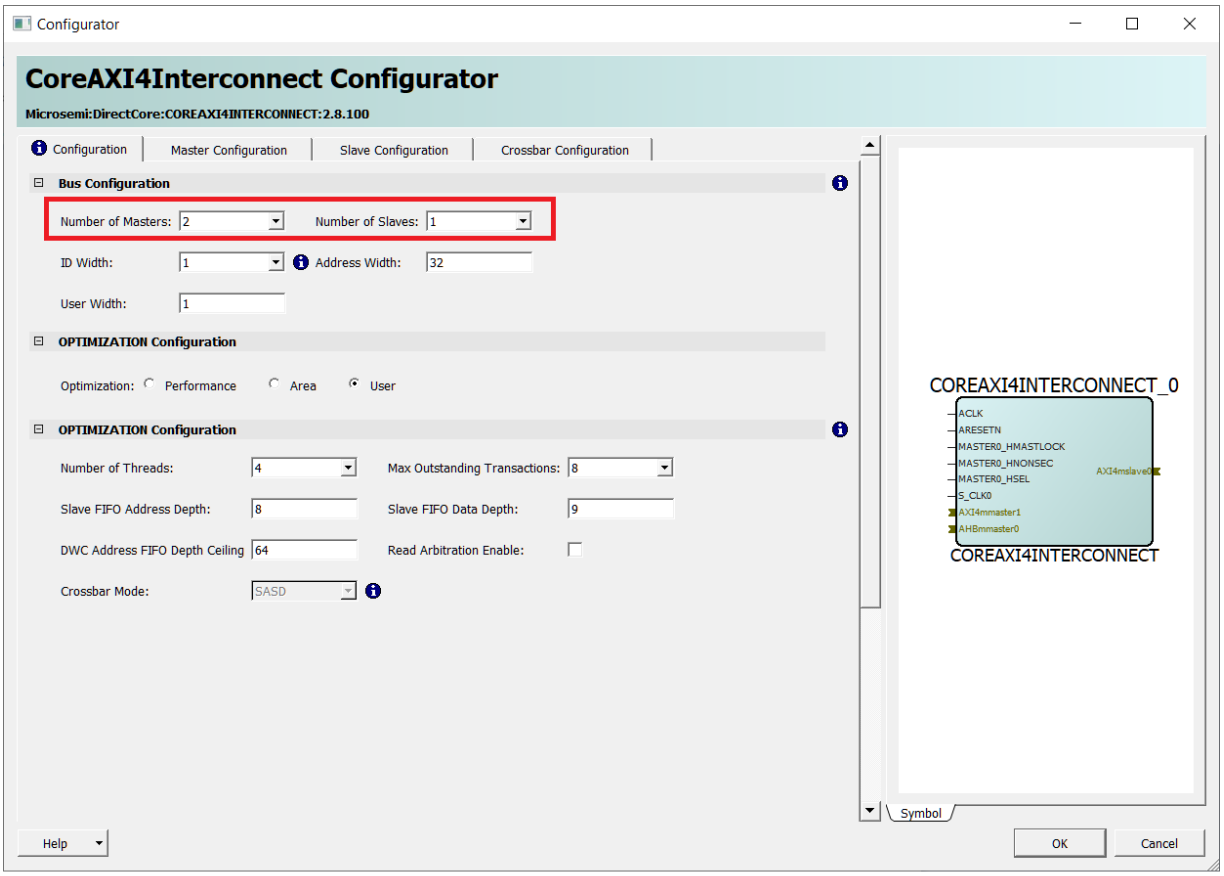
A common configuration for the AXI Interconnect core is when multiple master devices arbitrate for access to a single slave device, often a memory controller. In these cases, the address decoding logic might be unnecessary and omitted from the AXI Interconnect core (unless address range validation is needed). Pipeline and protocol conversions to or from AXI3 and AXI4Lite are useful. Another de-generative configuration of the AXI Interconnect core is when a single master device, typically a processor, accesses multiple memory-mapped slave peripherals. In these cases (N-to-1 and 1-to-M), all use the AXI4Interconnnect arbitration, address decode, and data pathways as normal.

To configure the AXI4 Interconnect core in N-to-1 mode, parameters configurations should be as follows:

- Number of Masters - Greater than 1
- Number of Slaves - 1.

Figure 7 illustrates the N-to-1 configuration.

Figure 7 • **N-to-1 Mode Configuration**

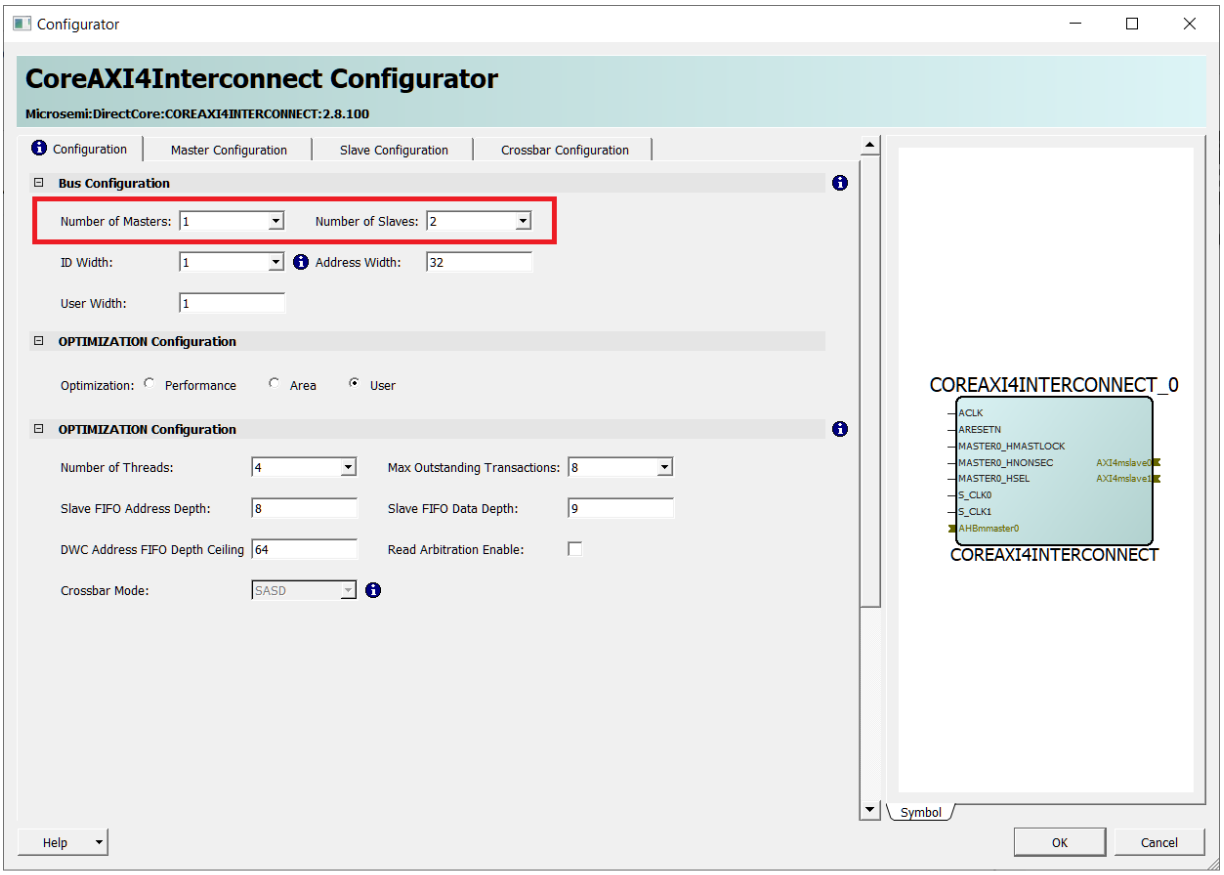


To configure the AXI4 Interconnect core in 1-to-M mode, parameters configurations should be as follows:

- Number of Masters - 1
- Number of Slaves - Greater than 1

Figure 8 illustrates the 1-to-M configuration.

Figure 8 • **1-to-M Mode Configuration**

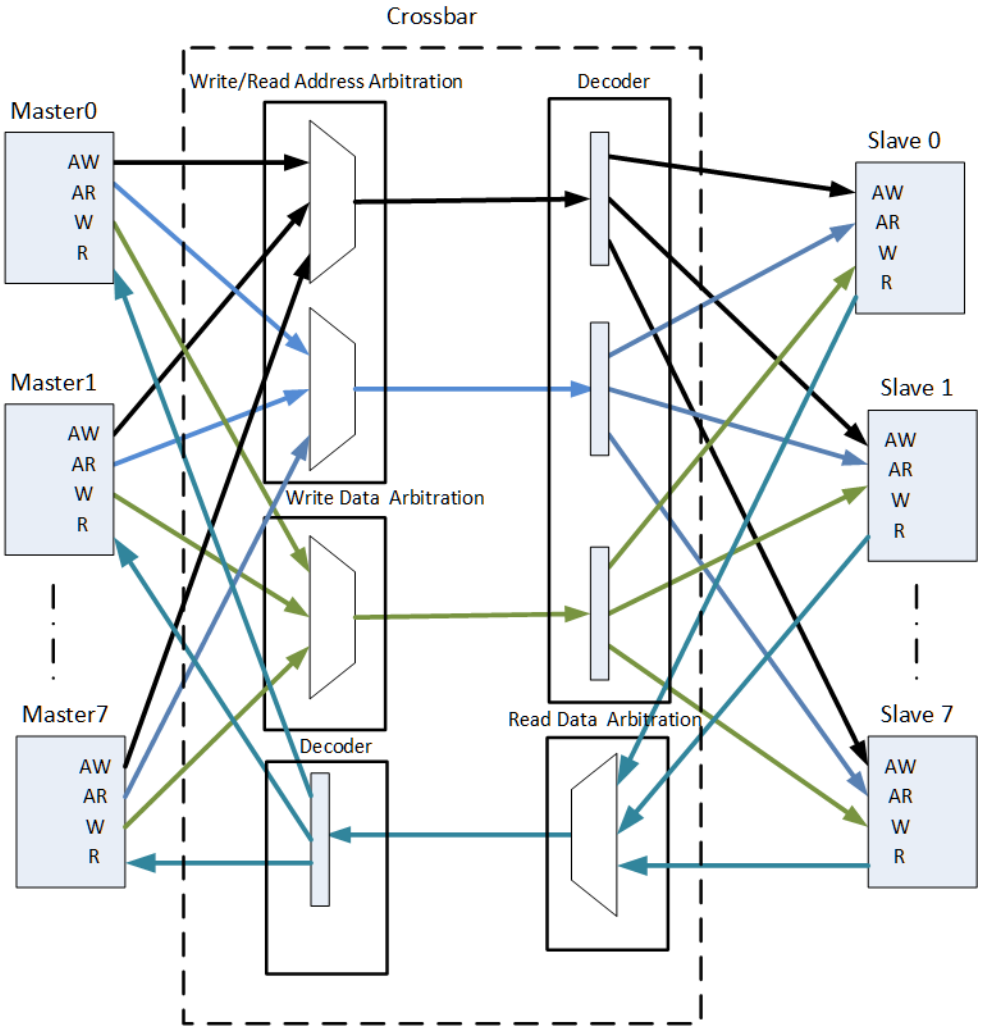


In both N-to-1 and 1-to-M mode, other parameters can be configured as per the requirement.

4.1.3 N-to-M Interconnect - Shared Address Shared Data Mode

For SASD mode, the N-to-M use case of the AXI Interconnect core supports for only one outstanding read transaction and one outstanding write transaction at a time. The arbiter selects from among the requesting masters. One write data transfer is enabled to the targeted slave device and after the data transfer (including the write response) completes, the next write request is arbitrated. Similarly, one read transaction is selected in parallel to a write cycle and performed at the same time. SASD mode minimizes the resources used to implement the crossbar module of the AXI4Interconnect.

Figure 9 • **N-to-M Interconnect - SASD Mode showing 8 × 8 Example**



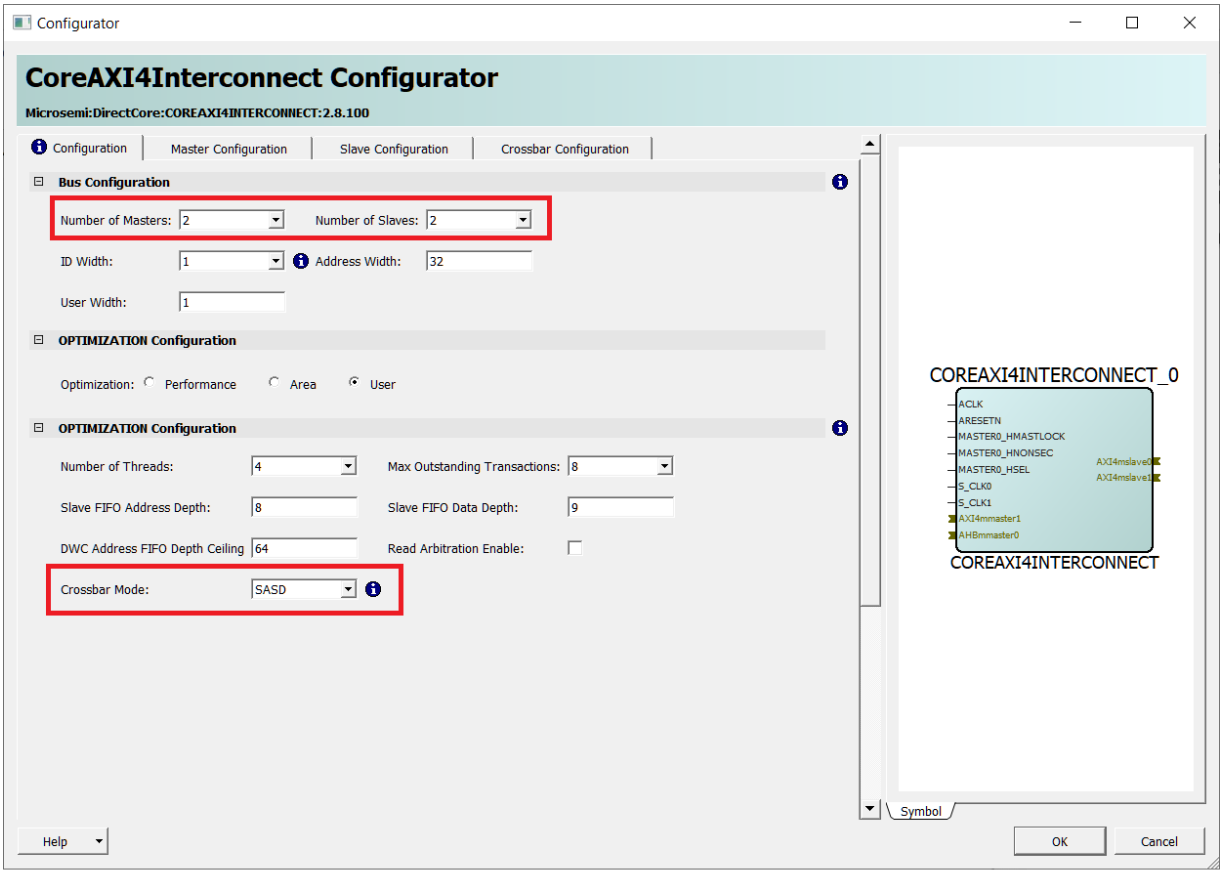
AW - Write Address Channel
W - Write Data Channel
AR - Read Address Channel
R - Read Data Channel

To configure the AXI4 Interconnect core in N-to-M SASD mode, parameters configurations should be as follows:

- Number of Masters - Greater than 1
- Number of Slaves - Greater than 1
- Crossbar Mode - SASD
- Optimization - Either Area or User

Figure 10 illustrates the N-to-M SASD mode configuration.

Figure 10 • **N-to-M SASD Mode Configuration**

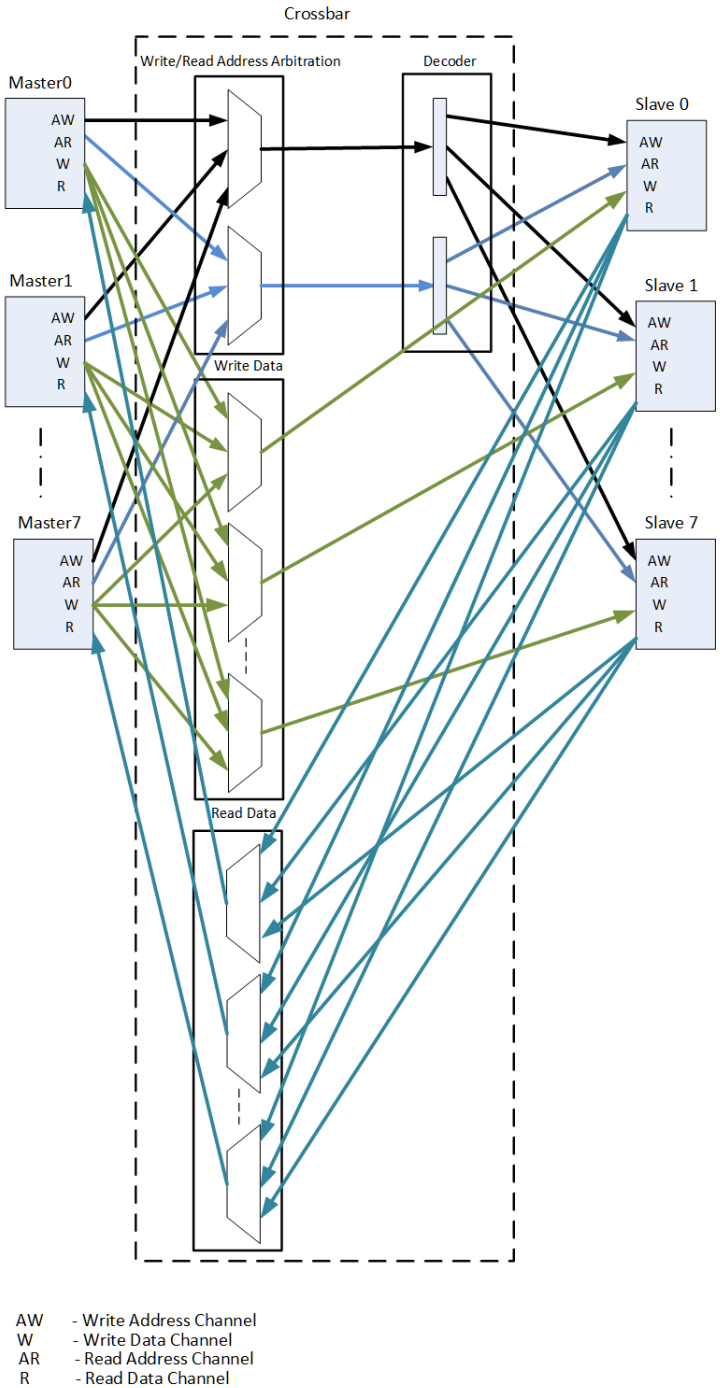


Other parameters can be configured as per the requirement.

4.1.4 N-to-M Interconnect - Shared Address Multiple Data Mode

For SAMD mode, the N-to-M use case of the AXI Interconnect core provides for parallel data pathways, to allow multiple writes and read transactions to be executed at the same time. There are up to thirty-two separate pathways for writes - one for each slave, and up to eight separate pathways for reads one for each master. There is one write address pathway, one read address pathway, and one response pathway that are shared by all masters and slaves. The sharing of the address and responses pathways does not normally affect performance as long as data bursts are longer than two. All these pathways - write address, read address, write data, read data, and response all operate in parallel.

Figure 11 • **N-to-M Interconnect - SAMD Mode Showing 8 × 8 example**

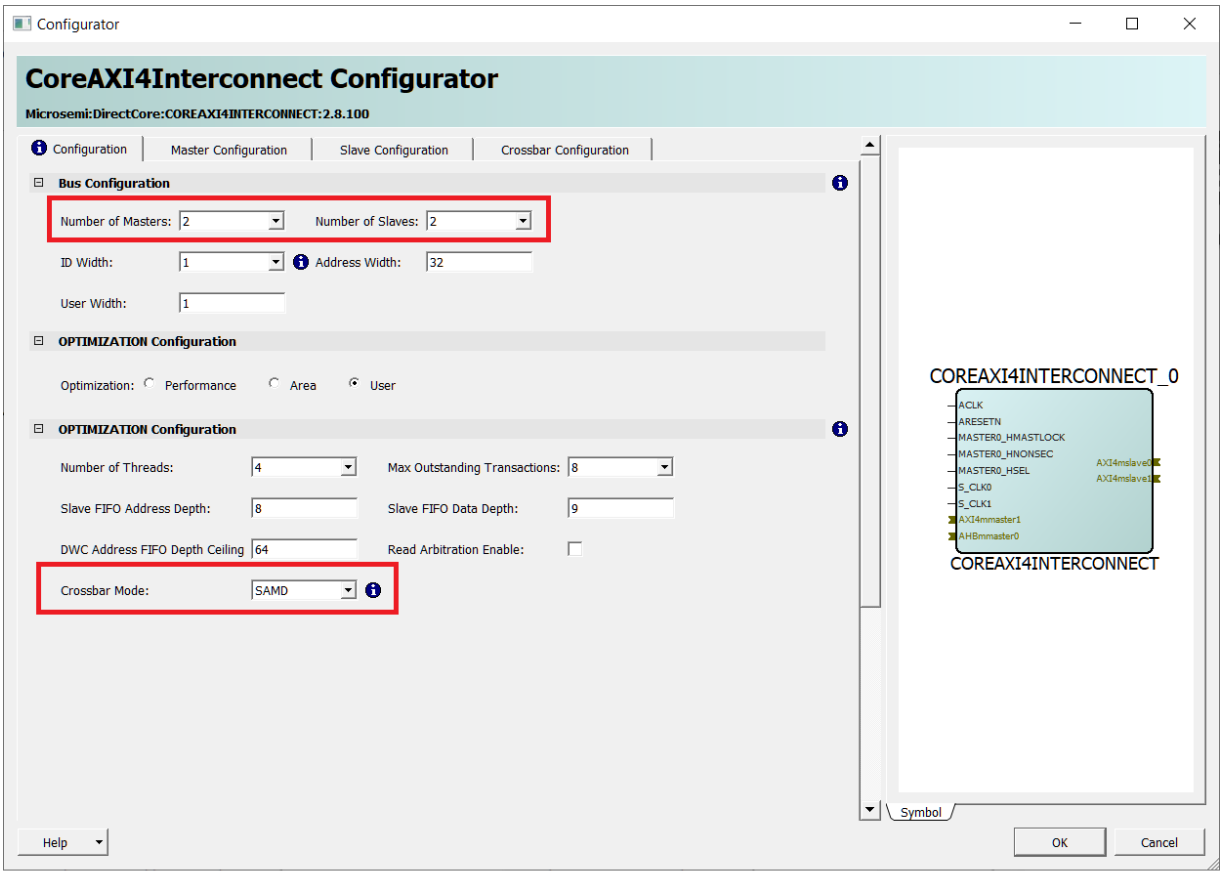


To configure the AXI4 Interconnect core in N-to-M SAMD mode, parameters configurations should be as follows:

- Number of Masters - Greater than 1
- Number of Slaves - Greater than 1
- Crossbar Mode - SAMD
- Optimization - Either Performance or User

Figure 12 illustrates the N-to-M SAMD mode configuration.

Figure 12 • **N-to-M SAMD Mode Configuration**



Other parameters can be configured as per the requirement.

4.1.5 AXI4Crossbar Limitations

The following list gives limitations of AXI4Crossbar:

- AXI4 QoS signals do not influence crossbar arbitration priority. QoS signals are only propagated from Masters to Slaves.
- The AXI4 Crossbar does not time out, if the destination of any AXI channel transfer stalls indefinitely. All slaves must respond to all received transactions, as required by AXI protocol.
- AXI4 Crossbar does not check, if burst crosses any illegal boundary. The master is required to handle it.

4.2 Data-Width Converter

The Data-Width Converter function depends on whether the data path width gets wider ("up-sizing") or narrower ("down-sizing") when moving in the direction from the Master port to Slave port. The data width conversion functions are the same in either the Master or Slave side. At the master side, up-sizing or data width up converter is implemented when master data width (Mx Data Width) is greater than AXI Interconnect core native width (Crossbar Data Width) and down-sizing or data width down converter is implemented when master data width (Mx Data Width) is less than AXI Interconnect core native width (Crossbar Data Width). Similarly, at the slave side, up-sizing or data width up converter is implemented when AXI Interconnect core native width (Crossbar Data Width) is less than slave data width (Sy Data Width) and down-sizing or data width down converter is implemented when AXI Interconnect core native width (Crossbar Data Width) is greater than slave data width (Sy Data Width).

- Master, Slave, and Crossbar can be any legal AHB-Lite or AXI data width from 32 to 512 bits. Master data width can be configured using **Mx Data Width** parameter, Slave data width can be configured using **Sy Data Width** and Crossbar data width can be configured using **Crossbar Data Width** parameter
- For upsizing, write data is packed (merged) where it is feasible to the full bus width, for efficient external memory access.
- For upsizing, both read and write data is buffered for efficient use of burst-type memory, such as DRAM and allows bursts of full data width accesses, when bus arbitration is achieved. Synchronous FIFO having different input and output widths are used to buffer read and write data during up-sizing. Synchronous FIFO depth can be configured using **Mx DWC Data Fifo Depth** parameter for the master and **Sy DWC Data Fifo Depth** parameter for the slave. Input and output data width of Synchronous FIFO derived internally in the AXI4 Interconnect core.
- For downsizing, the burst transactions are split into multiple transactions, if the maximum burst length is exceeded.
- For up sizing and downsizing, both read and write transactions control information such as size, burst type, length, id, address are buffered and write response also buffered using Synchronous FIFO. Synchronous FIFO depth can be configured for all the masters and all the slaves using **DWC Address FIFO Depth Ceiling** parameter.

Note: When read interleaving is enabled, multiple instance of Synchronous FIFO will be implemented to buffer out of order read data and read transactions, which may increase the resource utilization. Read interleaving can be configured using **Mx Read Interleaving** parameter for the master and **Sy Read Interleaving** parameter for the slave

4.3 Master Protocol Converter

The Master Protocol Converter transforms connected Masters from AXI3, AXI4Lite, or AHB-Lite to AXI4, which is the native protocol type of the AXI4Interconnect. AXI4 compliant masters do not require Master Protocol conversion. The parameter MASTERx_TYPE (Mx Type) defines, one per connected Master port, the type of Master that connects to the AXI4Interconnect. x represents the per port version.

For example: MASTER0_TYPE.

The following table shows the mapping for the MASTERx_TYPE parameter:

Table 5 • **MASTER TYPE Mapping**

MASTERx_TYPE [1:0]	Port Type
00	AXI4
01	AXI4Lite
10	AHB-Lite
11	AXI3

4.4 Slave Protocol Converter

The Slave Protocol Converter transforms the native AXI4 ports for connected slaves to AXI3 or AXI4-Lite. AXI4 compliant slaves do not require Slave Protocol conversion. The parameter SLAVEy_TYPE (Sy Type) defines, one per connected Slave port, the type of slave, which connects to the AXI4Interconnect. y represents the per port version.

For example: SLAVE0_TYPE

The following table shows the mapping for the SLAVEy_TYPE (Sy Type) parameter:

Table 6 • **SLAVE TYPE mapping**

SLAVEy_TYPE [1:0]	Port Type
00	AXI4
01	AXI4Lite
10	(reserved)
11	AXI3

The key mechanisms for AXI4-Lite protocol conversion are:

- The AXI Interconnect stores transaction IDs from Masters and retrieve them during response transfers.
- The AXI4 Interconnect core breaks-up burst transactions of more than 1 data beat from AXI4 or AXI3 masters, into multiple transactions of 1 beats.
- INCR, FIXED, and WRAP burst types are supported. The key mechanisms for AXI3 protocol conversions are:
- The AXI Interconnect core converts burst transactions of more than 16 data beats from AXI4 masters into multiple transactions of 16 beats.
- AXI3 write data interleaving is not supported. This feature was dropped by AXI4 protocol.
- Atomic locked transactions are not supported.

Note: When AXI4 burst transactions are breaks up in more than 1 data beat for AXI4 Lite slave and more than 16 data beat for AXI3 slave, multiple burst transactions and data are buffered using Synchronous FIFO. Depth of Synchronous FIFO used to buffer burst transactions can be configured using "**Slave FIFO Address Depth**" parameter and depth of Synchronous FIFO used to buffer data can be configured using "**Slave FIFO Data Depth**" parameter. These two parameters are common for all the AXI4-Lite or AXI3 slave.

4.5 Address Decoding

The AXI4Interconnect decodes the ARADDR and AWADDR in the AXI4Crossbar module to select, which SLAVE port, the MASTER wants to access. This decode is performed using the following equation. MATCHy SLAVE = (Addr[ADDR_WIDTH-1:0] >= SLAVEy_START_ADDR) and (Addr[ADDR_WIDTH-1:0] <= SLAVEy_END_ADDR)

Where:

- y is 0 to (NUM_SLAVES-1).
- MATCHy SLAVE is the target slave for the transaction.
- Addr is the ARADDR or AWADDR for the transaction.
- SLAVEy_START_ADDR is per slave parameter that defines start address for the associate SLAVEy port. SLAVEy_START_ADDR is the concatenation of SLAVEy_START_ADDR_UPPER (upper 32 bit of AWADDR or ARADDR)and SLAVEy_START_ADDR (lower 32 bit of AWADDR or ARADDR) when ADDR_WIDTH is more than 32.For ADDR_WIDTH less than 32, SLAVEy_START_ADDR_UPPER will be ignored.
- SLAVEy_END_ADDR is per slave parameter that defines end address for the associate SLAVEy port. SLAVEy_END_ADDR is the concatenation of SLAVEy_END_ADDR_UPPER (upper 32 bit of AWADDR or ARADDR)and SLAVEy_END_ADDR (lower 32 bit of AWADDR or ARADDR) when ADDR_WIDTH is more than 32. For ADDR_WIDTH less than 32, SLAVEy_END_ADDR_UPPER will be ignored.

SLAVEy_END_ADDR can be evaluated using the equation, SLAVEy_END_ADDR = SLAVEy_START_ADDR + slavey_range - 1

Note: slavey_range defines maximum address range of associate SLAVEy port. Maximum address range depends on the number of AXI address bits [maximum of AWADDR or ARADDR bits] of associate SLAVEy. The slave having 16 bits AXI address has maximum address range - 2^16 (64K). slavey_range is not a parameter.

4.5.1 Address Decode Example

For this example, the cores parameters are defined as follows:

- ADDR_WIDTH = 32;
- NUM_SLAVES = 4;

Address ranges of four slaves (not a part of cores parameters) are as follows:

- slave0_range = 256; //maximum address range of slave0
- slave1_range = 524288(512K); //maximum address range of slave1
- slave2_range = 16384 (16K); //maximum address range of slave2
- slave3_range = 512; //maximum address range of slave3

There are different possible combinations for addressing. The following list shows some of the valid combinations:

Incremental Addressing

- SLAVE0_START_ADDR = 0x0000_0000
- SLAVE0_END_ADDR = SLAVE0_START_ADDR + slave0_range - 1
  = 0x0000_0000 + 0x0000_0100 - 1
  = 0x0000_00FF
- SLAVE1_START_ADDR = SLAVE0_END_ADDR + 1
  = 0x0000_00FF + 1
  = 0x0000_0100
- SLAVE1_END_ADDR = SLAVE1_START_ADDR + slave1_range - 1
  = 0x0000_0100 + 0x0008_0000 - 1
  = 0x0008_00FF
- SLAVE2_START_ADDR = SLAVE1_END_ADDR+ 1
  = 0x0008_00FF + 1
  = 0x0008_0100
- SLAVE2_END_ADDR = SLAVE2_START_ADDR + slave2_range - 1
  = 0x0008_0100 + 0x0000_4000 - 1
  = 0x0008_40FF
- SLAVE3_START_ADDR = SLAVE2_END_ADDR + 1
  = 0x0008_40FF + 1
  = 0x0008_4100
- SLAVE3_END_ADDR = SLAVE3_START_ADDR + slave3_range - 1
  = 0x0008_4100 + 0x0000_0200 - 1
  = 0x0008_42FF

Random Addressing

- SLAVE0_START_ADDR = 0x0000_0000
- SLAVE0_END_ADDR = SLAVE0_START_ADDR + slave0_range - 1

  = 0x0000_0000 + 0x0000_0100 - 1

  = 0x0000_00FF

- SLAVE1_START_ADDR = 0x0008_0000
- SLAVE1_END_ADDR = SLAVE1_START_ADDR + slave1_range - 1

  = 0x0008_0000 + 0x0008_0000 - 1

  = 0x000F_FFFF

- SLAVE2_START_ADDR = 0x0010_0000
- SLAVE2_END_ADDR = SLAVE2_START_ADDR + slave2_range - 1

  = 0x0010_0000 + 0x0000_4000 - 1

  = 0x0010_3FFF

- SLAVE3_START_ADDR = 0x0010_4000
- SLAVE3_END_ADDR = SLAVE3_START_ADDR + slave3_range - 1

  = 0x0010_4000 + 0x0000_0200 - 1

  = 0x0010_41FF

Address Overlapping Example

- SLAVE0_START_ADDR = 0x0000_0100
- SLAVE0_END_ADDR = SLAVE0_START_ADDR + slave0_range - 1

  = 0x0000_0100 + 0x0000_0100 - 1

  = 0x0000_01FF

- SLAVE1_START_ADDR = 0x0020_0000
- SLAVE1_END_ADDR = SLAVE1_START_ADDR + slave1_range - 1

  = 0x0020_0000 + 0x0008_0000 - 1

  = 0x0027_FFFF

- SLAVE2_START_ADDR = 0x0000_4000
- SLAVE2_END_ADDR = SLAVE2_START_ADDR + slave2_range - 1

  = 0x0000_4000 + 0x0000_4000 - 1

  = 0x0000_7FFF

- SLAVE3_START_ADDR = 0x0000_5000
- SLAVE3_END_ADDR = SLAVE3_START_ADDR + slave3_range - 1

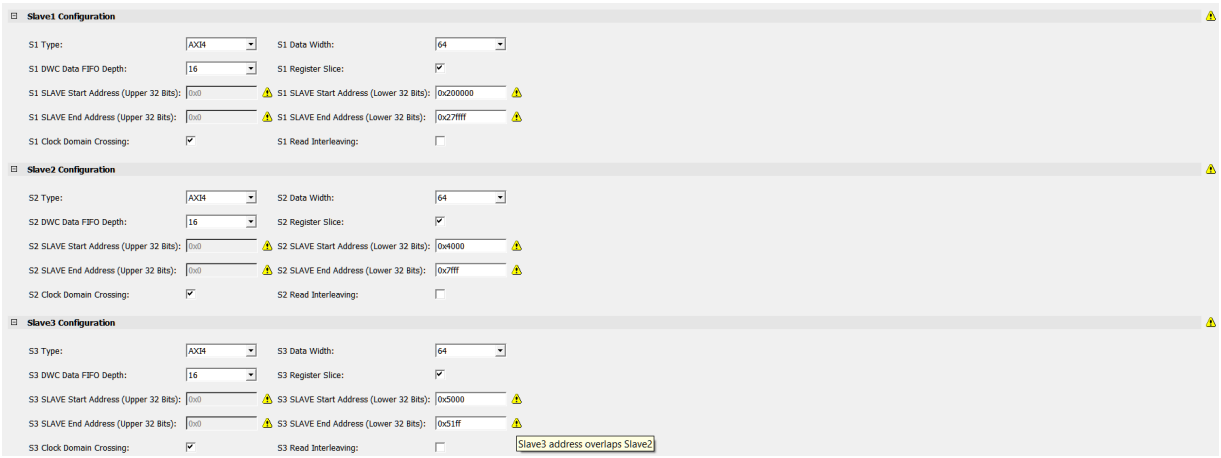
  = 0x0000_5000 + 0x0000_0200 - 1

  = 0x0000_51FF

SLAVE3 address overlaps SLAVE2.

The following figure shows address overlapping error in CoreAXI4Interconnect Configurator:

Figure 13 • **Overlapping Address**



4.6 Auxiliary Parameters Configuration

The following sections describe the auxiliary parameters and their configuration details:

4.6.1 Read Arbitration Enable

The **Read Arbitration Enable** parameter defines the mechanism used for Read Data paths. When Read Arbitration Enable is enabled, a round-robin arbitrator is used to select the next slave to perform read data transaction. This leads to a dead-cycle on back-to-back reads from the same slave device. While Read Arbitration Enable is disabled, an ordered queue is used to select the slave. In this mode, it is possible for back-to-back cycles from the same slave with no dead-cycles.

4.6.2 Crossbar Data Width

The **Crossbar Data Width** parameter configuration depends on the system design.

Following design example illustrates the configuration of **Crossbar Data Width** parameter

Number of Masters - 2

Number of Slaves - 4

Master 0 - Microprocessor

Master 1 - DMA

Master 0 Data Width - 32 Bits

Master 1 Data Width - 64 Bits

Slave 0 - LSRAM

Slave 1 - FIFO

Slave 2 - DDR3

Slave 3 - DMA

Slave 0 Data Width - 32 Bits

Slave 1 Data Width - 32 Bits

Slave 2 Data Width - 64 Bits

Slave 3 Data Width - 32 Bits

In the design, Microprocessor configures the DMA to transfer data between DDR3 and FIFO. LSRAM is used as an application memory. In this scenario, most of the time DMA occupies AXI4 bus that is, DMA is

busy in transferring data between DDR3 and FIFO. DDR3 data width is 64 and DMA data width is also 64. Microprocessor data width is 32 bits and data width of all the slaves accessed by Microprocessor is also 32 bits. However, Microprocessor occupies less AXI4 bus bandwidth compared to DMA and due to that the **Crossbar Data Width** should be configured to 64 to achieve high performance.

If **Crossbar Data Width** is configured to 32 bits then data width converter implemented between Master 1 (DMA) and the AXI4 crossbar and between AXI4 crossbar and Slave 2 (DDR3) which increases latency and reduce the performance.

4.6.3 Mx Read Interleaving and Sy Read Interleaving

The configuration of **Mx Read Interleaving and Sy Read Interleaving** parameters depends on the read interleave feature supported by masters and slaves used in the system design.

Following design example illustrates the configuration of **Mx Read Interleaving and Sy Read Interleaving** parameters

Number of Masters - 3

Number of Slaves - 4

Master 0 - DMA1. Supports read interleaving.

Master 1 - DMA2. Supports read interleaving.

Master 2 - Microprocessor. Doesn't support read interleaving.

Slave 0 - DMA1. Does not support read interleaving.

Slave 1 - DMA2. Does not support read interleaving.

Slave 2 - DDR3. Supports read interleaving.

Slave 3 - FIFO. Does not support read interleaving.

For the preceding configuration, masters and slaves supports read interleaving, its corresponding **Mx Read Interleaving** and **Sy Read Interleaving** parameters should be enabled to improve both resource utilization and performance. **Mx Read Interleaving** and **Sy Read Interleaving** configuration should be as follows:

M0 Read Interleaving - Enabled

M1 Read Interleaving - Enabled

M2 Read Interleaving - Disabled

S0 Read Interleaving - Disabled

S1 Read Interleaving - Disabled

S2 Read Interleaving - Enabled

S3 Read Interleaving - Disabled.

4.7 Connectivity Matrix

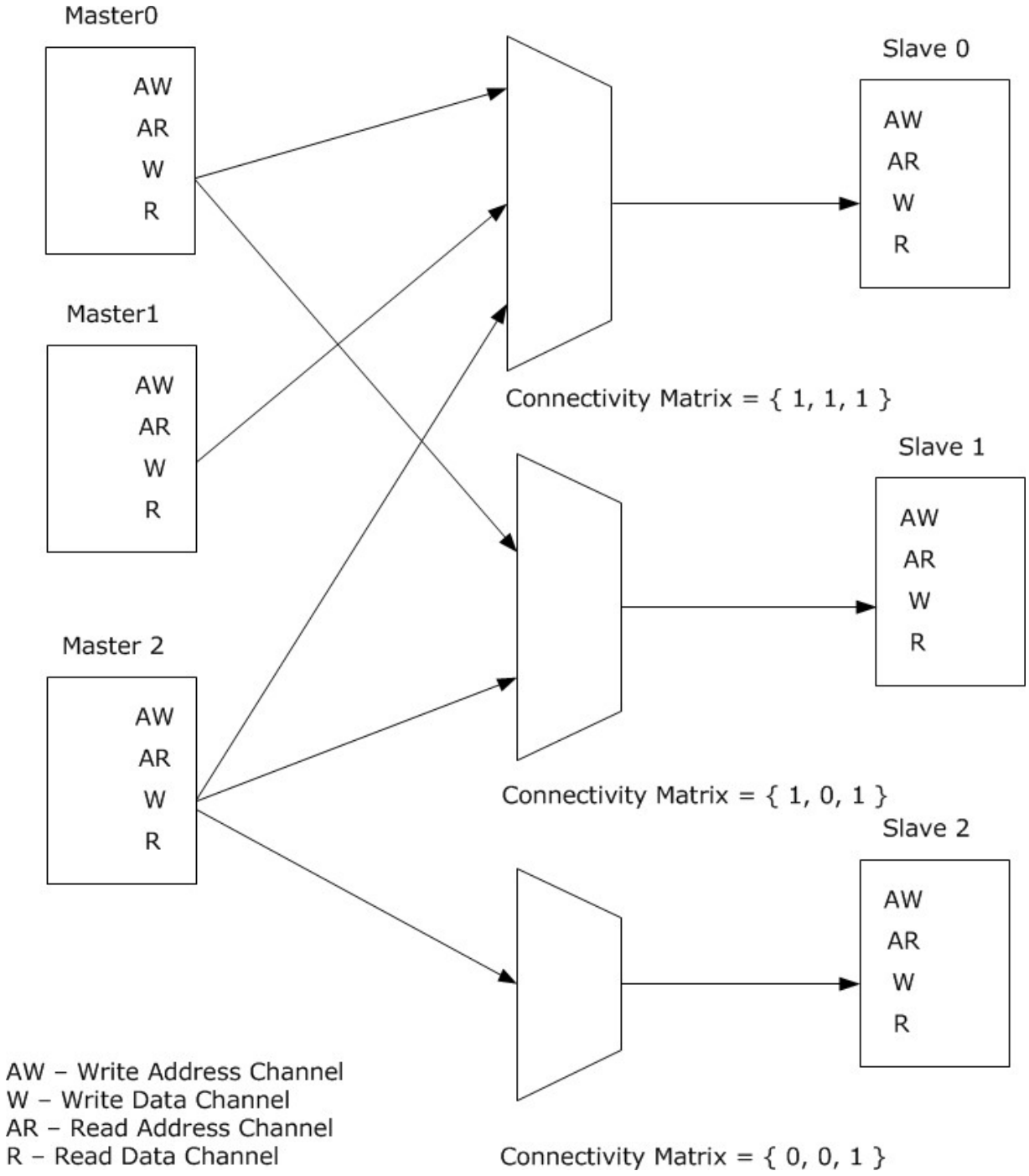
The MASTERx_WRITE_SLAVEy and MASTERx_READ_SLAVEy parameters are used to prune the decode and arbitration logic. Each master has a bit per slave that defines whether the master can write to the slave (MASTERx_WRITE_SLAVEy, where x is 0 to 15 referring to MASTER0 to MASTER15 and y is 0 to 31 referring to SLAVE0 to SLAVE31) and another bit per slave that defines whether the master canread from the slave (MASTERx_READ_SLAVEy).

MASTERx_WRITE_CONNECTIVITY is a concatenation of {MASTERx_WRITE_SLAVE31 …. MASTERx_WRITE_SLAVE0} and MASTERx_READ_CONNECTIVITY is a concatenation of

{MASTERx_READ_SLAVE31 …. MASTERx_READ_SLAVE0}. These matrices are used to remove slaves.whose connectivity bit set to zero from arbitration and other logic.

The following figure shows an example of how the MASTER_WRITE_CONNECTIVITY is used. MASTER_READ_CONNECTIVITY is defined in a similar manner.

Figure 14 • **Master Wire Connectivity Example**



4.8 AHB-Lite to AXI4 Master Conversion

A connected AHB-Lite master (MASTER_TYPE = 10) converts the AHB-Lite protocol to AXI4. The differences between protocols are as follows:

- AXI has five independent channels (write and read address/control, write and read data, and write response) whereas AHB-Lite has only one channel.
- AHB-Lite has an undefined length burst type, INCR, with no way to predict when it will end until a new transaction starts or the bus goes idle; AXI specifies the length of transactions at the start.
- AXI and AHB-Lite have "response" indications for error conditions, but they are handled differently. AHB-Lite is on per-beat basis within a burst for both reads and writes, whereas AXIs response comes on per beat basis for reads, and comes once at the end of a burst for writes.
- AHB-Lite has only one shared "ready" signal for address or control and, both read and write data, whereas AXI has a "ready" signal on each of the five channels, along with associated "valid" signals.

During back to back transactions without "BUSY type transfer, the last data beat coincides with the next control beat, and "HREADY" is not asserted until both the data are accepted by AXI (AXI RVALID for reads or WREADY for writes) and, the required read or write control channel is ready (AWREADY or ARREADY).The AHB-Lite converter always provides AHB-Lite response at the end of the AHB-Lite burst. AXI4 provides write response at the end of AXI4 write burst and read response at every beat transfer. In both the cases AHB-Lite converter provides response to AHB-Lite at the end of AHB-Lite burst. In case of the read operation, AHB-Lite converter stores the error response if error response is received during the burst transaction and provides error response at the end of the AHB-Lite burst.

5 Core Interfaces

The following sections describe the I/O signals and parameters for CoreAXI4Interconnect:

5.1 I/O Signals

The following table describes the port signals for the CoreAXI4Interconnect macro as shown in Figure 2, page 6.

Note: All signals are active High (logic 1) unless otherwise noted.

Note: In the signal names and parameters listed here, "x" represents a range of 0 to 15 for master signals and “y” represents a range of 0 to 31 for slave signals.

Table 7 • **CoreAXI4Interconnect I/O Signals**

Name	Type	Description
Global Signals		
ACLK	Input	AXI4 crossbar clock signal.
ARESETN	Input	**Note:** Active low asynchronous reset.Asynchronous reset is synchronized in ACLK, MCLK_x, and SCLK_y domains inside AXI4 Interconnect. After de-assertion of asynchronous reset, user must wait for two clock cycles of the respective MCLK_x, before initiating master transaction.
Master Signals – Port 0 to Port 15		
Master Clock signals		
M_CLKx	Input	Master clock for port x. If clock domain crossing is required on Master port x, this clock must be connected to clock associated with the ports bus (AHB-Lite/AXI). Parameter Mx Clock Domain Crossing is used to enable the port.
Master AHB-Lite signals		
MASTERx_HADDR[31: 0]	Input	Read/Write Write address. It gives the address of the first transfer in a transaction.
MASTERx_HBURST[2:0]	Input	Burst type. The burst type and the length information, determines how the address for each transfer within the burst is calculated.
MASTERx_HMASTLOCK	Input	When High, this signal indicates that the current transfer is part of a locked sequence. This is transferred to the lower bit of the slaves 2- bit AXI "lock" signal with the upper bit assigned to zero.
MASTERx_HPROT[6:0]	Input	HPROT[0] is inverted to produce AxPROT[2]. When HPROT[0] is low, it indicates an instruction/opcode access, high indicates a data access.

Table 7 • **CoreAXI4Interconnect I/O Signals**

Name	Type	Description
	Input	HPROT[1] is routed to AxPROT[0], to indicate user access when low, privileged access when high.
	Input	HPROT[2] when high, indicates that data is bufferable; this is routed to AxCACHE[0].
	Input	HPROT[3] when high, indicates data is catchable; this is routed to AxCACHE[1] and [2].
	Input	HPROT [6:4] are included for possible future expansion to support AHB5.
	Input	HPROT[5] indicates "allocate", where data must be looped-up in cache, and is routed to AxCACHE[3] and [2].
	Input	HPROT[4] and [6] are unused.
MASTERx_HSIZE[2:0]	Input	This signal indicates the size in bytes of each transfer in the burst.
MASTERx_HNONSEQ	Input	Non-secure access. This is not a standard AHB signal, but may be useful in bridging to AXI, where it becomes bit[1] of AxPROT[2:0] to indicate a non-secure access when driven high.
MASTERx_HTRANS[1:0]	Input	Transaction type: BUSY, NONSEQUENTIAL, SEQUENTIAL, or IDLE.
MASTERx_HWDATA [MASTERx_DATA_WIDTH-1:0]	Input	Write Data
MASTERx_HRDATA [MASTERx_DATA_WIDTH-1:0]	Output	Read Data
MASTERx_HWRITE	Input	Write transaction when high, read when low.
MASTERx_HREADY	Output	AXI Interconnect ready: when asserted during a write transfer, indicates it accepted the write; when asserted during a read transfer, indicates that read data is available; when asserted while HTRANS indicates “NONSEQ”, it indicates that the interconnect accepts the command, and also has processed the last read or write beat if it is in progress, simultaneously.
MASTERx_HRESP	Output	Interconnect error indicator when high.
MASTERx_HSEL	Input	Optional decode signal, when high indicates the current transaction is intended for this interconnect. Tie to “1” if not required.
Master Address Write Channels		
MASTERx_AWID [ID_WIDTH-1:0]	Input	Write address ID. The identification tag for the write address group of signals.
MASTERx_AWADDR [ADDR_WIDT H-1:0]	Input	Write address. The write address gives the address of the first transfer in a write burst transaction.
MASTERx_AWLEN[7:0]	Input	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers, associated with the address. This changes between AXI3 and AXI4.
MASTERx_AWSIZ[2:0]	Input	Burst size. Size of each transfer in the burst.

Table 7 • **CoreAXI4Interconnect I/O Signals**

Name	Type	Description
MASTERx_AWBURST[1: 0]	Input	Burst type. The burst type and the size information, determine how the address for each transfer within the burst is calculated.
MASTERx_AWLOCK[1: 0]	Input	Lock type. Provides additional information about the atomic characteristics of the transfer. This changes between AXI3 and AXI4.
MASTERx_AWCACHE[3: 0]	Input	Memory type. Shows how transactions are required to progress through a system.
MASTERx_AWPROT[2: 0]	Input	Protection type. It gives privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
MASTERx_AWQOS[3:0]	Input	Quality of Service, QoS. The QoS identifier sent for each write transaction. Implemented only in AXI4.
MASTERx_AWREGION [3:0]	Input	Region identifier. Permits a single physical interface on a slave, to be used for multiple logical interfaces. Implemented only in AXI4.
MASTERx_AWUSER [USER_WIDT H-1:0]	Input	User signal. Optional user-defined signal in the write address channel. Supported only in AXI4.
MASTERx_AWVALID	Input	Write address valid. Channel is signaling valid write address and control information.
MASTERx_AWREADY	Output	Write address ready. Slave is ready to accept an address and associated control signals.
MASTERx_WID [ID_WIDTH-1:0]	Input	Write ID tag. This signal is the ID tag of the write data transfer. Supported only in AXI3.
MASTERx_WDATA [MASTERx_DATA_WIDTH- 1:0]	Input	Write Data
MASTERx_WSTRB [(MASTERx_DATA_WIDTH/8-1:0]	Input	Write strobes. Shows which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
MASTERx_WLAST	Input	Write last. Last transfer in a write burst.
MASTERx_WUSER [USER_WIDTH-1:0]	Input	User signal. Optional user-defined signal in the write data channel. Supported only in AXI4.
MASTERx_WVALID	Input	Write valid. Valid write data and strobes are available.
MASTERx_WREADY	Output	Write ready. Slave can accept the write data.
Master Write Response Channels		
MASTERx_BID [ID_WIDTH-1:0]	Output	Response ID tag. This signal is the ID tag of the write response.
MASTERx_BRESP[1:0]	Output	Write response. Status of the write transaction.
MASTERx_BUSER [USER_WIDTH- 1:0]	Output	User signal. Optional user-defined signal in the write response channel. Supported only in AXI4.
MASTERx_BVALID	Output	Write response valid. Channel is signaling a valid write response.
MASTERx_BREADY	Input	Response ready. Master can accept a write response.
Master Address Read Channels		

Table 7 • **CoreAXI4Interconnect I/O Signals**

Name	Type	Description
MASTERx_ARID [ID_WIDTH-1:0]	Input	Read address ID. Tag for the read address group of signals.
MASTERx_ARADDR [ADDR_WIDT H-1:0]	Input	Read address. Address of the first transfer in a read burst transaction.
MASTERx_ARLEN[7:0]	Input	Burst length. Exact number of transfers in a burst. This information determines the number of data transfers associated with the address. This changes between AXI3 and AXI4.
MASTERx_ARSIZ[2:0]	Input	Burst size. Size of each transfer in the burst.
MASTERx_ARBURST[1: 0]	Input	Burst type. The burst type and the size information, determine how the address for each transfer within the burst is calculated.
MASTERx_ARLOCK[1:0]	Input	Lock type. Provides additional information about the atomic characteristics of the transfer. This changes between AXI3 and AXI4.
MASTERx_ARCACHE[3: 0]	Input	Memory type. Shows how transactions are required to progress through a system.
MASTERx_ARPROT[2:0]	Input	Protection type. It gives privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
MASTERx_ARQOS[3:0]	Input	Quality of Service, QoS. The QoS identifier sent for each read transaction. Implemented only in AXI4.
MASTERx_ARREGION[3: 0]	Input	Region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces. Implemented only in AXI4.
MASTERx_ARUSER [USER_WIDT H-1:0]	Input	User signal. Optional user-defined signal in the read address channel. Supported only in AXI4.
MASTERx_ARVALID	Input	Read address valid. Channel is signaling valid read address and control information.
MASTERx_ARREADY	Output	Read address ready. Slave is ready to accept an address and associated control signals.
Master Read Data Channels		
MASTERx_RID [ID_WIDTH-1:0]	Output	Read ID tag. This signal is the identification tag for the read data group of signals generated by the slave.
MASTERx_RDATA [MASTERx_DATA_WIDTH- 1:0]	Output	Read Data
MASTERx_RRESP[1:0]	Output	Read response. Status of the read transfer.
MASTERx_RLAST	Output	Read last. Last transfer in a read burst.
MASTERx_RUSER [USER_WIDTH- 1:0]	Output	User signal. Optional user-defined signal in the read data channel. Supported only in AXI4.
MASTERx_RVALID	Output	Read valid. Channel is signaling the required read data.
MASTERx_RREADY	Input	Read ready. Master can accept the read data and response information.
Slave Signals – Port 0 to Port 31		
Slave Clock Signals		

Table 7 • **CoreAXI4Interconnect I/O Signals**

Name	Type	Description
S_CLKy	Input	Slave clock for port x. If clock domain crossing is required on slave port x, this clock must be connected to clock associated with the port's AXI bus. Parameter SLAVEy_CLOCK_DOMAIN_CROSSING is used to enable this port.
Slave Address Write Channels		
SLAVEy_AWID [(ID_WIDTH + Log2 (NUM_MASTERS))-1:0]	Output	Write address ID. This signal is the identification tag for the write address group of signals.
SLAVEy_AWADDR [ADDR_WIDTH- 1:0]	Output	Write address. The write address gives the address of the first transfer in a write burst transaction.
SLAVEy_AWLEN[7:0]	Output	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. This changes between AXI3 and AXI4.
SLAVEy_AWSIZ[2:0]	Output	Burst size. This signal indicates the size of each transfer in the burst.
SLAVEy_AWBURST[1:0]	Output	Burst type. The burst type and the size information, determine how the address for each transfer within the burst is calculated.
SLAVEy_AWLOCK[1:0]	Output	Lock type. Provides additional information about the atomic characteristics of the transfer. This changes between AXI3 and AXI4.
SLAVEy_AWCACHE[3:0]	Output	Memory type. It shows how transactions are required to progress through a system.
SLAVEy_AWPROT[2:0]	Output	Protection type. It gives privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
SLAVEy_AWQOS[3:0]	Output	Quality of Service, QoS. The QoS identifier sent for each write transaction. Implemented only in AXI4.
SLAVEy_AWREGION[3: 0]	Output	Region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces. Implemented only in AXI4.
SLAVEy_AWUSER [USER_WIDTH- 1:0]	Output	User signal. Optional user-defined signal in the write address channel. Supported only in AXI4.
SLAVEy_AWVALID	Output	Write address valid. Channel is signaling valid write address and control information.
SLAVEy_AWREADY	Input	Write address ready. Slave is ready to accept an address and associated control signals.
Slave Write Data Channels		
SLAVEy_WID [(ID_WIDTH + Log2 (NUM_MASTERS))-1:0]	Output	Write ID tag. This signal is the ID tag of the write data transfer. Supported only in AXI3.
SLAVEy_WDATA [SLAVEy_DATA_WIDTH-1:0]	Output	Write Data
SLAVEy_WSTRB [(SLAVEy_DATA_WIDTH/8)-1:0]	Output	Write strobes. It gives, which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
SLAVEy_WLAST	Output	Write last. Last transfer in a write burst.

Table 7 • **CoreAXI4Interconnect I/O Signals**

Name	Type	Description
SLAVEy_WUSER [USER_WIDTH- 1:0]	Output	User signal. Optional user-defined signal in the write data channel. Supported only in AXI4.
SLAVEy_WVALID	Output	Write valid. Valid write data and strobes are available.
SLAVEy_WREADY	Input	Write ready. Slave can accept the write data.
Slave Write Response Channels		
SLAVEy_BID [(ID_WIDTH + Log2 (NUM_MASTERS))-1:0	Input	Response ID tag. This signal is the ID tag of the write response.
SLAVEy_BRESP[1:0]	Input	Write response. Status of the write transaction.
SLAVEy_BUSER [USER_WIDTH- 1:0	Input	User signal. Optional user-defined signal in the write response channel. Supported only in AXI4.
SLAVEy_BVALID	Input	Write response valid. Channel is signaling a valid write response.
SLAVEy_BREADY	Output	Response ready. Master can accept a write response.
Slave Address Read Channels		
SLAVEy_ARID [(ID_WIDTH + Log2 (NUM_MASTERS))-1:0]	Output	Read address ID. This signal is the identification tag for the read address group of signals.
SLAVEy_ARADDR [ADDR_WIDTH- 1:0]	Output	Read address. The read address gives the address of the first transfer in a read burst transaction.
SLAVEy_ARLEN[7:0]	Output	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. This changes between AXI3 and AXI4.
SLAVEy_ARSIZ[2:0]	Output	Burst size. Size of each transfer in the burst.
SLAVEy_ARBURST[1:0]	Output	Burst type. The burst type and the size information, determine how the address for each transfer within the burst is calculated.
SLAVEy_ARLOCK[1:0]	Output	Lock type. Provides additional information about the atomic characteristics of the transfer. This changes between AXI3 and AXI4.
SLAVEy_ARCACHE[3:0]	Output	Memory type. It shows how transactions are required to progress through a system.
SLAVEy_ARPROT[2:0]	Output	Protection type. It gives privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
SLAVEy_ARQOS[3:0]	Output	Quality of Service, QoS. The QoS identifier sent for each read transaction. Implemented only in AXI4.
SLAVEy_ARREGION[3: 0]	Output	Region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces. Implemented only in AXI4.
SLAVEy_ARUSER [USER_WIDTH- 1:0]	Output	User signal. Optional user-defined signal in the read address channel. Supported only in AXI4.
SLAVEy_ARVALID	Output	Read address valid. Channel is signaling valid read address and control information.
SLAVEy_ARREADY	Input	Read address ready. Slave is ready to accept an address and associated control signals.

Table 7 • **CoreAXI4Interconnect I/O Signals**

Name	Type	Description
Slave Read Data Channels		
SLAVEy_RID[(ID_WIDTH + Log2 (NUM_MASTERS))-1:0]	Input	Read ID tag. This signal is the identification tag for the read data group of signals generated by the slave.
SLAVEy_RDATA [SLAVEy_DATA_WIDTH-1:0]	Input	Read data.
SLAVEy_RRESP[1:0]	Input	Read response. Status of the read transfer.
SLAVEy_RLAST	Input	Read last. Last transfer in a read burst.
SLAVEy_RUSER [USER_WIDTH- 1:0]	Input	User signal. Optional User-defined signal in the read data channel. Supported only in AXI4.
SLAVEy_RVALID	Input	Read valid. Channel is signaling the required read data.
SLAVEy_RREADY	Output	Read ready. Master can accept the read data and response information.

5.2 Core Parameters

The following table describes the CoreAXI4Interconnect parameters for configuring the RTL code. All parameters are integer types:

Table 8 • **CoreAXI4Interconnect Parameters**

Parameter Name	Valid Values	Default	Description
NUM_MASTERS	1 to 16	2	**Number of Masters** Number of connected masters to be supported by the crossbar.
NUM_SLAVES	1 to 32	2	**Number of Slaves** Number of connected slaves to be supported by the crossbar.
ID_WIDTH	1 to 8	1	**ID Width** Number of identification tag bits to be supported for each port. It is same for all masters and slaves. Slave ID bits have master infrastructure number pre-pended to ID bits. The slave port ID has more bits than master port ID as Log2 (NUM_MASTER) bits are pre-pended to slave port ID. **Note:** When Mx Read Interleaving or Sy Read Interleaving parameters are configured to 1, resource utilization increase as ID_WIDTH increases. User should configured ID_WIDTH appropriately.
DATA_WIDTH	32, 64, 128, 256, 512	64	**Crossbar Data Width** Data width for the core, Crossbar, and other components.
ADDR_WIDTH	16 to 64	32	**Address Width** Number of bits in the address.
USER_WIDTH	1-64	1	**User Width** Number of bits for USER signals RUSER and WUSER.

Table 8 • **CoreAXI4Interconnect Parameters**

Parameter Name	Valid Values	Default	Description
DWC_ADDR_FIFO_DEPTH_CEILING	4 to 64	10	**DWC Address FIFO Depth Ceiling** DWC Address FIFO Depth Ceiling is used to bind the size of the command FIFOs in the data width converter. It is a general parameter that the user can set to have small command FIFOs if they run out of resources.
NUM_THREADS	1-4	1	**Number of Threads** Number of independent threads per master supported.
OPEN_TRANS_MAX	1-8	2	**Max Outstanding Transactions** Maximum number of outstanding transactions per thread, per Master, that is, transaction acceptance limits for reads and writes. Used for both reads and writes, separate counts are used.
SLV_AXI4PRT_ADDRDEPTH	2-8	4	**Slave FIFO Address Depth** Depth of synchronous FIFO used in Slave Protocol Converter, to hold address for burst read or write transactions, when converting from AXI4 to AXI3 or AXI4Lite. Used to decouple Master from Slave side when burst translation is needed (breaking AXI4 burst to multiple smaller bursts) for reads and writes. **Note:** This parameter is common for all the AXI4-Lite and AXI3 slaves
SLV_AXI4PRT_DATADEPTH	2-9	4	**Slave FIFO Data Depth** Depth of synchronous FIFO used in Slave Protocol Converter, to hold data for burst read or write transactions, when converting from AXI4 to AXI3 or AXI4Lite. Used to decouple Master from Slave side when burst translation is needed (breaking AXI4 burst to multiple smaller bursts) for reads and writes. **Note:** This parameter is common for all the AXI4-Lite and AXI3 slaves
CROSSBAR_MODE	SASD SAMD	SAMD	**Crossbar Mode** Indicates if crossbar is SAMD or SASD. SAMD is the performance-optimized option with Shared- address paths but with multiple data paths. SASD is the area optimized architecture having shared address and data paths. **Note:** This parameter is user configurable only if Number of Masters is greater than 1, Number of Slaves is greater than 1 and Optimization is selected to User.

Table 8 • **CoreAXI4Interconnect Parameters**

Parameter Name	Valid Values	Default	Description
RD_ARB_EN	0 or 1	1	**Read Arbitration Enable** RD_ARB_EN defines the way Read Data transactions are handled. When RD_ARB_EN is not asserted, Read Data cycles are handled in an ordered manner on a per port basis (in order they were issued). When RD_ARB_EN is asserted, a round-robin arbitrator is used to select next slave to read from based on RVALIDs asserted.
OPTIMIZATION	Performance Area User	User	**Optimization** This parameter is used to choose performance, area or customized optimization. When Performance is selected, below parameters are configured to achieve maximum performance optimization: • NUM_THREADS (Number Of Threads)- 4 • OPEN_TRANS_MAX (Max Outstanding Transactions) - 8 • SLV_AXI4PRT_ADDRDEPTH (Slave FIFO Address Depth) - 8 • SLV_AXI4PRT_DATADEPTH (Slave FIFO Data Depth)- 9 • DWC_ADDR_FIFO_DEPTH (DWC Address FIFO Depth Ceiling)- 64 • RD_ARB_EN (Read Arbitration Enable) - 0 • CROSSBAR_MODE (Crossbar Mode) - SAMD When Area is selected, below parameters are configured to achieve maximum area optimization: • NUM_THREADS (Number Of Threads) - 1 • OPEN_TRANS_MAX (Max Outstanding Transactions) - 1 • SLV_AXI4PRT_ADDRDEPTH (Slave FIFO Address Depth) - 2 • SLV_AXI4PRT_DATADEPTH (Slave FIFO Data Depth) - 2 • DWC_ADDR_FIFO_DEPTH_CEILING (DWC Address FIFO Depth Ceiling) - 4 • RD_ARB_EN (Read Arbitration Enable) - 1 • CROSSBAR_MODE (Crossbar Mode) - SASD When User is selected, user can configure customized optimization.

Table 8 • **CoreAXI4Interconnect Parameters**

Parameter Name	Valid Values	Default	Description
MASTERx_TYPE	2'b00, 2'b01, 2'b10, 2'b11	2'b00	**Mx Type** Type of interface for the master port. Valid values are as follows: • 2'b00 - AXI4 master • 2'b01 - AXI4-Lite master • 2'b10 - AHB-Lite master • 2'b11 - AXI3 master
MASTERx_DATA_WIDTH	32, 64, 128, 256, 512	64	**Mx Data Width** Data widths of master ports are as follows: • If master data width <Crossbar data width, the master data converter up-scales. • If master data width >Crossbar data width, the master data converter down-scales. • If master data width =Crossbar data width no data- width conversion, so master data converter is in pass-through mode. • AXI4-Lite masters are limited by protocol to data width of 32 and 64.
MASTERx_DWC_DATA_FIFO_DEPTH	16, 32, 64	16	**Mx DWC Data FIFO Depth** Depth of synchronous FIFO used in master data width converter to buffer read or write data.
MASTERx_CHAN_RS	0 or 1	1	**Mx Register Slice** When 1, full Register Slice is added to the five channels of master i.e master write address channel, master write data channel, master write response channel, master read address channel and master read data channel. Adding a register slice can increase maximum frequency of operation at the cost of one clock cycle latency on all the channels.
MASTERx_CLOCK_DOMAIN_CROSSING	0 or 1	0	**Mx Clock Domain Crossing** When high, an asynchronous FIFO is instantiated for all the five channels (Write Address, Read Address, Write Data, Read Data, and Write Response) at the master port to transfer data between master clock domain to crossbar clock domain and vice versa. **Note:** Asynchronous FIFO with flow control is used. When asynchronous FIFO is full, no additional write is performed and when asynchronous FIFO is empty, no additional read is performed. Asynchronous FIFO depth is fixed to 8.

Table 8 • **CoreAXI4Interconnect Parameters**

Parameter Name	Valid Values	Default	Description
MASTERx_READ_INTERLEAVING	0 or 1	0	**Mx Read Interleaving** When '1' enables read interleaving and **Master Converter** passes master address id (AWID/ARID) and write data id (WID) for AXI3 mater to the **AXI4 Crossbar.** The AXI4 Crossbar append Log2(Number of Masters) bits to the address id and data id and passes to the slave if read interleaving is enabled in targeted slave. The AXI4 Crossbar decodes response id (BID) and read id (RID) received from the **Slave Converter** and routed to the targeted master When '0' disables read interleaving and **Master Converter** drives 0 to master address id (AWID/ARID) and write data id (WID) for AXI3 master and passes to the **AXI4 Crossbar**. The **Master Converter** stores the master address id (AWID/ARID) and data id (WID) for AXI3 master, internally and retrieve it when write/read response received from the AXI4 Crossbar. **Note:** This parameter is configurable only for AXI4 and AXI3 masters.
SLAVEy Type	2'b00, 2'b01, 2'b11 (2'b10 is reserved)	2'b00	**Sy Type** Type of interface for the slave port. Valid values are as follows: • 2'b00 - AXI4 slave • 2'b01 - AXI4-Lite slave • 2'b11 - AXI3 slave
SLAVEy_DATA_WIDTH	32, 64, 128, 256, 512	64	**Sy Data Width** Data widths of slave ports are as follows: • If slave data width <Crossbar data width the slave data converter down-scales. • If slave data width >Crossbar data width the slave data converter up-scales. • If slave data width =Crossbar data width no data-width conversion so slave data converter is in pass- through mode. • AXI4-Lite slaves are limited by protocol to data width of 32 and 64.
SLAVEy_DWC_DATA_FIFO_DEPTH	16, 32, 64	16	**Sy DWC Data FIFO Depth** Depth of synchronous FIFO used in slave data width converter to buffer read or write data.

Table 8 • **CoreAXI4Interconnect Parameters**

Parameter Name	Valid Values	Default	Description
SLAVEy_CLOCK_DOMAIN_CROSSING	0 or 1	0	**SLAVEy Clock Domain Crossing** When high, an asynchronous FIFO is instantiated for all the five channels (Write Address, Read Address, Write Data, Read Data and Write Response) at the slave port to transfer data between crossbar clock domain to slave clock domain and vice versa **Note:** Asynchronous FIFO with flow control is used. When asynchronous FIFO is full, no additional write is performed and when asynchronous FIFO is empty, no additional read is performed. Asynchronous FIFO depth is fixed to 8.
SLAVEy_READ_INTERLEAVING	0 or 1	0	**Sy Read Interleaving** When '1' enables read interleaving and **Slave Converter** passes slave address id (AWID/ARID) and write data id (WID) for AXI3 slave received from **AXI4 Crossbar** to the slave. When '0' disables read interleaving and **Slave Converter** drives 0 to slave address id (AWID/ARID) and write data id (WID) for AXI3 slave received from the **AXI4 Crossbar** and passes to the slave. **Note:** This parameter is configurable only for AXI4 and AXI3 masters.
SLAVEy_CHAN_RS	0 or 1	1	**Slavey Register Slice** When 1, full Register Slice is added to the five channels of slave that is, slave write address channel, slave write data channel, slave write response channel, slave read address channel and slave read data channel. Adding a register slice can increase maximum frequency of operation at the cost one clock cycle latency on all the channels.
SLAVEy_START_ADDR	0 to 0xFFFFFFFF	-	**Sy Slave Start Address (Lower 32 Bits)** Defines the lower 32 bits of slavey start address
SLAVEy_START_ADDR_UPPER	0 to 0xEFFFFFFF	-	**Sy Slave Start Address (Upper 32 Bits)** Defines the upper 32 bits of slavey start address. This parameter is configurable only if **Address Width** parameter is greater than 32. For **Address Width** less than 32, this parameter will be ignored.
SLAVEy_END_ADDR	0 to 0xFFFFFFFF	-	**Sy Slave End Address (Lower 32 Bits)** Defines the lower 32 bits of slavey end address.
SLAVEy_END_ADDR_UPPER	0 to 0xFFFFFFFF	-	**Sy Slave End Address(Upper 32 Bits)** Defines the upper 32 bits of slavey end address. This parameter is configurable only if **Address Width** parameter is greater than 32. For **Address Width** less than 32, this parameter will be ignored.

Table 8 • **CoreAXI4Interconnect Parameters**

Parameter Name	Valid Values	Default	Description
MASTERx_WRITE_SLAVEy (where, "x is 0 … 15 i.e. one per master port "y is 0 … 31 that is, one per slave port)	0 or 1	1	**Mx access Sy (Enable Master Write Access)** Bit per master per slave indicating if a master can write to a slave port. Used to "trim" internal decode/arbitration logic for Masters that cannot write to a specific slave.
MASTERx_READ_SLAVEy (where,"x is 0 … 15 that is, one per master port "y is 0 … 31 that is, one per slave port)	0 or 1	1	**Mx access Sy (Enable Master Read Access)** Bit per master per slave. Used to "trim" internal decode/arbitration logic for Masters that cannot read from a specific slave.

6 Clocking and Reset

This section describes the options available for clocking and reset used in the IP cores.

6.1 Clocking

Following clocks are used in the IP core:

- ACLK: Crossbar clock.
- M_CLKx: AXI4/AXI3/AXI4-Lite/AHB-Lite master clock. This clock is available only if **Mx Clock Domain Crossing** parameter is enabled.
- S_CLKy: AXI4/AXI3/AXI4-Lite slave clock. This clock is available only if **Sy Clock Domain Crossing** parameter is enabled.

All the clocks are asynchronous that is clocks can have different frequencies and out of phase. When **Mx Clock Domain Crossing** parameter is enabled. Asynchronous FIFO used to transfer data between M_CLKx and ACLK. Same way, when **Sy Clock Domain Crossing** parameter is enabled. Asynchronous FIFO used to transfer data between S_CLKy and ACLK.

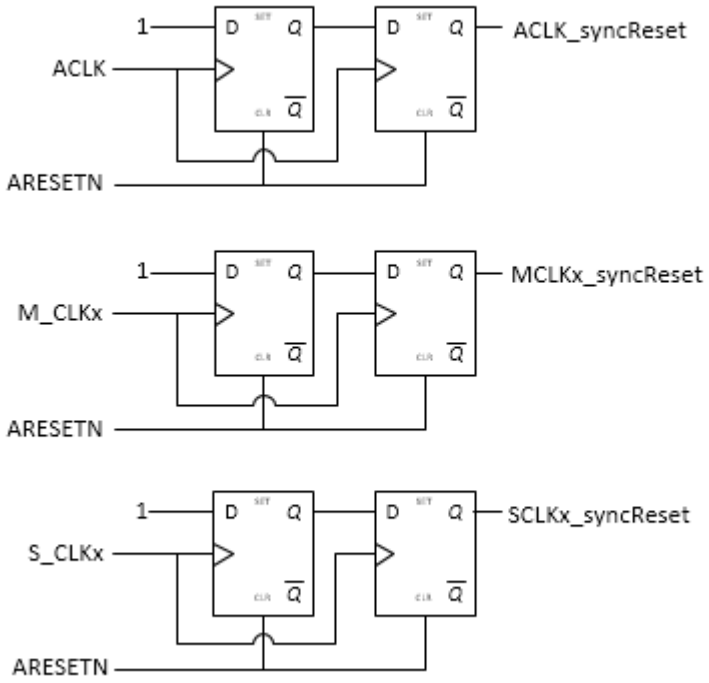
6.2 Reset

Following resets are used in the IP core.

ARESETN: Active low ARESETN is used as asynchronous reset to reset the IP core. ARESETN is synchronized in ACLK, M_CLKx (when **Mx Clock Domain Crossing** parameter is enabled and S_CLKy (when **Sy Clock Domain Crossing** parameter is enabled) domain internally.

Following figure shows the reset synchronizers:

Figure 15 • **Reset Synchronizer**



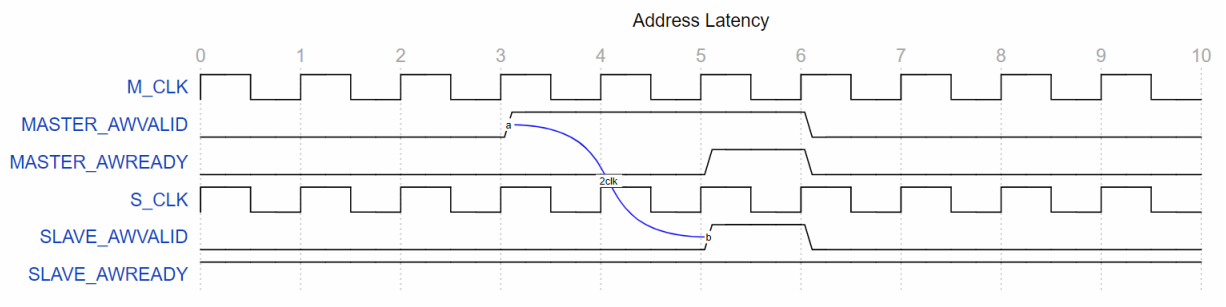
7 Timing Diagrams

The following figure shows the timing diagrams:

7.1 Write Cycles

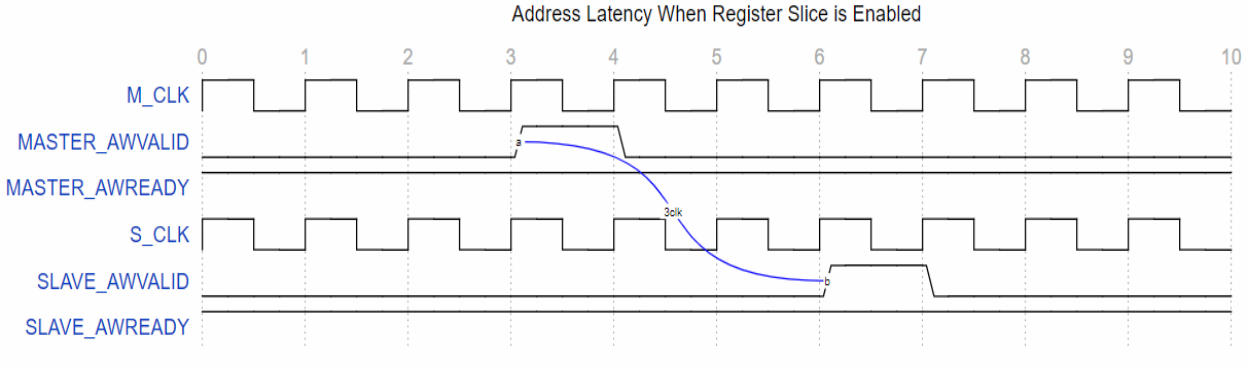
The AXI4Interconnect, with no converters, normally adds a latency of two ACLK ticks from the Master, asserting AWVALID and the target Slave device, seeing AWVALID asserted as shown in the following figure. The AWREADY is asserted with no extra latency to the Master when the Slave asserts it. The two latency cycles are required for the address arbitrator to see signal asserted and then to make a decision on which Master goes next. If the Master is not next, based on the round-robin scheme, it may have to wait additional cycles until its turn.

Figure 16 • **Address Latency**



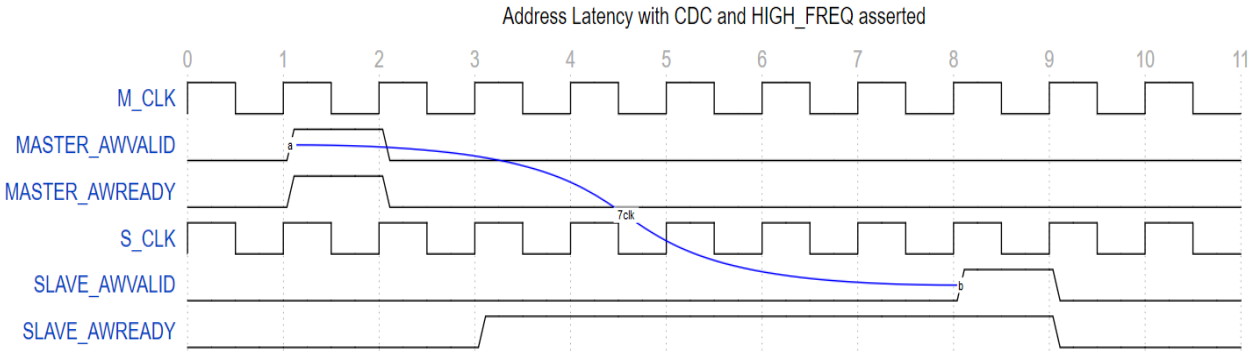
When a register slice is enabled on master side (that is, when **Mx Register Slice** parameter is enabled), it adds another latency cycle to AWVALID being asserted to the target slave as shown in the following figure. Also, AWREADY is normally asserted to Master as the RegSlice "consumes" the cycle.

Figure 17 • **Address Latency When Register Slice is Enabled**



When a register slice is enabled on slave write address channel (that is, **Sy Register Slice** parameter is enabled), another latency cycle is added to the delay in AWVALID path from Master to Slave.

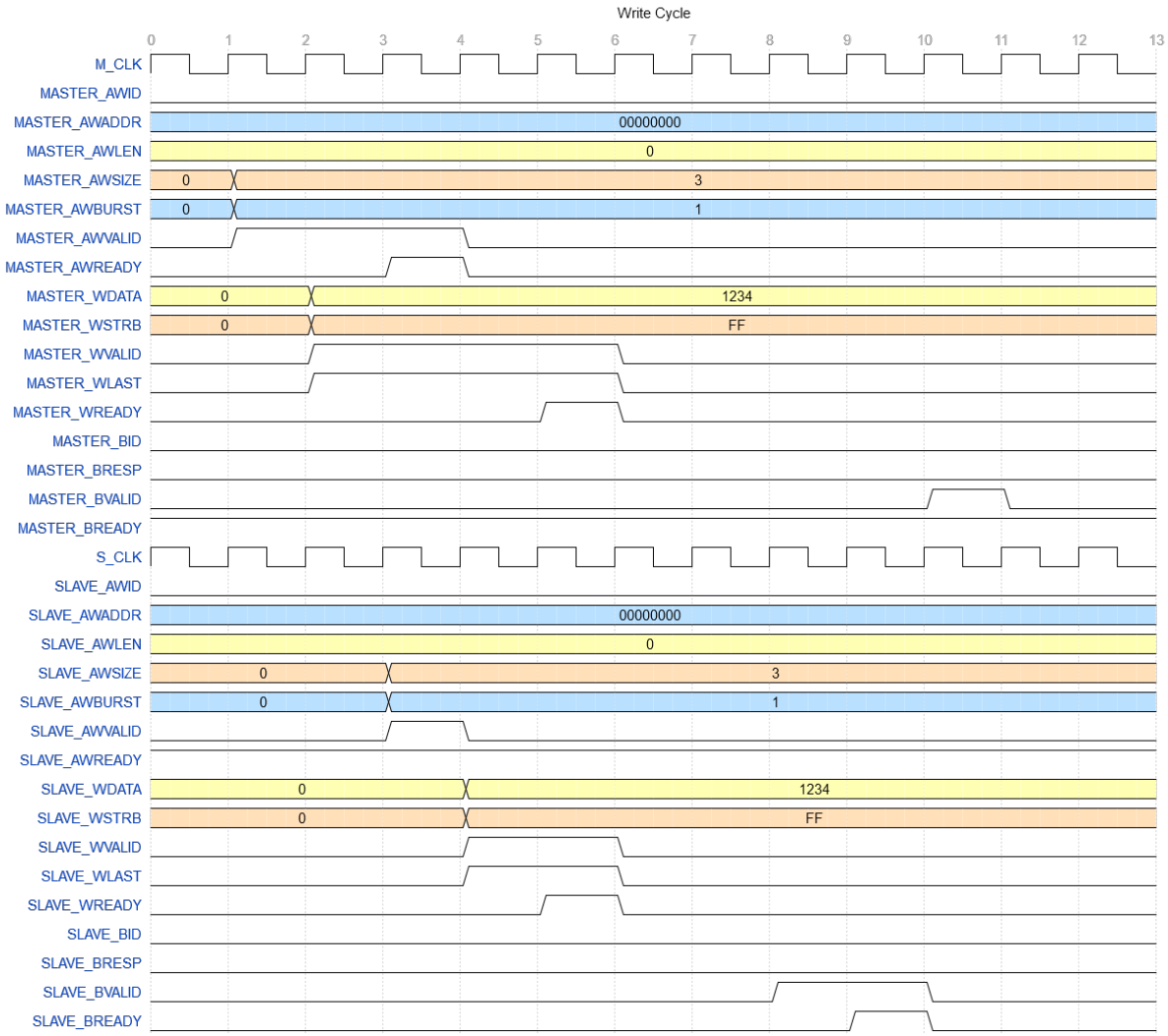
Figure 18 • **Address Latency with CDC and HIGH_FREQ asserted**



When clock domain crossing is enabled on master side (that is **Mx Clock Domain Crossing** parameter is enabled) and crossbar mode is SAMD, four additional clock cycle latency added to the delay in AWVALID path from Master to Slave. Three clock cycle latency is added due to clock domain crossing and one clock cycle latency is added due to SAMD mode.

A typical write cycle is shown in the following figure. In this example, register slices, SAMD mode of crossbar and clock domain crossings are not enabled.

Figure 19 • **Write Cycle**



7.2 Read Cycles

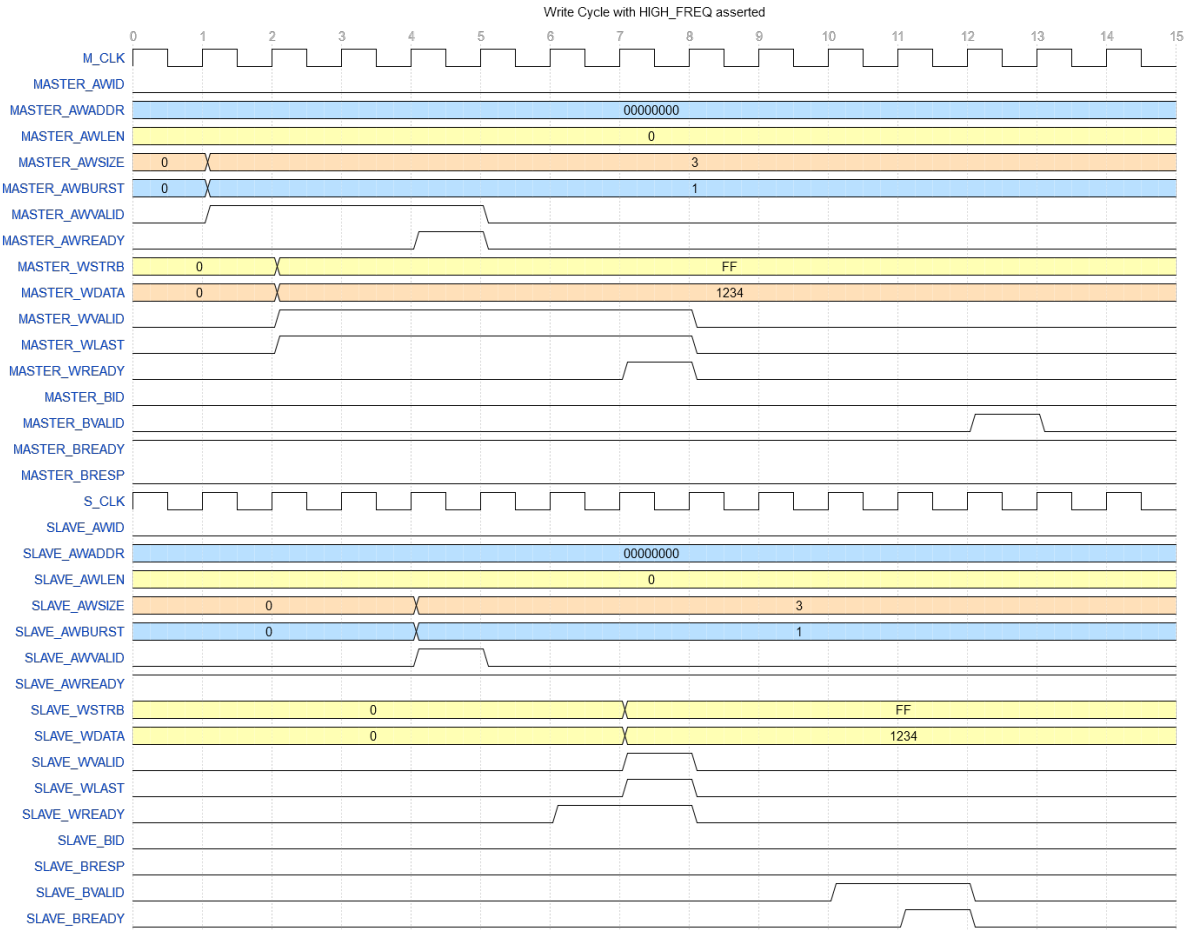
The AXI4Interconnect reads address without converters is similar to write address cycles. In general, a latency of two ACLK ticks from the Master asserting ARVALID and the target Slave device seeing ARVALID asserted. The ARREADY is asserted with no extra latency to the Master, when the Slave asserts it. The two latency cycles are required for the address arbitrator to see signal asserted and then to make a decision on which master goes next. If this Master is not next based on the round-robin scheme, it may have to wait additional cycles until its turn. The following figure shows an example of a Read Cycle. In this example, register slice is not enabled.

Figure 20 • **Read Cycle**

7.3 HI_FREQ (High Frequency)

The HI_FREQ parameter is derived internally in CoreAXI4Interconnect. HI_FREQ parameter is enabled when **CrossbarMode** parameter is set to SAMD or **Read Arbitration Enable** parameter is set to 0. It increases the maximum possible frequency of operation, at the cost of extra latency cycles. An additional cycle is added to the address paths - Address Write and Address Read, as well as another on the RESP paths. The following figure shows, how HI_FREQ changes write transactions compared to standard Write transactions (that is, when HI_FREQ=0).

Figure 21 • **Write Cycle with HIGH_FREQ asserted**

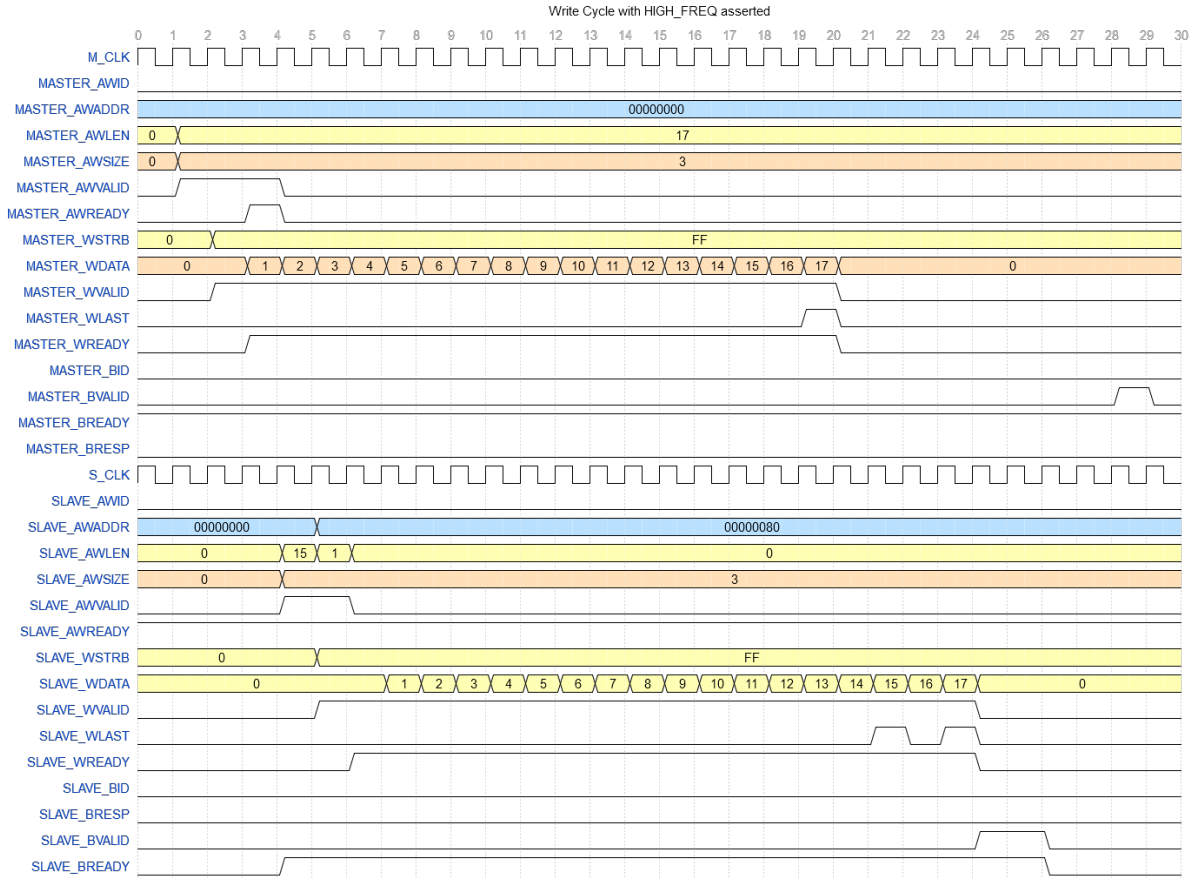


7.4 AXI3 and AXI4Lite Slave Configuration

A connected AXI3 or AXI4Lite Slave is supported by configuring SLAVE_TYPE for that port. The parameters **Slave FIFO Address Depth** and **Slave FIFO Data Depth** configures how much storage is used to hold address transactions and data transactions. Typically, both are set to 3 which means that, storage of a depth of 8 is provided to hold transactions between internal AXI4 bus and external AXI3. Typically **Slave FIFO Address Depth** should be the same value as **Max Outstanding Transactions**, to avoid holding up address transactions and **Slave FIFO Data Depth** of 3 which provides storage of depth of 8 to "buffer" Write or Read data between AXI3 slave and the internal AXI4 bus.

The following figure shows an example of a write of LEN = 17 (that is, 18 beats) from Master 0 to Slave 0. On the SLAVE0 connections this burst is broken into two - one of LEN=15 (16 beats) and a second of LEN=1 (two beats). These two bursts RESPONSES are combined into one to be sent back to MASTER0. In this example no errors are indicated in the response. In cases where an error is indicated in BRESP, all responses are combined. The BRESP returned to the connected master is set to the first error response received from the connected slave.

Figure 22 • **AXI3 Write Example**



A port configured as AXI4Lite (SLAVE_TYPE set to 2'b01 for that port) operates similar to what is shown in the preceding figure except that, all Slave side LEN will be 0. The AXI4Interconnect stores transactions' AWID/WID/ARID/RID/BID as necessary to label each transaction correctly for the Master. As with the width converters, the SLAVE ID is always sent out as "zero" to prevent read interleaving problems. Instead the Master ID is stored in a FIFO and returned to the master intact.

8 Tool Flows

8.1 License

The CoreAXI4Interconnect does not require any license.

8.2 RTL

Complete RTL source code is provided for the core and testbenches.

8.3 Smart Design

CoreAXI4Interconnect is preinstalled in the SmartDesign IP deployment design environment. An example instantiated view is shown in the following figure. The core can be configured using the configuration GUI within SmartDesign, as shown in Figure 24, page 50. For more information on using SmartDesign to instantiate and generate cores, see, **Using DirectCore in Libero SoC User Guide or consult the Libero SoC online help**.

Figure 23 • **CoreAXI4Interconnect Instance View**

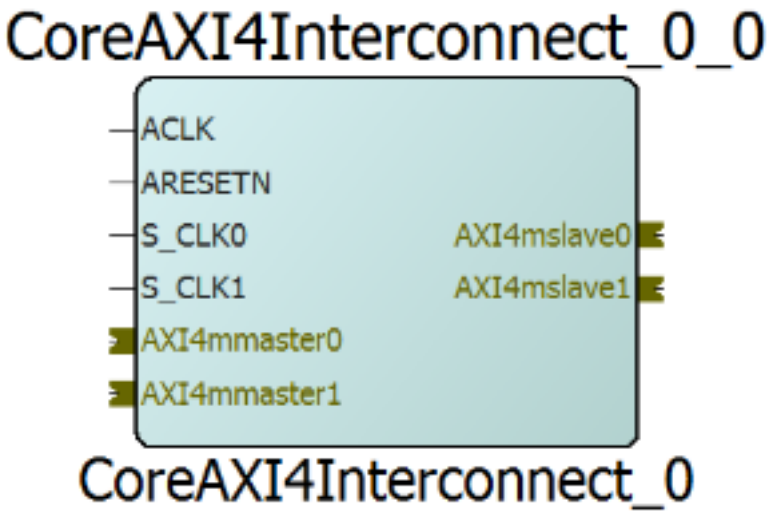
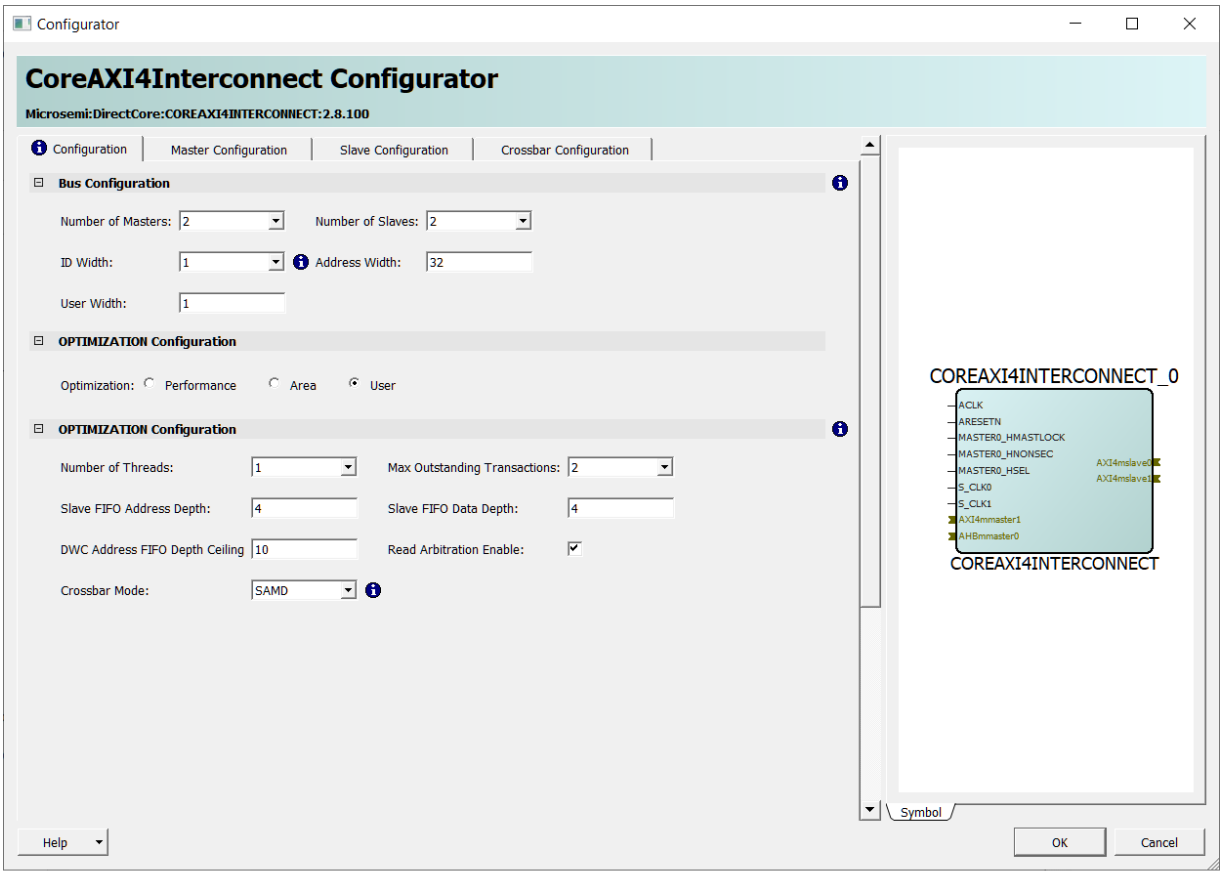


Figure 24 • **SmartDesign Configuration Window**



8.4 Simulation Flow

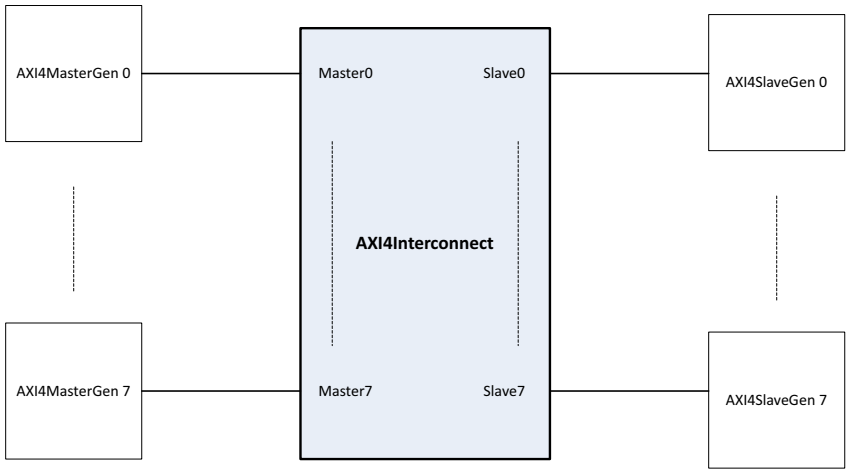
The User Testbench for AXI4Interconnect is included in all releases. To run simulations, perform the following steps.

1. To run the user testbench, set the design root to the CoreAXI4Interconnect instantiation in the Libero SoC design hierarchy pane
2. Right-click Simulate in the Libero SoC Design Flow window under Verify Pre- Synthesized Design and select Open Interactively. This invokes ModelSim and automatically runs the simulation.

8.4.1 User Testbench

The following figure shows the user simulation testbench and it includes the instantiation of the CoreAXI4Interconnect macro and multiple AXI4MasterGen and AXI4SlaveGen blocks. The AXI4MasterGen block is a primitive model of an AXI master, which performs, write and read operations, initiated by rdStart and wrStart signals within the testbench. The AXI4SlaveGen block is a primitive model of a AXI slave with embedded storage for read or write. The user tests are executed in tasks in the testbench. These tasks can be edited.

Figure 25 • **CoreAXI4Interconnect User Testbench showing 8x8 Example**



User test bench supports up to 8 Masters and 16 Slaves. It does not support outstanding transactions and more than 32 bit Address Width. As per the user test bench, each Master performs write/read operation with each slave and with different burst.

User test bench is provided for reference purpose only. It is recommended to use it with default configuration. User can find the default parameter configuration from `<project_dir/simulation/parameter_incl.v>` file. User may need to change the user test bench `<project_dir/component/Actel/DirectCore/COREAXI4INTERCONNECT/IP Core Version Number (ex 2.7.100)/sim/User_Test.v>` if user want to modify parameters in parameter_incl.v file.

Following are the details of the default configuration:

- Four Masters and Four Slaves having different types and different data width.
- Crossbar is configured in SAMD mode and its data width is 64
- Except Slave 0, all the Masters and Slaves have different clock frequency then crossbar clock i.e MASTERx_CLOCK_DOMAIN_CROSSING (where x is 0 to 3) and SLAVEy_CLOCK_DOMAIN_CROSSING (where y is 1 to 3)are configured to 1
- SLAVE address are configured in incremental order of 0x10000000 starting from 0

8.5 Synthesis in Libero SoC

After setting the design root appropriately for the design, use the following steps to run the Synthesis.

1. Click Synthesis in the Libero SoC software. The Synthesis window appears displaying the Synplicity project.
2. Set Synplicity to use the Verilog 2001 standard, if Verilog is used.
3. Click **Run**.

8.6 Place and Route

After setting the design route appropriately for the design, and running Synthesis, click Layout in the Libero SoC software to invoke Designer. CoreAXI4Interconnect does not require special place-and-route settings.

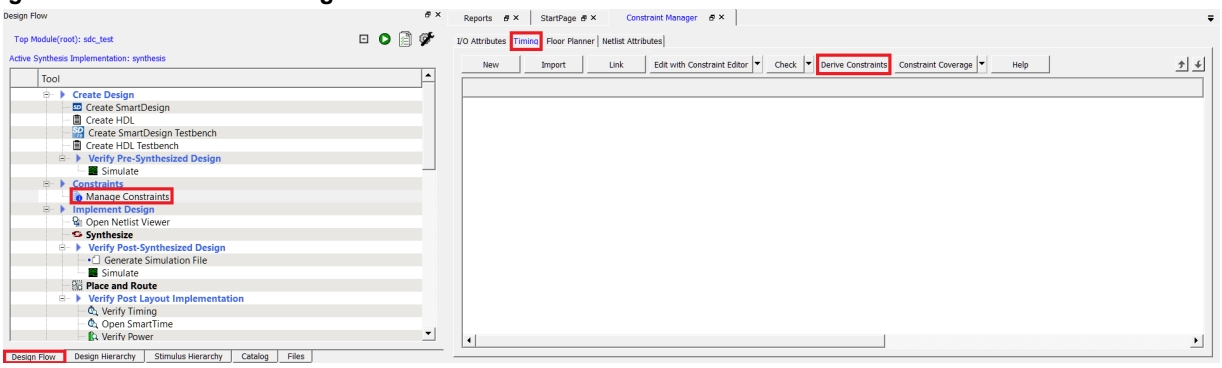
9 Design Constraints

This section describes the timing constraints of the CoreAXI4Interconnect IP core.

9.1 Timing Constraints

Asynchronous FIFO and **Reset Synchronizer** used in the core to transfer data between asynchronous clock domains and to synchronize asynchronous reset in all the clock domains respectively, requires timing constraint for synthesis, place and route, and timing verification. To generate these timing constraints, select the **Timing** tab in **Constraint Manager**, and click **Derive Constraints**, as shown in the following figure.

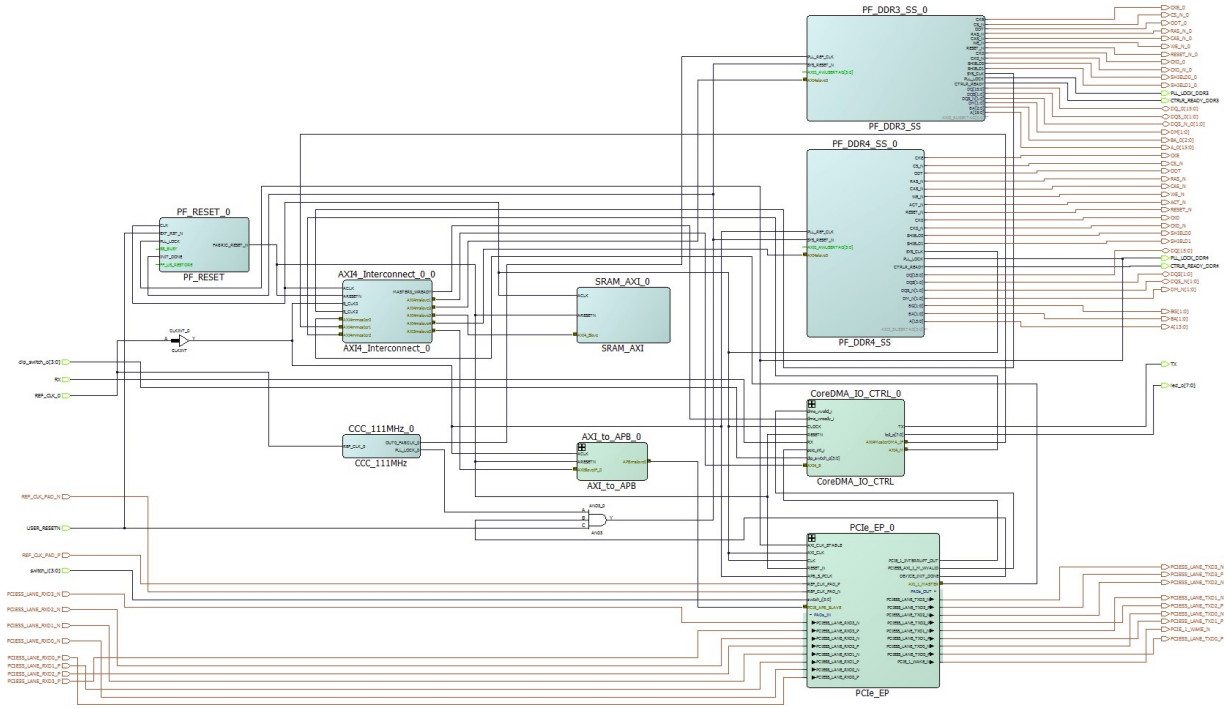
Figure 26 • **Constraint Manager—Derive Constraints Tab**



10 System Integration

This section provides an example, that shows the integration of CoreAXI4Interconnect.

Figure 27 • **CoreAXI4Interconnect Example Design**



The example design described in this section contains CoreAXI4Interconnect, which is configured for three masters and five slaves. The following are the components of the design.

- FABRIC_RESET_N of PF_RESET_0 is used for AXI4_Interconnect_0_0 reset “ARESETN”.
- The AXI4_Interconnect_0_0 has ACLK, S_CLK0 and S_CLK2 clocks. S_CLK0 and S_CLK2 are the slave0 and slave2 clocks, respectively enabled for clock domain crossing (CDC).
- ACLK is a 200 MHz clock, driven from the output port, SYS_CLK of PF_DDR4_SS_0.
- S_CLK0 is a 50MHz clock, driven from the on-board oscillator.
- S_CLK2 is a 166.665MHz clock, driven from the output port, SYS_CLK of PF_DDR3_SS_0.

Run the Libero flow by enabling the **Timing Driven, High Effort Layout,** and **Driver Replication** options. The example design can be obtained from the Microsemi technical support team.

11 Reference Documents

The following table gives the list of documents referred in this document.

Table 9 • **Reference Documents**

Document ID	Document Name
[R1]	IHI0022E - AMBA AXI and ACE Protocol Specification
[R2]	ARM_IHI0033A_AMBA_AHB-Lite_SPEC.pdf