# AXI4 Multi Port Bridge for Memory Controller Module

## User Guide

FPGA-IPUG-02246-1.0

December 2023

## Disclaimers

## Inclusive Language

This document was created consistent with Lattice Semiconductor's inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice's inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

# Contents

## Figures

# Tables

## Acronyms in This Document

A list of acronyms used in this document.

| Acronym | Definition |
|---|---|
| ACE | AXI Coherence extension |
| AHBL | Advanced High-performance Bus Lite |
| AMBA | Advanced Microcontroller Bus Architecture |
| APB | Advanced Peripheral Bus |
| AXI4 | Advanced eXtensible Interface 4 |
| CDC | Clock Domain Crossing |
| CKE | Clock enable |
| FIFO | First In First Out |
| FMAX | Maximum frequency |
| FPGA | Field Programmable Gate Array |
| GPIO | General purpose I/O |
| ID | Identification |
| LPDDR4 | Low-power Double Data Rate 4 |
| LUT | Lookup Table |
| MHz | Megahertz |
| MI | Manager Interface (Interfacing with the external subordinate) |
| MPMC | Multi Port Bridge for Memory Controller |
| ns | nanosecond |
| RSD | Reset Synchronous Deassertion |
| RTL | Register Transfer Logic |
| QoS | Quality of service |
| UART | Universal asynchronous receiver/transmitter |
| SI | Subordinate Interface (Interfacing with the external manager) |

# 1. Introduction

## 1.1. Overview of the IP

The AXI4 Multi Port Bridge for Memory Controller (MPMC) IP connects multiple external managers to a single memory controller. This IP runs only on AXI4, and supports separate clock domains between each external managers and the subordinate.

## 1.2. Quick Facts

**Table 1.1. Summary of the AXI4 MPMC IP**

| | | |
|---|---|---|
| **IP Requirements** | Supported FPGA family | Lattice CertusPro$^{TM}$-NX devices<br>Lattice Avant$^{TM}$ devices |
| | IP version | 1.0.0 |
| **Resource Utilization** | Targeted devices | Refer to the Resource Utilization section |
| | Supported user interface | Lattice Propel$^{TM}$ Builder<br>Lattice Radiant$^{TM}$ software |
| **Design Tool Support** | Lattice implementation | Lattice Radiant software |
| | Synthesis | Lattice Radiant software |
| | Simulation | Lattice Radiant software |

## 1.3. Features

Key features of the AXI4 MPMC IP include:

- Supports multiple clock domains:
  - Each subordinate interface (SI) block runs on its own clock/reset domain.
  - The manager interface (MI) block runs on its own clock/reset domain.
  - Proper clock domain crossing (CDC) mechanism is in place to ensure proper clock domain crossing between the SI and MI blocks. There are no signals crossing between each SI block.
- Supports 2 to 4 external managers and 1 external subordinate.
- Performs width conversion (upsizing) only.
- Supports the round robin arbitration scheme.
- All resets have the deassertion edge synchronized to the respective clock edge with an internal reset synchronous deassertion (RSD) block.

## 1.4. Licensing and Ordering Information

The AXI4 MPMC IP does not require license string as the IP is provided at no additional cost with the Lattice Radiant software.

## 1.5. IP Validation Summary

The AXI4 MPMC IP is validated with the following synthesis tools:

- Lattice Radiant software 2023.1
- Lattice Radiant software 2023.2
- Lattice Propel Builder 2023.2

Table 1.2 shows the validation status for the AXI4 MPMC IP core. The ✓ mark indicates whether the IP has been validated for Simulation, Timing, or with Hardware.

**Table 1.2. IP Validation Level**

| Device Family | IP Version | Validation Level | | |
|---|---|---|---|---|
| | | Simulation | Timing | Hardware |
| CertusPro-NX devices | 1.0.0 | ✓ | ✓ | ✓ |
| Avant devices | 1.0.0 | ✓ | ✓ | TBD |

## 1.6. Naming Conventions

### 1.6.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

### 1.6.2. Signal Names

- *si_* are signals interfacing with external manager
- *mi_* are signals interfacing with external subordinate
- *_i* are input signals
- *_o* are output signals

# 2. Functional Description

## 2.1. IP Architecture Overview

Figure 2.1 shows the block diagram for the Lattice AXI4 MPMC IP.

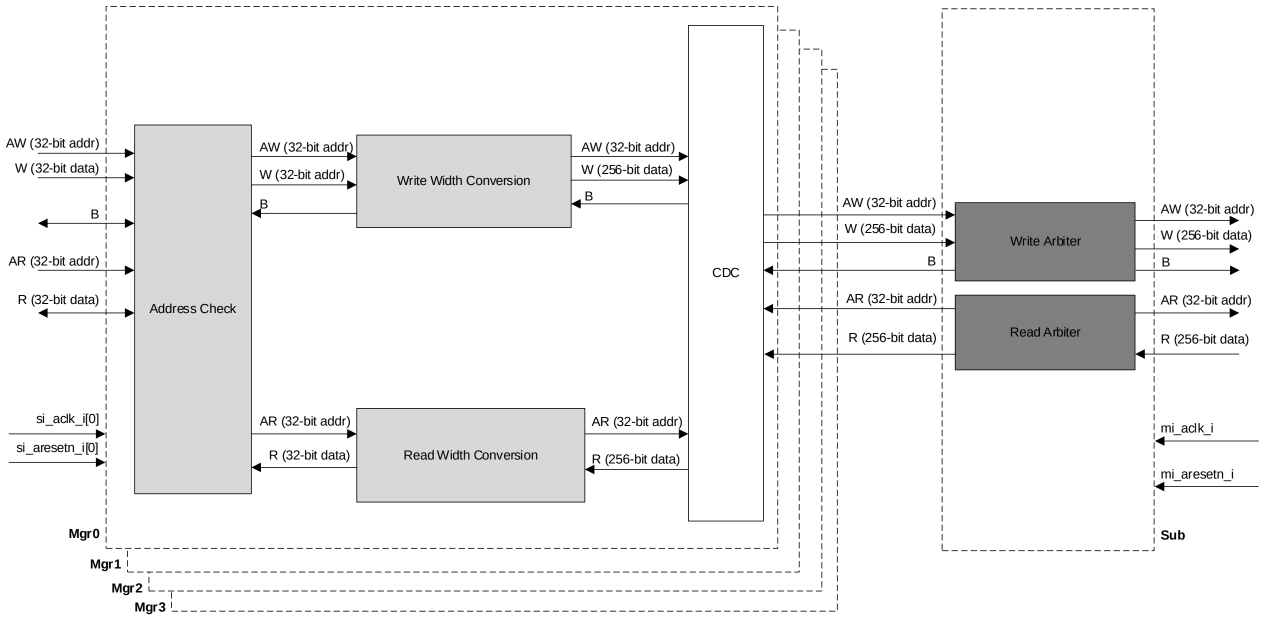


**Figure 2.1. Lattice AXI4 MPMC IP Core Block Diagram**

The AXI4 MPMC IP is divided into three sections:

- SI block: This block is instantiated per external manager, and implements the width conversion logic for both write and read transactions.
- MI block: This block is instantiated once, and interfaces with the subordinate. This block implements the arbitration scheme.
- CDC block: This block consists of dual clock asynchronous FIFOs and ensures proper clock domain crossing handling. If CDC is disabled, single clock synchronous FIFOs are instantiated.

## 2.2. Clocking

The different clock domains are indicated by different colors, as shown in Figure 2.1. Mgr0, Mgr1, Mgr2, and Mgr3 each has a dedicated clock, si_aclk_i[0:3]. Each si_aclk_i[*] clock goes to the external manager that connects to the respective SI block. The arbiter section runs on the mi_aclk_i clock domain which goes to the subordinate.

### 2.2.1. Clocking Overview

The clocks support the following frequency ranges:

- Up to 140 MHz (CertusPro-NX devices) or 260 MHz (Avant devices) for si_aclk_i[*]
- Up to 180 MHz (CertusPro-NX devices) or 300 MHz (Avant devices) for mi_aclk_i

Each clock is independent of each other. The clocks are on the same clock domain with the respective external managers or subordinate that the clocks interface with.

## 2.3. Reset

The different reset domains are indicated by different colors, as shown in Figure 2.1. Mgr0, Mgr1, Mgr2, and Mgr3 each has a dedicated reset, si_aresetn_i[0:3]. Each si_aresetn_i[*] reset goes to the external manager that connects to the respective SI block. The arbiter section runs on the mi_aresetn_i reset domain which goes to the subordinate.

### 2.3.1. Reset Overview

Each reset is independent of each other. There are no relations between them. Each of the reset has the deassertion edge synchronized to the clock. If one reset asserts, the others must be asserted too, regardless of the assertion sequence. The entire IP must be reset if one or more reset signal is asserted.

## 2.4. User Interfaces

Table 2.1 shows the interfaces for the Lattice AXI4 MPMC IP.

**Table 2.1. User Interfaces and Supported Protocols**

| User Interface | Supported Protocols | Description |
|---|---|---|
| Subordinate interface (SI) | AXI4 | This interface connects to an external AXI4 manager. The IP has at least two subordinate interfaces. |
| Manager interface (MI) | AXI4 | This interface connects to an external AXI4 subordinate. The IP has only one manager interface. |

## 2.5. Blocks and Functions

Table 2.2 shows the functions of each block in the Lattice AXI4 MPMC IP.

**Table 2.2. Functions of the Blocks in the AXI4 MPMC IP**

| Block | Sub Block | Description |
|---|---|---|
| SI block | Width conversion | • Write width converter: This block upsizes the write data bus.<br>• Read width converter: This block upsizes the read data bus. |
| | CDC | This block handles the clock domain crossing to and from the MI block. |
| MI block | Arbitration | • Write arbiter: This block performs arbitration on the write channels: Write Address (AW), Write Data (W) and Write Response (B).<br>• Read arbiter: This block performs arbitration on the read channels: Read Address (AR) and Read Response (R). |

# 3. IP Parameter Description

The configurable attributes of the AXI4 Multi Port Bridge for Memory Controller IP are shown in Table 3.1 and Table 3.2. You can configure the IP by setting the attributes in the IP Catalog Module or IP wizard of the Lattice Radiant software.

Wherever applicable, default values are in bold.

## 3.1. General

Table 3.1 shows the general attributes of the AXI MPMC IP.

**Table 3.1. General Attributes**

| Attribute | Selectable Values | Parameter Name | Description |
|---|---|---|---|
| Total External AXI4 Managers | **2**<br>3<br>4 | TOTAL_MGR_COUNT_TOP | Configures the number of external managers connected to the IP. This value determines the number of SI block that is instantiated internally. |
| Manager and Subordinate Address Width | 32 | ADDR_WIDTH_TOP | Configures the allowable address bits for all the SI and MI address ports interfacing with external managers and subordinate.<br>*Grayed out – non-configurable.* |
| Manager Maximum Data Width | 32 | SI_MAX_DATA_WIDTH_TOP | Configures the maximum allowable data bits for all the SI data ports interfacing with the external managers.<br>*Grayed out – non-configurable.* |
| Manager and Subordinate ID Width | 2<br>3<br>4<br>5<br>**6**<br>7<br>8 | ID_WIDTH_TOP | Configures the allowable ID bits for all SI and MI ID ports interfacing with the external managers and subordinate. |
| Subordinate Data Width | 256 | MI_DATA_WIDTH_TOP | Configures the allowable data bits for the MI data port interfacing with the external subordinate.<br>*Grayed out – non-configurable.* |
| Maximum Supported Outstanding Transactions | 4<br>5<br>6<br>7<br>**8**<br>9<br>10<br>11<br>12<br>13<br>14<br>15<br>16 | MAX_SUPPORTED_OUTSTANDING_TRANSACTIONS_TOP | Configures the maximum allowable number of outstanding transactions.<br>*Grayed out – non-configurable.* |
| Arbitration Scheme | RoundRobin | ARBITER_TYPE_TOP | Configures the type of arbitration scheme.<br>*Grayed out – non-configurable.* |

## 3.2. External Manager Settings

Table 3.2 shows the external manager settings attributes of the AXI MPMC IP.

**Table 3.2. External Manager Settings Attributes**

| **External Manager Outstanding Limit Settings** | | | |
|---|---|---|---|
| **Attribute** | **Selectable Values** | **Parameter Name** | **Description** |
| Outstanding Limit Manager N | 4<br>5<br>6<br>7<br>**8**<br>9<br>10<br>11<br>12<br>13<br>14<br>15<br>16 | OUTSTANDING_LIMIT_TOP | • Configures the depth of the internal buffer that holds the outstanding transactions of the SI block interfacing with external manager.<br>• N can be 0, 1, 2, and 3 depending on the value of the Total External AXI4 Managers attribute. |
| **External Manager CDC Enable Settings** | | | |
| **Attribute** | **Selectable Values** | **Parameter Name** | **Description** |
| CDC Enable Manager N | **DISABLED**<br>ENABLED | ENABLE_SI_CDC_TOP | • Enables or disables CDC for the SI block interfacing with external manager.<br>• N can be 0, 1, 2, and 3 depending on the value of the Total External AXI4 Managers attribute. |
| **External Manager Data Width Settings** | | | |
| **Attribute** | **Selectable Values** | **Parameter Name** | **Description** |
| Data Width Manager N | 32 | SI_DATA_WIDTH_TOP | • Configures the allowable data bits for the SI data port interfacing with external manager.<br>• N can be 0, 1, 2, and 3 depending on the value of the Total External AXI4 Managers attribute.<br>• *Grayed out – non-configurable* |
| **External Manager Access Mode Settings** | | | |
| **Attribute** | **Selectable Values** | **Parameter Name** | **Description** |
| Access Mode Manager N | **Read_and_Write**<br>Write_Only<br>Read_Only | ACCESS_MODE_TOP | • Configures the access mode for the SI block interfacing with external manager.<br>• N can be 0, 1, 2, and 3 depending on the value of the Total External AXI4 Managers attribute. |

# 4. Signal Description

This section describes the AXI4 Multi Port Bridge for Memory Controller IP ports.

## 4.1. Clock Interface

Table 4.1 shows the clock ports of the AXI MPMC IP.

**Table 4.1. Clock Ports**

| Port | Type | Description |
|---|---|---|
| si_aclk_i | Input | • This is a vector with the number of elements corresponding to the number of external managers as specified by the "Total External AXI4 Managers" attribute.<br>• Each element in si_aclk_i is a clock domain, and do not interact with each other.<br>• Each si_aclk_i element drives a separate SI block and interfaces with a separate external manager. |
| mi_aclk_i | Input | • This is the MI clock domain.<br>• This is the clock that interfaces with the external subordinate.<br>• All transactions cross between the si_aclk_i domains and the mi_aclk_i domain.<br>• A FIFO interfaces between each SI block and MI block to ensure safe clock domain crossings. |

## 4.2. Reset Interface

Table 4.2 shows the reset ports of the AXI MPMC IP.

**Table 4.2. Reset Ports**

| Port | Type | Description |
|---|---|---|
| si_aresetn_i | Input | • This is a vector with the number of elements corresponding to the number of external managers as specified by the "Total External AXI4 Managers" attribute.<br>• The si_aresetn_i ports are asynchronous resets, and each element belongs to an SI block that interfaces with a separate external manager.<br>• The deassertion of the resets are synchronously driven internally, and are synchronous to the respective si_aclk_i. For example, the deassertion of si_aresetn_i[0] is synchronous to si_aclk_i[0].<br>• An internal logic handles the resets. No external RSD logic is required.<br>• As the IP handles the reset, it is recommended that the reset connected to this port is not synchronously deasserted externally. If the reset connecting to this port is synchronously deasserted, the actual deassertion internally has an additional of two clock latency. |
| mi_aresetn_i | Input | • The mi_aresetn_i is an asynchronous reset, and belong to the MI block, interfacing with the external subordinate.<br>• The deassertion of this reset is synchronously driven internally, being synchronous to mi_aclk_i.<br>• An internal logic handles the resets. No external RSD logic is required.<br>• As the IP handles the reset, it is recommended that the reset connected to this port is not synchronously deasserted externally. If the reset connecting to this port is synchronously deasserted, the actual deassertion internally has an additional of two clock latency. |

## 4.3. SI Block

All the ports in the SI block are vectors, with the number of elements corresponding to the number of external managers, as specified by the "Total External AXI4 Managers" attribute. Each element belongs to a separate SI block, interfacing with a separate external manager. Table 4.3 lists the AXI ports that comply with the AXI standard as specified in the IHI0022E (AMBA AXI and ACE Protocol Specification) document. All signals are mandatory and supported unless otherwise specified.

**Table 4.3. SI Ports**

| Port | Type | Description |
|---|---|---|
| si_awvalid_i | Input | Write address valid. Indicates that the write address group of signals have valid data. |
| si_awid_i | Input | Write ID. Holds the ID for the write transaction. |
| si_awaddr_i | Input | Write address. Holds the address offset for the first transfer in a burst transaction. |
| si_awsize_i | Input | Write burst size. Holds the transfer size for the transaction. |
| si_awprot_i | Input | Write protection type. Determines if this transaction is a data access or instruction access, and the privilege and security level.<br>*Optional – Not supported. Direct passthrough.* |
| si_awlen_i | Input | Write burst length. Holds the number of beats for a burst transaction. |
| si_awburst_i | Input | Write burst type. Holds the burst information: FIXED, INCR, or WRAP. This IP only supports INCR. |
| si_awcache_i | Input | Write memory type. Describes the progress of the transaction through a system.<br>*Optional – Not supported. Direct passthrough.* |
| si_awlock_i | Input | Write lock type. Holds information on the atomic characteristics of the transfer.<br>*Optional – Not supported. Direct passthrough.* |
| si_awqos_i | Input | Write quality of service. Holds the quality of service (QoS) identifier for the transaction.<br>*Optional – Not supported. Direct passthrough.* |
| si_awregion_i | Input | Write region identifier. Describes additional signaling required to support multiple region interfaces.<br>*Optional – Not supported. Direct passthrough.* |
| si_awuser_i | Input | Write user signal. User-defined signal for the write address transaction.<br>*Optional – Not supported. Direct passthrough.* |
| si_awready_o | Output | Write address ready. Indicates that the interface is ready to receive incoming traffic. Stays asserted until valid asserts. |
| si_wvalid_i | Input | Write data valid. Indicates that the write data group of signals have valid data. |
| si_wdata_i | Input | Write data. Holds the data to be transferred. |
| si_wstrb_i | Input | Write strobe. Holds the strobe that indicates which byte lanes has valid data. |
| si_wuser_i | Input | Write user signal. User-defined signal for the write data transaction.<br>*Optional – Not supported. Direct passthrough.* |
| si_wlast_i | Input | Write last. Indicates that the current transfer is the final beat of the transaction. |
| si_wready_o | Output | Write data ready. Indicates that the interface is ready to receive incoming traffic. Stays asserted until valid asserts. |
| si_bvalid_o | Output | Write response valid. Indicates that the write response group of signals have valid data. Stays asserted until ready asserts. |
| si_bid_o | Output | Write response ID. Holds the ID for the write transaction. |
| si_bresp_o | Output | Write response. Holds the status of the write transaction. |
| si_buser_o | Output | Write user signal. User-defined signal for the write response transaction.<br>*Optional – Not supported. Direct passthrough.* |
| si_bready_i | Input | Write response ready. Indicates that the outgoing traffic has been received. |
| si_arvalid_i | Input | Read address valid. Indicates that the read address group of signals have valid data. |

| Port | Type | Description |
|---|---|---|
| si_arid_i | Input | Read ID. Holds the ID for the read transaction. |
| si_araddr_i | Input | Read address. Holds the address offset for the first transfer in a burst transaction. |
| si_arsize_i | Input | Read burst size. Holds the transfer size for the transaction. |
| si_arprot_i | Input | Read protection type. Determines if this transaction is a data access or instruction access, and the privilege and security level.<br>*Optional – Not supported. Direct passthrough.* |
| si_arlen_i | Input | Read burst length. Holds the number of beats for a burst transaction. |
| si_arburst_i | Input | Read burst type. Holds the burst information: FIXED, INCR, or WRAP. This IP only supports INCR. |
| si_arcache_i | Input | Read memory type. Describes the progress of the transaction through a system.<br>*Optional – Not supported. Direct passthrough.* |
| si_arlock_i | Input | Read lock type. Holds information on the atomic characteristics of the transfer.<br>*Optional – Not supported. Direct passthrough.* |
| si_arqos_i | Input | Read quality of service. Holds the QoS identifier for the transaction.<br>*Optional – Not supported. Direct passthrough.* |
| si_arregion_i | Input | Read region identifier. Describes additional signaling required to support multiple region interfaces.<br>*Optional – Not supported. Direct passthrough.* |
| si_aruser_i | Input | Read user signal. User-defined signal for the read address transaction.<br>*Optional – Not supported. Direct passthrough.* |
| si_arready_o | Output | Read address ready. Indicates that the interface is ready to receive incoming traffic. Stays asserted until valid asserts. |
| si_rvalid_o | Output | Read data valid. Indicates that the read data group of signals have valid data. Stays asserted until ready asserts. |
| si_rid_o | Output | Read response ID. Holds the ID for the read transaction. |
| si_rdata_o | Output | Read data. Holds the data for the read transaction. |
| si_rresp_o | Output | Read response. Holds the status of the read transaction. |
| si_ruser_o | Output | Read user signal. User-defined signal for the read response transaction.<br>*Optional – Not supported. Direct passthrough.* |
| si_rlast_o | Output | Read last. Indicates that the current transfer is the final beat of the transaction. |
| si_rready_i | Input | Read response ready. Indicates that the outgoing traffic has been received. |

## 4.4. MI Block

Table 4.4 lists the AXI ports that comply with the AXI standard as specified in the IHI0022E (AMBA AXI and ACE Protocol Specification) document. All signals are mandatory and supported unless otherwise specified.

**Table 4.4. MI Ports**

| Port | Type | Description |
|---|---|---|
| mi_awvalid_o | Output | Write address valid. Indicates that the write address group of signals have valid data. Stays asserted until ready asserts. |
| mi_awid_o | Output | Write ID. Holds the ID for the write transaction. |
| mi_awaddr_o | Output | Write address. Holds the address offset for the first transfer in a burst transaction. |
| mi_awsize_o | Output | Write burst size. Holds the transfer size for the transaction. |
| mi_awprot_o | Output | Write protection type. Determines if this transaction is a data access or instruction access, and the privilege and security level.<br>*Optional – Not supported. Direct passthrough.* |
| mi_awlen_o | Output | Write burst length. Holds the number of beats for a burst transaction. |

| Port | Type | Description |
|---|---|---|
| mi_awburst_o | Output | Write burst type. Holds the burst information: FIXED, INCR, or WRAP. This IP only supports INCR. |
| mi_awcache_o | Output | Write memory type. Describes the progress of the transaction through a system. *Optional – Not supported. Direct passthrough.* |
| mi_awlock_o | Output | Write lock type. Holds information on the atomic characteristics of the transfer. *Optional – Not supported. Direct passthrough.* |
| mi_awqos_o | Output | Write quality of service. Holds the QoS identifier for the transaction. *Optional – Not supported. Direct passthrough.* |
| mi_awregion_o | Output | Write region identifier. Describes additional signaling required to support multiple region interfaces. *Optional – Not supported. Direct passthrough.* |
| mi_awuser_o | Output | Write user signal. User-defined signal for the write address transaction. *Optional – Not supported. Direct passthrough.* |
| mi_awready_i | Input | Write address ready. Indicates that the external subordinate is ready to receive the transaction. |
| mi_wvalid_o | Output | Write data valid. Indicates that the write data group of signals have valid data. Stays asserted until ready asserts. |
| mi_wdata_o | Output | Write data. Holds the data to be transferred. |
| mi_wstrb_o | Output | Write strobe. Holds the strobe that indicates which byte lanes has valid data. |
| mi_wuser_o | Output | Write user signal. User-defined signal for the write data transaction. *Optional – Not supported. Direct passthrough.* |
| mi_wlast_o | Output | Write last. Indicates that the current transfer is the final beat of the transaction. |
| mi_wready_i | Input | Write data ready. Indicates that the external subordinate is ready to receive the transaction. |
| mi_bvalid_i | Input | Write response valid. Indicates that the write response group of signals have valid data. |
| mi_bid_i | Input | Write response ID. Holds the ID for the write transaction. |
| mi_bresp_i | Input | Write response. Holds the status of the write transaction. |
| mi_buser_i | Input | Write user signal. User-defined signal for the write response transaction. *Optional – Not supported. Direct passthrough.* |
| mi_bready_o | Output | Write response ready. Indicates that the interface is ready to receive incoming traffic. Stays asserted until valid asserts. |
| mi_arvalid_o | Output | Read address valid. Indicates that the read address group of signals have valid data. Stays asserted until valid asserts. |
| mi_arid_o | Output | Read ID. Holds the ID for the read transaction. |
| mi_araddr_o | Output | Read address. Holds the address offset for the first transfer in a burst transaction. |
| mi_arsize_o | Output | Read burst size. Holds the transfer size for the transaction. |
| mi_arprot_o | Output | Read protection type. Determines if this transaction is a data access or instruction access, and the privilege and security level. *Optional – Not supported. Direct passthrough.* |
| mi_arlen_o | Output | Read burst length. Holds the number of beats for a burst transaction. |
| mi_arburst_o | Output | Read burst type. Holds the burst information: FIXED, INCR, or WRAP. This IP only supports INCR. |
| mi_arcache_o | Output | Read memory type. Describes the progress of the transaction through a system. *Optional – Not supported. Direct passthrough.* |
| mi_arlock_o | Output | Read lock type. Holds information on the atomic characteristics of the transfer. *Optional – Not supported. Direct passthrough.* |
| mi_arqos_o | Output | Read quality of service. Holds the QoS identifier for the transaction. *Optional – Not supported. Direct passthrough.* |

| Port | Type | Description |
|---|---|---|
| mi_arregion_o | Input | Read region identifier. Describes additional signaling required to support multiple region interfaces.<br>*Optional – Not supported. Direct passthrough.* |
| mi_aruser_o | Output | Read user signal. User-defined signal for the read address transaction.<br>*Optional – Not supported. Direct passthrough.* |
| mi_arready_i | Input | Read address ready. Indicates that the outgoing traffic has been received. |
| mi_rvalid_i | Input | Read data valid. Indicates that the read data group of signals have valid data. |
| mi_rid_i | Input | Read response ID. Holds the ID for the read transaction. |
| mi_rdata_i | Input | Read data. Holds the data for the read transaction. |
| mi_rresp_i | Input | Read response. Holds the status of the read transaction. |
| mi_ruser_i | Input | Read user signal. User-defined signal for the read response transaction.<br>*Optional – Not supported. Direct passthrough.* |
| mi_rlast_i | Input | Read last. Indicates that the current transfer is the final beat of the transaction. |
| mi_rready_o | Output | Read response ready. Indicates that the interface is ready to receive incoming traffic. Stays asserted until valid asserts. |

# 5. Example Design

The example design for this IP showcases the behavior of the AXI4 MPMC IP using an LPDDR4 memory controller. This IP supports two instances of the LPDDR4 memory controller, one for Lattice CertusPro-NX devices and the other for Lattice Avant devices.

## 5.1. Example Design Supported Configuration

**Note:** In the table below, ✓ refers to a checked option in the AXI4 MPMC IP example design and X refers to an unchecked option or a non-applicable option in the AXI4 MPMC IP example design.

**Table 5.1. AXI4 MPMC IP Configuration Supported by the Example Design**

| AXI4 MPMC IP GUI Parameter | AXI4 MPMC IP Configuration Supported in Example Demo Design |
|---|---|
| Total External AXI4 Managers | ✓ |
| Manager and Subordinate ID Width | ✓ |
| Maximum Supported Outstanding Transactions | ✓ |
| CDC Enable Manager 0:3 | ✓ |
| Outstanding Limit Manager 0:3 | ✓ |
| Access Mode Manager 0:3 | X (Only Read_and_Write is supported) |

## 5.2. Overview of the Example Design and Features

Key features of the example design include:

- Synthesizable example design that runs on simulation.
- Supports multiple external managers. The AXI4 MPMC IP design supports up to 4 external managers.
- Uses generic AXI traffic generator to provide random stimulus.
- Uses the actual LPDDR4 Memory Controller IP (statically generated) as the external subordinate.
- Runs on the CertusPro-NX Versa Evaluation Board.

Simulation runs on both CertusPro-NX and Avant devices, while synthesized design is validated only on CertusPro-NX devices.

## 5.3. Example Design Components

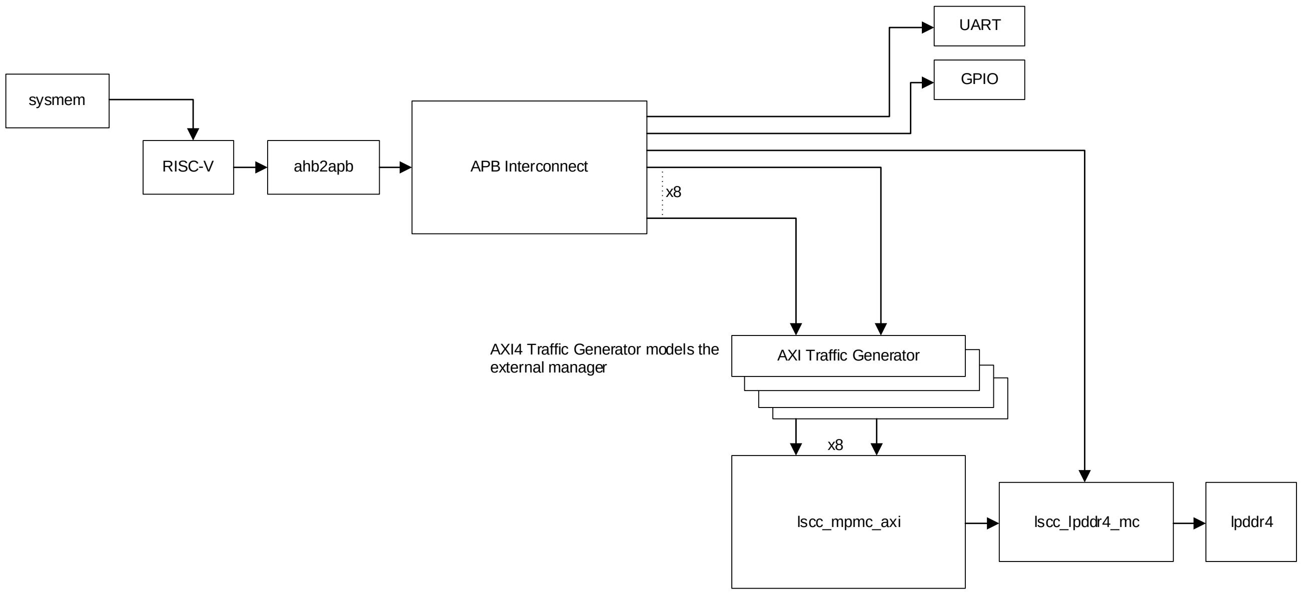


**Figure 5.1. AXI4 MPMC IP Example Design Block Diagram**

The AXI4 MPMC IP example design includes the following blocks, as shown in Figure 5.1:

- System memory and RISC-V
- AHBL to APB bridge and APB interconnect
- UART IP and GPIO IP
- AXI4 traffic generator
- AXI4 MPMC IP
- LPDDR4 Memory Controller IP

## 5.4. System Memory and RISC-V

The system memory holds the precompiled firmware, which is the source of the program flow. There are 4 instances of the system memory. As the firmware code differs, each system memory instance is for different number of external managers. These firmware runs on the RISC-V CPU. On startup, the RISC-V CPU fetches the precompiled firmware from the system memory and loads the firmware into the internal instruction memory before starting the sequence.

## 5.5. AHBL to APB Bridge and APB Interconnect

The bus infrastructure that provides the necessary protocol conversion from AHBL to APB and the arbitration logic needed to handle multiple APB subordinates.

## 5.6. UART IP and GPIO IP

These two peripherals provide interactive status on the example design flow. UART provides a means to display messages on the console. GPIO is used to toggle the on-board LEDs on or off, depending on test status.

## 5.7. AXI4 Traffic Generator

This generic instance generates random AXI4 transactions. This instance serves as the external AXI4 managers to the AXI4 MPMC IP. There is one instance of the AXI4 traffic generator per external manager. This IP supports a minimum of 2 external managers and maximum of 4 external managers.

## 5.8. AXI4 MPMC IP

This IP receives transactions from multiple AXI4 traffic generator instances and routes the transaction to the LPDDR4 Memory Controller IP.

## 5.9. LPDDR4 Memory Controller IP

This LPDDR4 Memory Controller IP is statically generated with default configuration. This example design supports both Lattice CertusPro-NX and Lattice Avant devices using two different instances of the LPDDR4 memory controller. Depending on device type, the corresponding LPDDR4 Memory Controller IP is instantiated. The IP version supported for the devices:

- Lattice CertusPro-NX devices: LPDDR4 Memory Controller for Nexus, version 2.1.0
- Lattice Avant devices: Memory Controller for Avant Devices, version 1.3.0

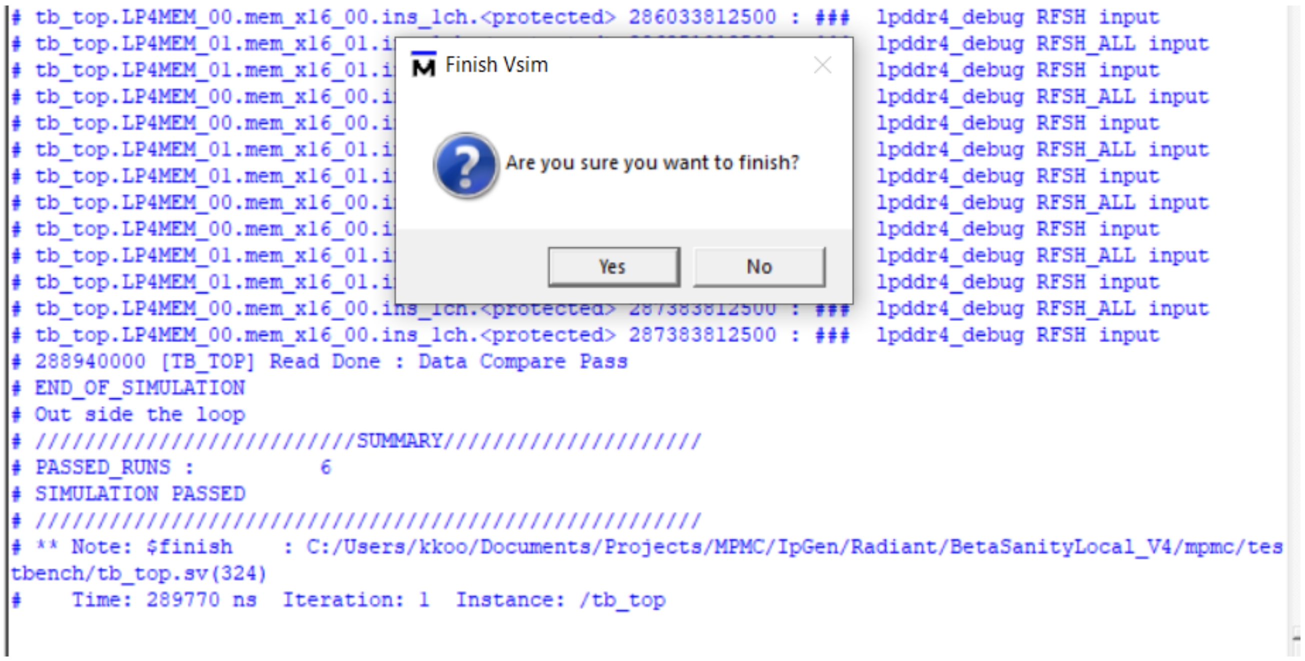
## 5.10. Simulating the Example Design

Simulating the example design is similar to synthesizing the example design, with the following changes:

- Disables UART connection to reduce the simulation time.
- External LPDDR4 memory is replaced with LPDDR4 memory model.

To simulate the example design, follow these steps:

1. Launch the Lattice Radiant software and install the latest AXI4 MPMC IP from the IP Catalog.
2. Double click AXI4 MPMC to configure the IP for your design. Select Read_and_Write for the access mode configuration for all external managers.
3. Navigate to the File List tab, right click **impl_1**, and select **Add > Existing File**.
4. Navigate to the eval folder and select *eval_top.sv*. The *eval_top.sv* file is automatically detected as the top-level instance.
5. Right click **impl_1** again and select **Add > Existing Simulation File**.
6. Navigate to the testbench folder and select *tb_top.sv*.
7. Click **Simulation Wizard** to launch the Simulation Wizard window. Follow through the instructions, click **Next** on each step. Simulation Top Module is automatically assigned tb_top. Check that there are three source files:
   - tb_top.sv
   - eval_top.sv
   - AXI4 MPMC IP
8. Adjust the simulation time as needed. The simulation takes more than 300 µs of simulation time to complete (approximately 50 minutes).
9. To begin the simulation, click **Finish**.
10. When simulation completes successfully, a "Simulation Passed" message is displayed, as shown in Figure 5.2.



```
# tb_top.LP4MEM_00.mem_x16_00.ins_lch.<protected> 286033812500 : ###  lpddr4_debug RFSH input
# tb_top.LP4MEM_01.mem_x16_01.i                                       lpddr4_debug RFSH_ALL input
# tb_top.LP4MEM_01.mem_x16_01.i                                       lpddr4_debug RFSH input
# tb_top.LP4MEM_00.mem_x16_00.i                                       lpddr4_debug RFSH_ALL input
# tb_top.LP4MEM_00.mem_x16_00.i                                       lpddr4_debug RFSH input
# tb_top.LP4MEM_01.mem_x16_01.i                                       lpddr4_debug RFSH_ALL input
# tb_top.LP4MEM_01.mem_x16_01.i                                       lpddr4_debug RFSH input
# tb_top.LP4MEM_00.mem_x16_00.i                                       lpddr4_debug RFSH_ALL input
# tb_top.LP4MEM_00.mem_x16_00.i                                       lpddr4_debug RFSH input
# tb_top.LP4MEM_01.mem_x16_01.i                                       lpddr4_debug RFSH_ALL input
# tb_top.LP4MEM_01.mem_x16_01.i                                       lpddr4_debug RFSH input
# tb_top.LP4MEM_00.mem_x16_00.ins_lch.<protected> 287383812500 : ###  lpddr4_debug RFSH_ALL input
# tb_top.LP4MEM_00.mem_x16_00.ins_lch.<protected> 287383812500 : ###  lpddr4_debug RFSH input
# 288940000 [TB_TOP] Read Done : Data Compare Pass
# END_OF_SIMULATION
# Out side the loop
# ////////////////////////SUMMARY////////////////////
# PASSED_RUNS :           6
# SIMULATION PASSED
# ////////////////////////////////////////////////////
# ** Note: $finish    : C:/Users/kkoo/Documents/Projects/MPMC/IpGen/Radiant/BetaSanityLocal_V4/mpmc/tes
tbench/tb_top.sv(324)
#    Time: 289770 ns  Iteration: 1  Instance: /tb_top
```

Finish Vsim

Are you sure you want to finish?

**Figure 5.2. AXI4 MPMC IP Example Design Simulation Complete**

### 5.10.1. Example Design Test Bench

Testbench file, tb_top.sv, acts as the top-level simulation instance. This file displays messages on the simulation console, monitors the test status, and instantiates the necessary instances. The firmware that runs in RISC-V handles the test flow. The firmware performs the configuration required for initialization and configures each AXI traffic generator instance to drive the required AXI4 stimulus. Each AXI traffic generator writes to separate memory ranges, no address overlapping. All AXI traffic generator instances are triggered to run in parallel.
This test performs the following 6 transactions:

1. Test 0: Single write followed by single read.
2. Test 1: Burst write with address incrementing by 2 followed by a burst read.
3. Test 2: Burst write with address incrementing by 4 followed by a burst read.
4. Test 3: Burst write with address incrementing by 8 followed by a burst read.
5. Test 4: Burst write with address incrementing by 8 followed by a delay before issuing burst read.
6. Test 5: Burst write with address incrementing by 64 followed by a burst read.

Checker is done within each of the AXI traffic generator, and the test status is printed on the Modelsim transcript.

## 5.11. Hardware Testing

The example design is synthesizable and is used for hardware testing. Refer to the Designing with the IP section.

# 6. Designing with the IP

This section provides information on generating the AXI4 MPMC IP using the Lattice Radiant software and running the simulation and synthesis. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide.

## 6.1. Generating and Instantiating the IP

You can use the Lattice Radiant software to generate IP modules and integrate them into the device's architecture. The following sections describe the steps to create a Lattice Radiant project and to generate the AXI4 MPMC IP in the Lattice Radiant software.

### 6.1.1. Creating a Lattice Radiant Project

To generate an instance of the AXI4 MPMC IP, follow these steps to create a Lattice Radiant project:

1. Launch the Lattice Radiant software and select **File > New > Project**, as shown in Figure 6.1. The **New Project** window opens. Click **Next.**

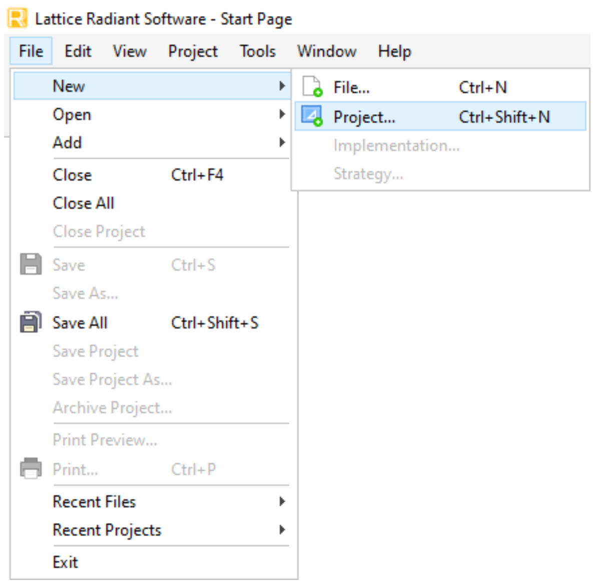
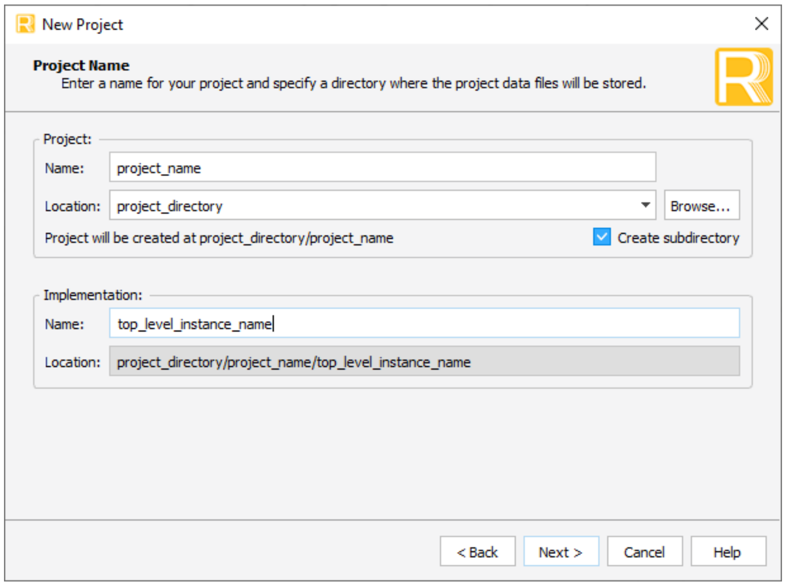


**Figure 6.1. Creating a New Lattice Radiant Project**

2. In the **New Project** window, specify a name name (<project_name>) for the Lattice Radiant project, a directory (<project_directory>) to store the project files, and a top-level design implementation name (<top_level_instance_name>), as shown in Figure 6.2. Click **Next** two times.



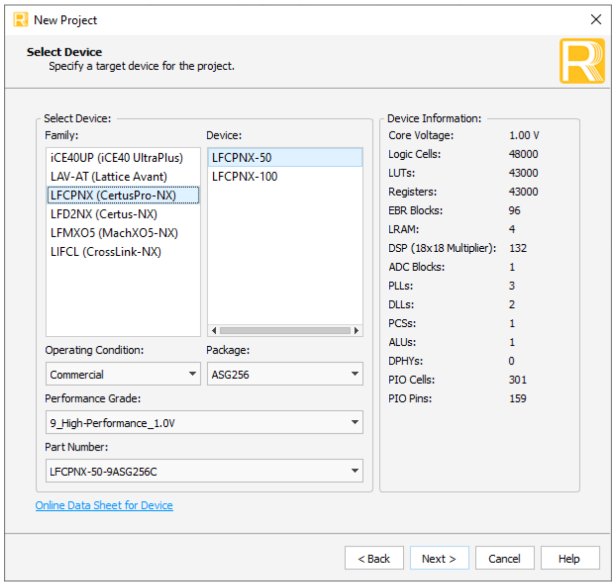
**Project Name**
Enter a name for your project and specify a directory where the project data files will be stored.

Project:
Name: project_name
Location: project_directory
Project will be created at project_directory/project_name
Create subdirectory ✓

Implementation:
Name: top_level_instance_name
Location: project_directory/project_name/top_level_instance_name

**Figure 6.2. New Project Settings**

3. Under **Family**, select CertusPro-NX. Under **Device**, **Package**, **Operating Condition**, and **Performance Grade**, make the appropriate selections representative of the selected device part number, as shown in Figure 6.3. Click **Next**.



**Select Device**
Specify a target device for the project.

Select Device:
Family:
iCE40UP (iCE40 UltraPlus)
LAV-AT (Lattice Avant)
LFCPNX (CertusPro-NX)
LFD2NX (Certus-NX)
LFMXO5 (MachXO5-NX)
LIFCL (CrossLink-NX)

Device:
LFCPNX-50
LFCPNX-100

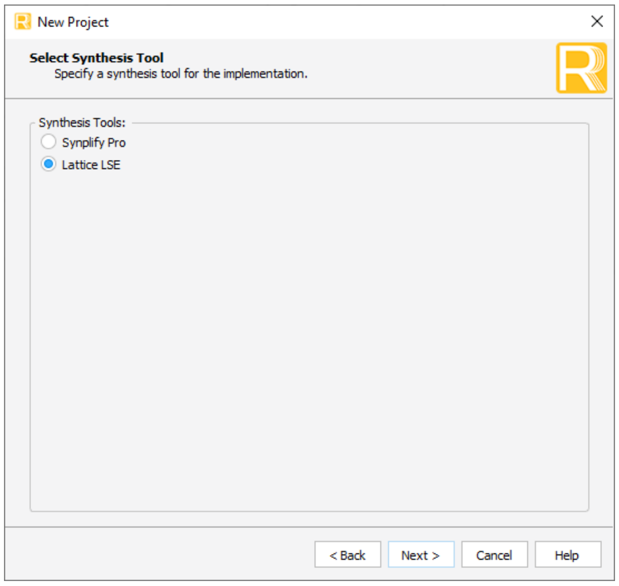
Operating Condition: Commercial
Package: ASG256
Performance Grade: 9_High-Performance_1.0V
Part Number: LFCPNX-50-9ASG256C

Device Information:

| | |
|---|---|
| Core Voltage: | 1.00 V |
| Logic Cells: | 48000 |
| LUTs: | 43000 |
| Registers: | 43000 |
| EBR Blocks: | 96 |
| LRAM: | 4 |
| DSP (18x18 Multiplier): | 132 |
| ADC Blocks: | 1 |
| PLLs: | 3 |
| DLLs: | 2 |
| PCSs: | 1 |
| ALUs: | 1 |
| DPHYs: | 0 |
| PIO Cells: | 301 |
| PIO Pins: | 159 |

**Figure 6.3. Project Device Settings**

4. Specify the desired synthesis tool for implementation of the Lattice Radiant project, as shown in Figure 6.4. Click **Next** and **Finish**.



Select Synthesis Tool
Specify a synthesis tool for the implementation.

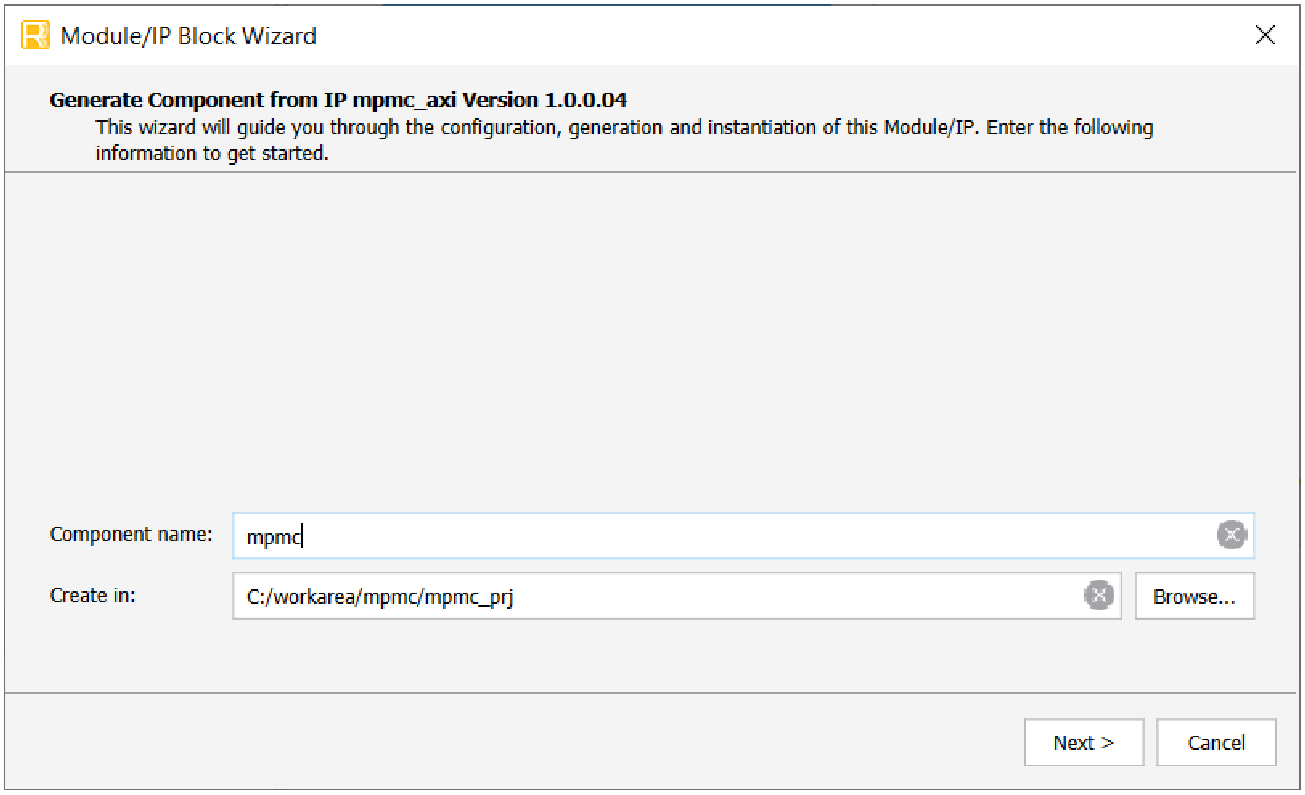
Synthesis Tools:
Synplify Pro
[✓ Lattice LSE]

Figure 6.4. Project Synthesis Tool Selection

### 6.1.2. Configuring and Generating the IP

To generate the AXI4 MPMC IP:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click **AXI4 MPMC** under **IP, Processors_Controllers_and_Peripherals** category. The **Module/IP Block Wizard** opens as shown in Figure 6.5. Enter values in the **Component name** and the **Create in** fields and click **Next**.

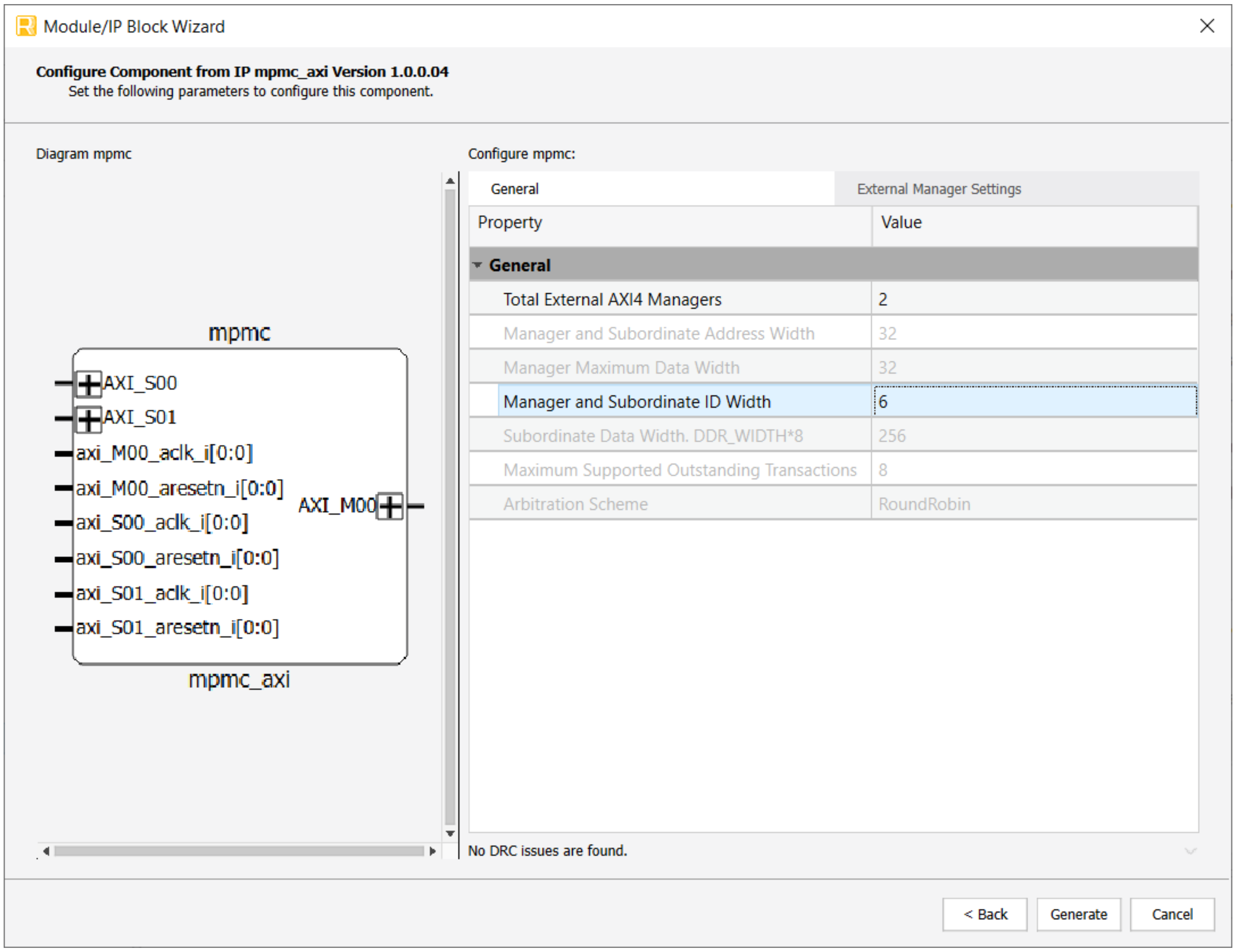


Generate Component from IP mpmc_axi Version 1.0.0.04
This wizard will guide you through the configuration, generation and instantiation of this Module/IP. Enter the following information to get started.

Component name: mpmc
Create in: C:/workarea/mpmc/mpmc_prj

Figure 6.5. Module/IP Block Wizard

3. In the next **Module/IP Block Wizard** window, customize the selected AXI4 MPMC IP using drop-down lists. Figure 6.6 shows an example configuration of the AXI4 MPMC IP. For details on the configuration options, refer to the IP Parameter Description section.



**Configure Component from IP mpmc_axi Version 1.0.0.04**
Set the following parameters to configure this component.

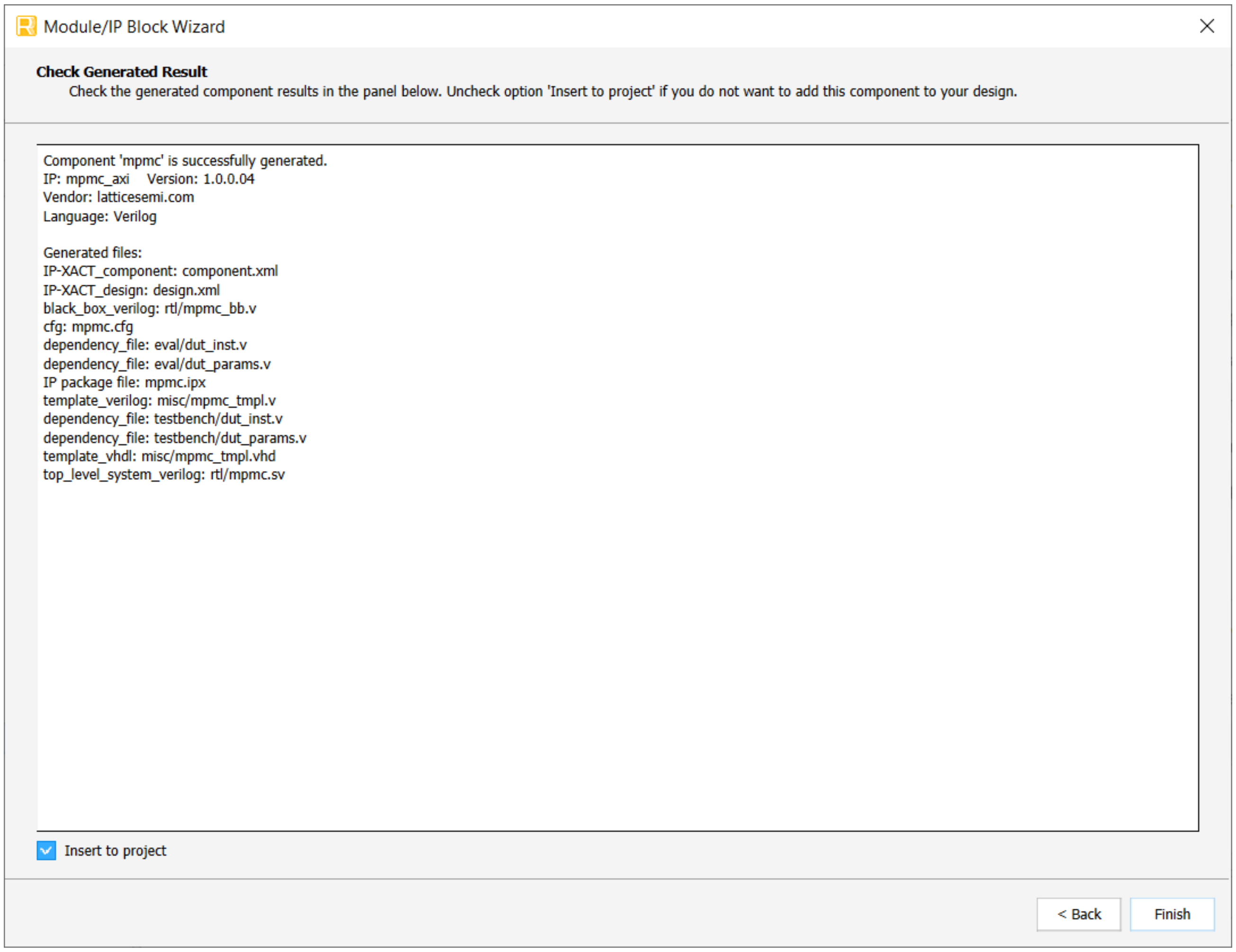
Configure mpmc:

| Property | Value |
|---|---|
| **General** | |
| Total External AXI4 Managers | 2 |
| Manager and Subordinate Address Width | 32 |
| Manager Maximum Data Width | 32 |
| Manager and Subordinate ID Width | 6 |
| Subordinate Data Width. DDR_WIDTH*8 | 256 |
| Maximum Supported Outstanding Transactions | 8 |
| Arbitration Scheme | RoundRobin |

No DRC issues are found.

**Figure 6.6. IP Configuration**

4. Click **Generate**. The **Check Generated Result** dialog box opens, showing design block messages and results as shown in Figure 6.7.



**Check Generated Result**
Check the generated component results in the panel below. Uncheck option 'Insert to project' if you do not want to add this component to your design.

```
Component 'mpmc' is successfully generated.
IP: mpmc_axi   Version: 1.0.0.04
Vendor: latticesemi.com
Language: Verilog

Generated files:
IP-XACT_component: component.xml
IP-XACT_design: design.xml
black_box_verilog: rtl/mpmc_bb.v
cfg: mpmc.cfg
dependency_file: eval/dut_inst.v
dependency_file: eval/dut_params.v
IP package file: mpmc.ipx
template_verilog: misc/mpmc_tmpl.v
dependency_file: testbench/dut_inst.v
dependency_file: testbench/dut_params.v
template_vhdl: misc/mpmc_tmpl.vhd
top_level_system_verilog: rtl/mpmc.sv
```

Insert to project [✓]

**Figure 6.7. Check Generated Result Window**

5. Click **Finish**. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields shown in Figure 6.5.

### 6.1.3. Generated Files and File Structure

The generated AXI4 MPMC module package includes the black box (<Component name>_bb.v) and instance templates (<Component name>_tmpl.v/vhd) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file (<Component name>.sv) that can be used as an instantiation template for the module is also provided. You may also use this top-level reference as the starting template for the top-level for their complete design. The generated files are listed in Table 6.1.

**Table 6.1. Generated File List**

| Attribute | Description |
|---|---|
| <Component name>.ipx | This file contains the information on the files associated to the generated IP. |
| <Component name>.cfg | This file contains the parameter values used in IP configuration. |
| component.xml | Contains the ipxact:component information of the IP. |
| design.xml | Documents the configuration parameters of the IP in IP-XACT 2014 format. |
| rtl/<Component name>.sv | This file provides an example RTL top file that instantiates the module. |

| Attribute | Description |
|---|---|
| rtl/<Component name>_bb.v | This file provides the synthesis black box. |
| misc/<Component name>_tmpl.v<br>misc /<Component name>_tmpl.vhd | These files provide instance templates for the module. |

## 6.2. Design Implementation

Completing your design includes additional steps to specify analog properties, pin assignments, and timing and physical constraints. You can add and edit the constraints using the Device Constraint Editor or by manually creating a PDC File.

Post-Synthesis constraint files (.pdc) contain both timing and non-timing constraints. The .pdc is the source file for storing logical timing/physical constraints. Constraints that are added using the Device Constraint Editor are saved to the active .pdc file. The active post-synthesis design constraint file is then used as input for post-synthesis processes.

Refer to the relevant sections in the Lattice Radiant Software User Guide for more information on creating or editing constraints and using the Device Constraint Editor.

## 6.3. Constraints

The timing constraints are based on the clock frequency used. The timing constraints for the IP are defined in constraint.pdc file, located in the eval folder of the generated IP. These constraints need to be copied to the top-level .pdc file. This file contains the constraints needed for the AXI4 MPMC IP and other constraints needed for the example design. Refer to the Running Example Design Evaluation section for more details on the constraints needed for synthesizing the example design on hardware. To understand how to constraint your design, refer to the Lattice Radiant Timing Constraints Methodology.

## 6.4. Specifying the Strategy

The Lattice Radiant software provides two predefined strategies: Area and Timing. You can also create customized strategies using the software. For details on how to create a new strategy, refer to the Strategies section of the Lattice Radiant software user guide.

## 6.5. Running Example Design Evaluation

After successfully configuring and generating the AXI4 MPMC IP, you can use the included synthesis example design for hardware evaluation of the AXI4 MPMC. Refer to the Example Design section for more details. All components specified in the Example Design Components section reside in the mc_axi4_traffic_gen.v file. This file, together with the oscillator, forms the eval_top module. The mc-axi4_traffic_gen fileset is located under the *eval/traffic_gen* directory and the content are described in Table 6.2.

**Table 6.2. Contents of the eval/traffic_gen Directory**

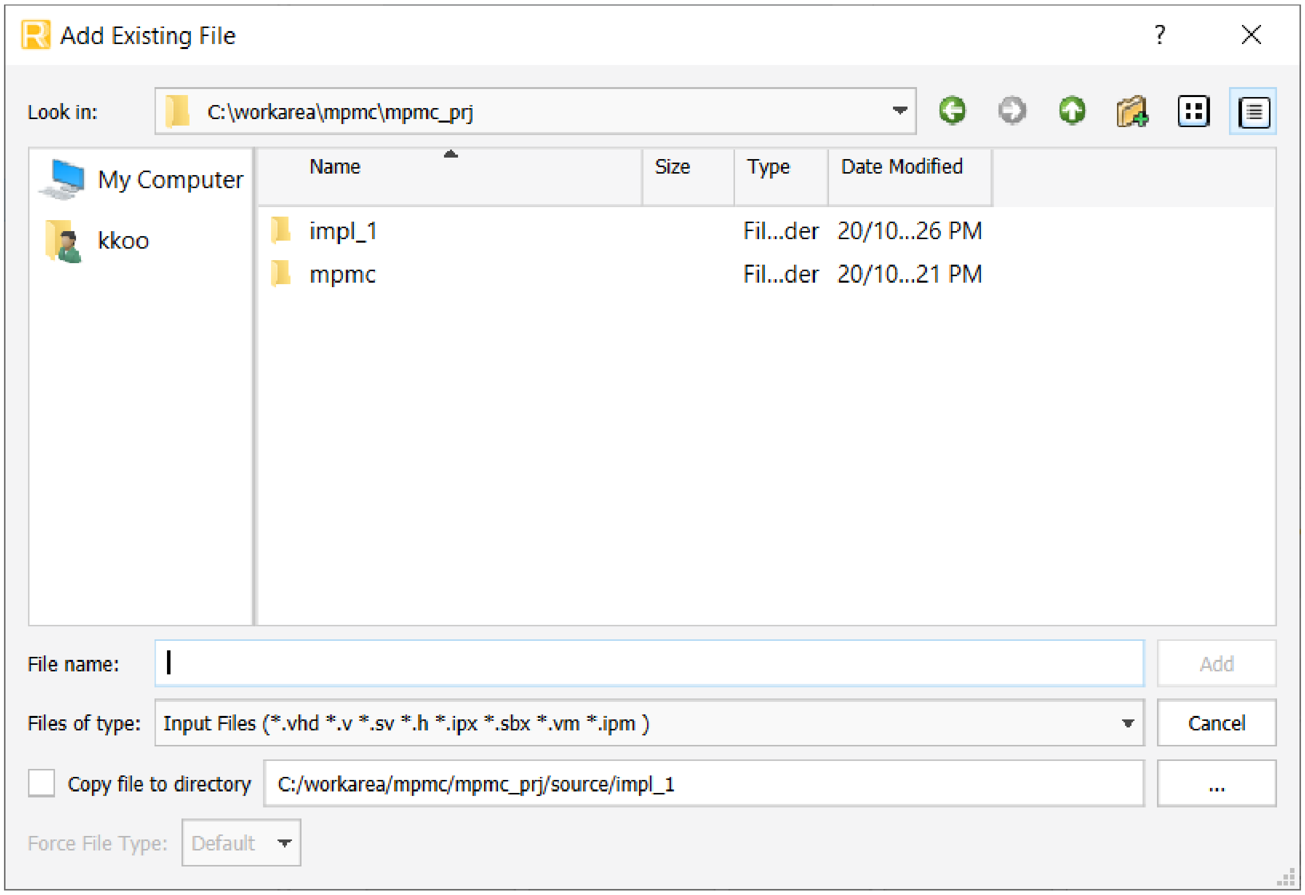
| File List | Description |
|---|---|
| ahbl0.v | AHBL_1x2. This file routes CPU data access to SYS_MEM or AHBL2APB. |
| ahbl2apb0.v | AHBL to APB bridge. |
| apb0.v | APB_1xM. This file routes CPU data access via AHBL2APB going to the CST of each module. M has a minimum of 4 and a maximum of 8, depending on the number of SI ports configured on the AXI4 MPMC IP. |
| cpu0.v | RISC-V CPU. |
| gpio0.v | GPIO module. |
| ahbl_tragen.v | RTL files for AHB-Lite traffic generator (IP v1.x.x). |
| lscc_ahbl_traffic_gen.sv, | |
| lscc_ahb_master.sv | |
| lscc_traffic_gen_csr.sv | |
| lscc_lfsr.v | |
| memc_traffic_gen.v | |

| File List | Description |
|---|---|
| ctrl_fifo.v | RTL files for AXI4 traffic generator (IP v2.x.x). |
| lscc_axi4_traffic_gen.sv | |
| lscc_axi4_m_csr.sv | |
| lscc_axi4_m_rd.sv | |
| lscc_axi4_m_wr.sv | |
| lscc_axi4_perf_calc.sv | |
| mc_axi4_traffic_gen.v | |
| lscc_osc.v | RTL files for OSC module. |
| osc0.v | |
| lscc_ram_dp_true.v | A copy of the Lattice Radiant RAM_DP_True foundation IP – required by SYSM_MEM. |
| memc_apb.v | An APB bridge to the LPDDR4 Memory Controller IP. |
| sysmem#.v | The SYS_MEM for hardware validation, enabled when eval_top.SIM=0 (Implementation).<br># can be 2, 3, or 4, is the value for the number of SI ports on the AXI4 MPMC IP.<br># = 0 is the toplevel wrapper. |
| sysmem#_sim.v | The SYS_MEM for RTL simulation, enabled when eval_top.SIM=1 (Simulation).<br># can be 2, 3, or 4, is the value for the number of SI ports on the AXI4 MPMC IP.<br># = 0 is the toplevel wrapper. |
| uart0.v | The UART module. |

### 6.5.1. Preparing the Bitstream

After configuring and generating the AXI4 MPMC IP, all associated example design files are created under the *eval* directory. To prepare the AXI4 MPMC example design project and generate the associated bitstream, follow these steps:

1. After generating the AXI4 MPMC IP, the Lattice Radiant project contains the <instance_name>.ipx under the input files of the project. Alternatively, to locate the file, right-click on **Input Files** and select **Add > Existing File** under the **File List** tab at the lower-left corner of the Lattice Radiant project window. The **Add Existing File** dialog box opens. Navigate to the *<instance_directory>/<instance_name>* directory and select the <instance_name>.ipx file. Click **Add**.
2. To add the top-level example design file to the project, right-click on **Input Files** and select **Add > Existing File**. The **Add Existing File** dialog box opens, as shown in Figure 6.8. Navigate to the eval directory and select the eval_top.sv file. Click **Add**.


Add Existing File

Look in: C:\workarea\mpmc\mpmc_prj

My Computer
kkoo

| Name | Size | Type | Date Modified |
|---|---|---|---|
| impl_1 | | Fil...der | 20/10...26 PM |
| mpmc | | Fil...der | 20/10...21 PM |

File name:
Files of type: Input Files (*.vhd *.v *.sv *.h *.ipx *.sbx *.vm *.ipm )
Copy file to directory: C:/workarea/mpmc/mpmc_prj/source/impl_1
Force File Type: Default

**Figure 6.8. Adding Existing File**

3. This example design uses a statically generated LPDDR4 Memory Controller IP, and requires the .ldc file. To add this file, right-click <instance_name>.ipx and select **Attach Constraint File**. In the **Attach Constraint File** dialog box, navigate to the eval directory and select the mc.ldc file. Click **Add**.
4. To add the pre-synthesis constraint file to the project, right-click **Pre-Synthesis Constraint Files** and select **Add > Existing File**. In the **Add Existing File** dialog box, check the **Copy file to directory** option to ensure any user modifications is not overwritten when the AXI4 MPMC IP Core is regenerated. Navigate to the eval directory and select the clock_constraint.sdc file. Click **Add**.
5. To add the post-synthesis constraint file to the project, right-click **Post-Synthesis Constraint Files** and select **Add > Existing File**. In the **Add Existing File** dialog box, check the **Copy file to directory** option. Navigate to the eval directory and select the constraint.pdc file. The constraint.pdc is provided with the pin assignment for the CertusPro-NX Versa Board. Click **Add**.
6. Before running the example design on hardware, generate a bitstream by clicking the ▶ button. A <project_name>_<top_level_instance_name>.bit file is generated under the *<project_directory>/<project_name>/<top_level_instance_name>* directory.

## 6.5.2. Running on Hardware

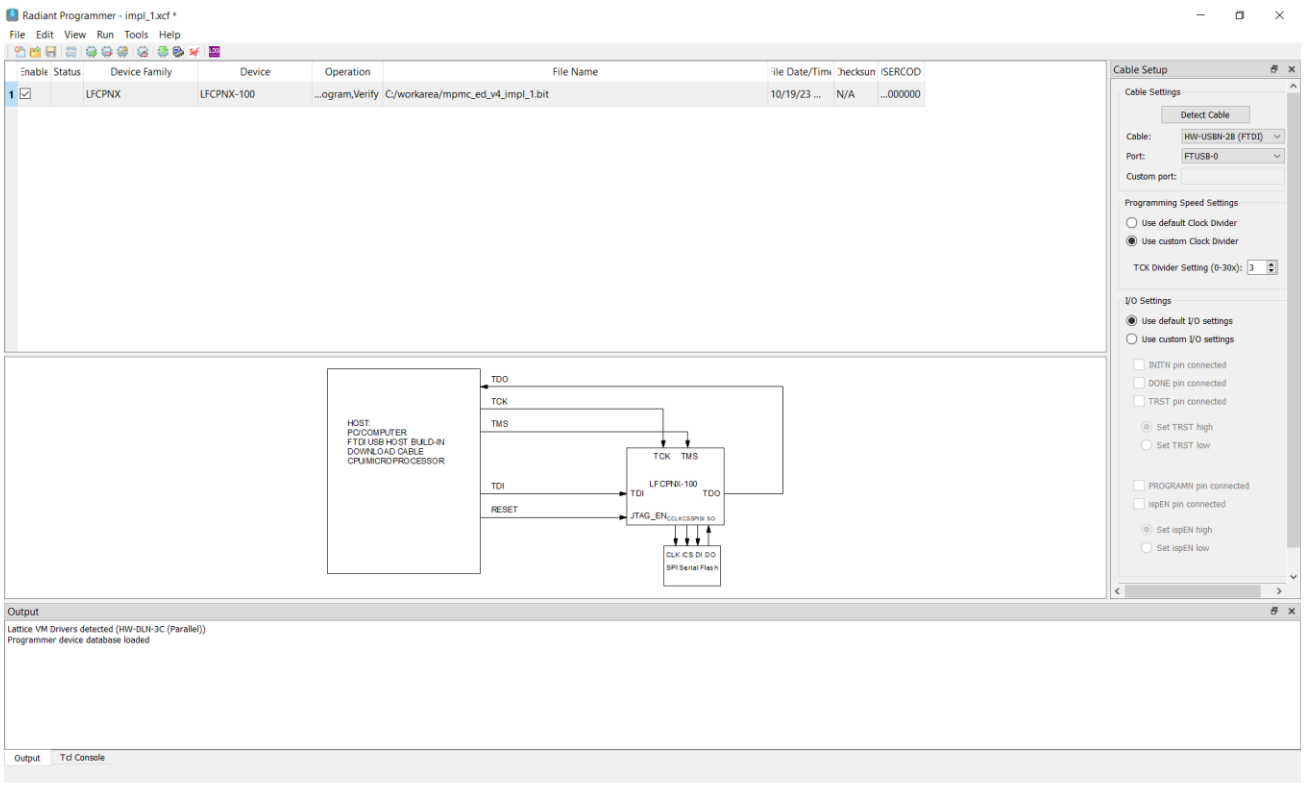
The requirements to perform hardware evaluation of the AXI4 MPMC example design are as follows:

- CertusPro-NX FPGA board with UART connection
- Associated power supply and programming cable
- Personal computer running the Lattice Radiant software version 2023.1 or later
- Lattice Propel software version 2023.1 or later, or any terminal that supports serial communication

To run the example design on hardware, a bitstream file is required. To generate the .bit file, refer to the Preparing the Bitstream section.

To program the FPGA board with the example design, follow these steps:

1. Connect the FPGA board to the computer and power on the board.
2. Run the Lattice Radiant Programmer by clicking the button. The Lattice Radiant Programmer launches as shown in Figure 6.9, scans for devices, and configures the programmer automatically.



| Enable | Status | Device Family | Device | Operation | File Name | File Date/Time | Checksum | USERCODE |
|---|---|---|---|---|---|---|---|---|
| 1 ☑ | | LFCPNX | LFCPNX-100 | ...ogram,Verify | C:/workarea/mpmc_ed_v4_impl_1.bit | 10/19/23 ... | N/A | ...000000 |

Cable Setup

Cable Settings

Detect Cable

Cable: HW-USBN-2B (FTDI)

Port: FTUSB-0

Custom port:

Programming Speed Settings

○ Use default Clock Divider

● Use custom Clock Divider

TCK Divider Setting (0-30x): 3

I/O Settings

● Use default I/O settings

○ Use custom I/O settings

☐ INITN pin connected

☐ DONE pin connected

☐ TRST pin connected

● Set TRST high

○ Set TRST low

☐ PROGRAMN pin connected

☐ ispEN pin connected

● Set ispEN high

○ Set ispEN low

Output

```
Lattice VM Drivers detected (HW-DLN-3C (Parallel))
Programmer device database loaded
```

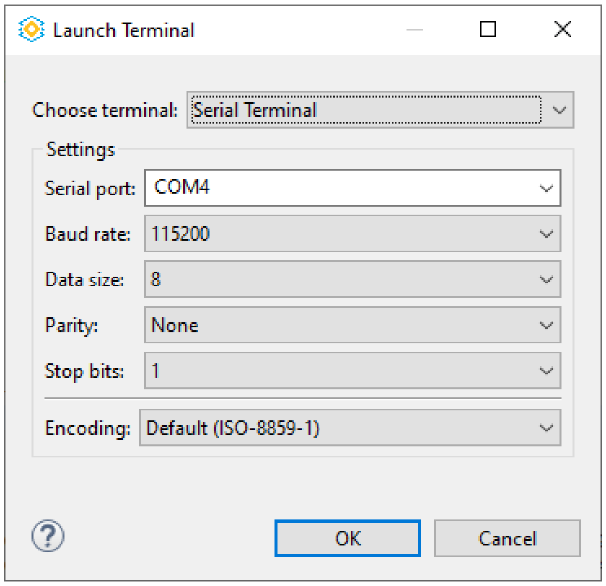
**Figure 6.9. Lattice Radiant Programmer**

3. Click under the File Name field and then click **…** to the right of the field to launch the **Open File** dialog box. Navigate to the bitstream generated from Step 4 in the Preparing the Bitstream section. Click **Open**.
4. Click the button to program the Lattice FPGA device.

After programming the Lattice FPGA with the example design bitstream, follow these steps to launch a serial terminal:

**Note:** If you use the Lattice Propel terminal, continue with Step 5. If you use a different serial communication terminal, skip to Step 7.

5. Open the Lattice Propel software and select Launch. The default workspace opens.
6. To open a terminal, click the button.
7. Configure the terminal settings to be a Serial Terminal with the appropriate **Baud rate**, **Data size**, **Parity**, **Stop bits**, and **Encoding**, as shown in Figure 6.10. Click **OK** to launch the **Terminal** pane at the bottom of the Lattice Propel software window. Note that the **Serial port** varies depending on the computer setup.



Choose terminal: Serial Terminal
Settings
Serial port: COM4
Baud rate: 115200
Data size: 8
Parity: None
Stop bits: 1
Encoding: Default (ISO-8859-1)

**Figure 6.10. Serial Terminal Settings**

To run the example design, assert the the rstn_i signal. This signal prompts the following message to appear in the terminal. If no message appears in the terminal, close the current serial terminal and open a new terminal using a different **Serial port**.

```
Press 1 to enter initial Vref values or press any key to proceed Training
```

After you provide input over the serial terminal, training of the LPDDR4 SDRAM begins, followed by data access checks and a printout of the performance of the memory interface.

```
Training Passed.
Starting Data Access Check.
0 1 2 3 4 5 6
Performance Values :
Bus_efficiency : 80 Perf_mbps : 27261

Data Access Check Pass.
```

If you encounter failure during training, a message is sent over the serial connection notifying the failure at a particular stage. The loop-back data access check is aborted.

```
Write Training Failed!
Aborting Data Access Check...
```

## 6.6. Running Functional Simulation

After successfully configuring and generating the AXI4 MPMC IP, you can perform functional simulation of the AXI4 MPMC IP using the example design. All associated simulation files are located under the testbench directory. To prepare the AXI4 MPMC example design project for simulation, follow these steps:

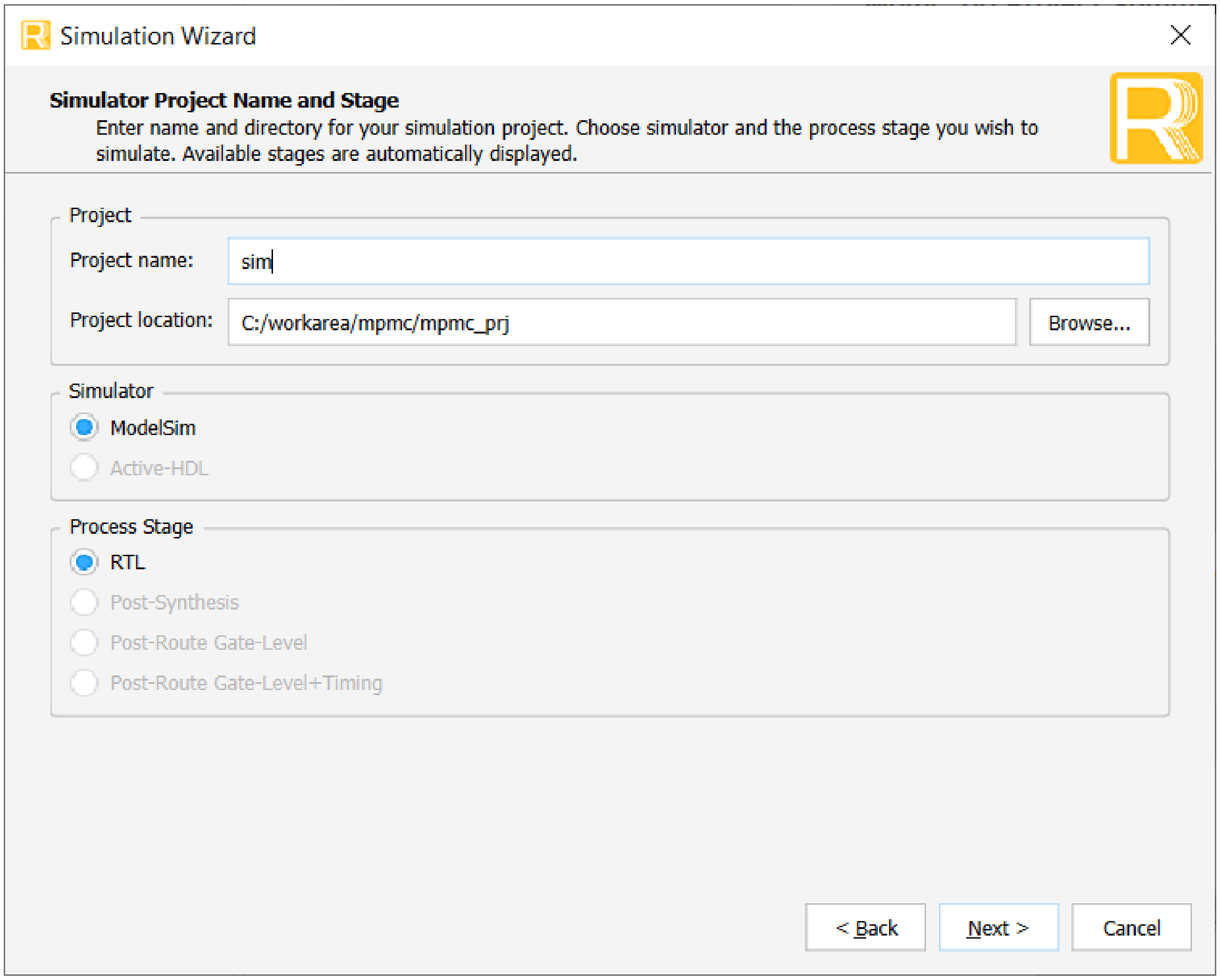
1. Before simulating the example design, complete steps 1-2 in the Preparing the Bistream section.
2. To add the top-level testbench file to the project, select **File > Add > Existing Simulation File**. The **Add Existing Simulation File** dialog box opens. Navigate to the testbench directory and select the tb_top.sv file. Click **Add**.

The tb_top.sv file instantiates the top-level example design file, eval_top.sv, with the SIM parameter set to 1. This parameter shortens the initialization and training sequence of the LPDDR4 interface and disables the UART interface for simulation. Do not modify the SIM parameter for hardware implementation.

When SIM parameter is set to 1, this parameter programs 0x1E to the Training Operation Register (TRN_OP_REG) of the LPDDR4 memory controller to speed up the simulation runtime. For more details, refer to the LPDDR4 Memory Controller for Nexus Devices User Guide (FPGA-IPUG-02127).

To create the simulation environment to simulate the AXI4 MPMC example design, follow these steps:

1. Launch the simulation wizard in the Lattice Radiant software by selecting **Tools > Simulation Wizard**. The **Simulation Wizard** dialog box opens. Click **Next** and provide a **Project name** (<sim_name>) and **Project location** (<sim_directory>) for the simulation project, as shown in Figure 6.11. The default directory is set to *<project_directory>/<project_name>*. Click **Next.**


Simulation Wizard

**Simulator Project Name and Stage**

Enter name and directory for your simulation project. Choose simulator and the process stage you wish to simulate. Available stages are automatically displayed.

Project

Project name: sim

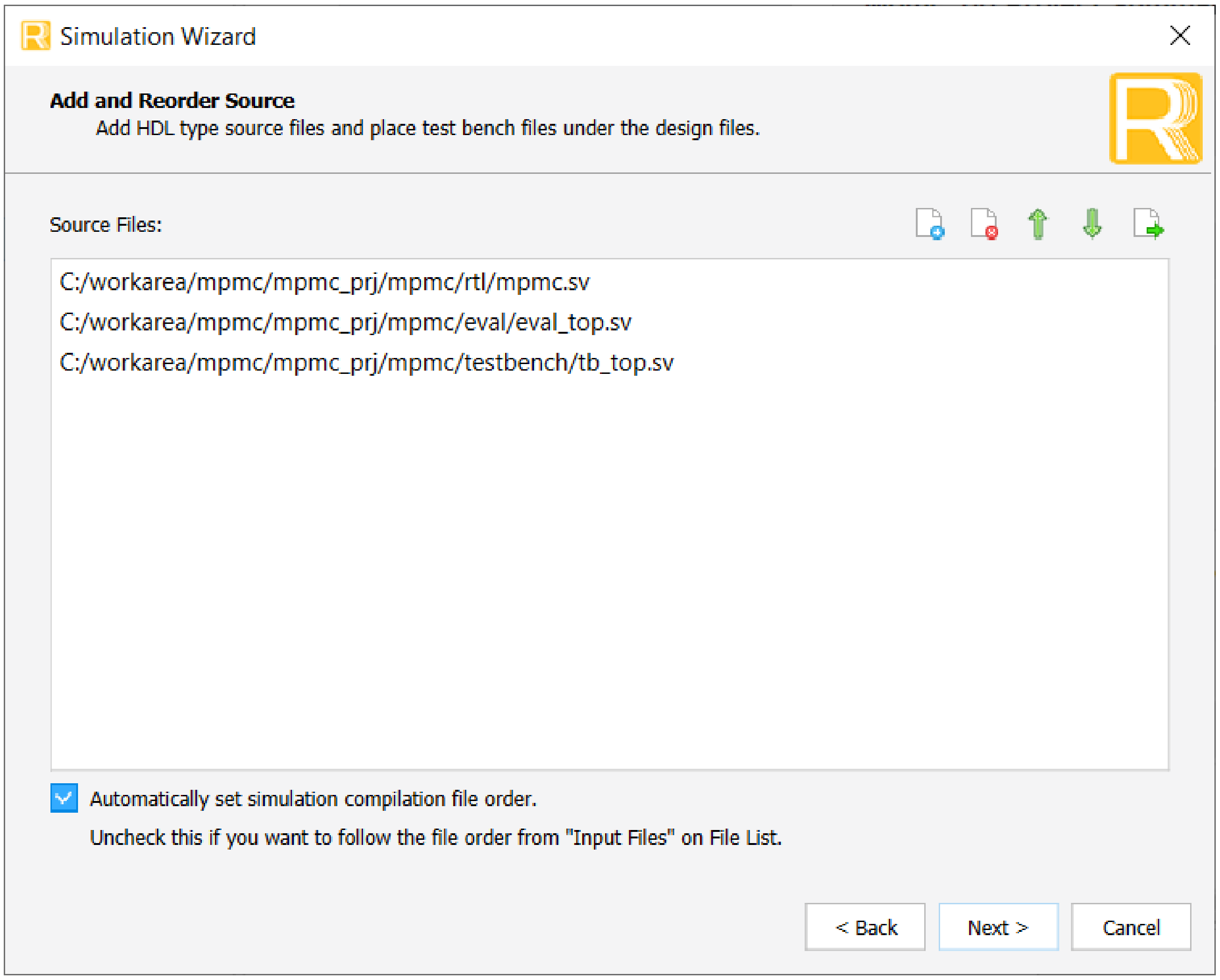
Project location: C:/workarea/mpmc/mpmc_prj

Simulator: [✓ ModelSim] Active-HDL

Process Stage: [✓ RTL] Post-Synthesis, Post-Route Gate-Level, Post-Route Gate-Level+Timing

**Figure 6.11. Simulation Wizard**

2. The **Add and Reorder Source** window lists the source files in this example design, as shown in Figure 6.12. Besides the IP file, check that the **Source Files** list contains only the top-level evaluation (eval_top.sv) and top-level testbench (tb_top.sv) files. Click **Next.**


Simulation Wizard

**Add and Reorder Source**

Add HDL type source files and place test bench files under the design files.

Source Files:

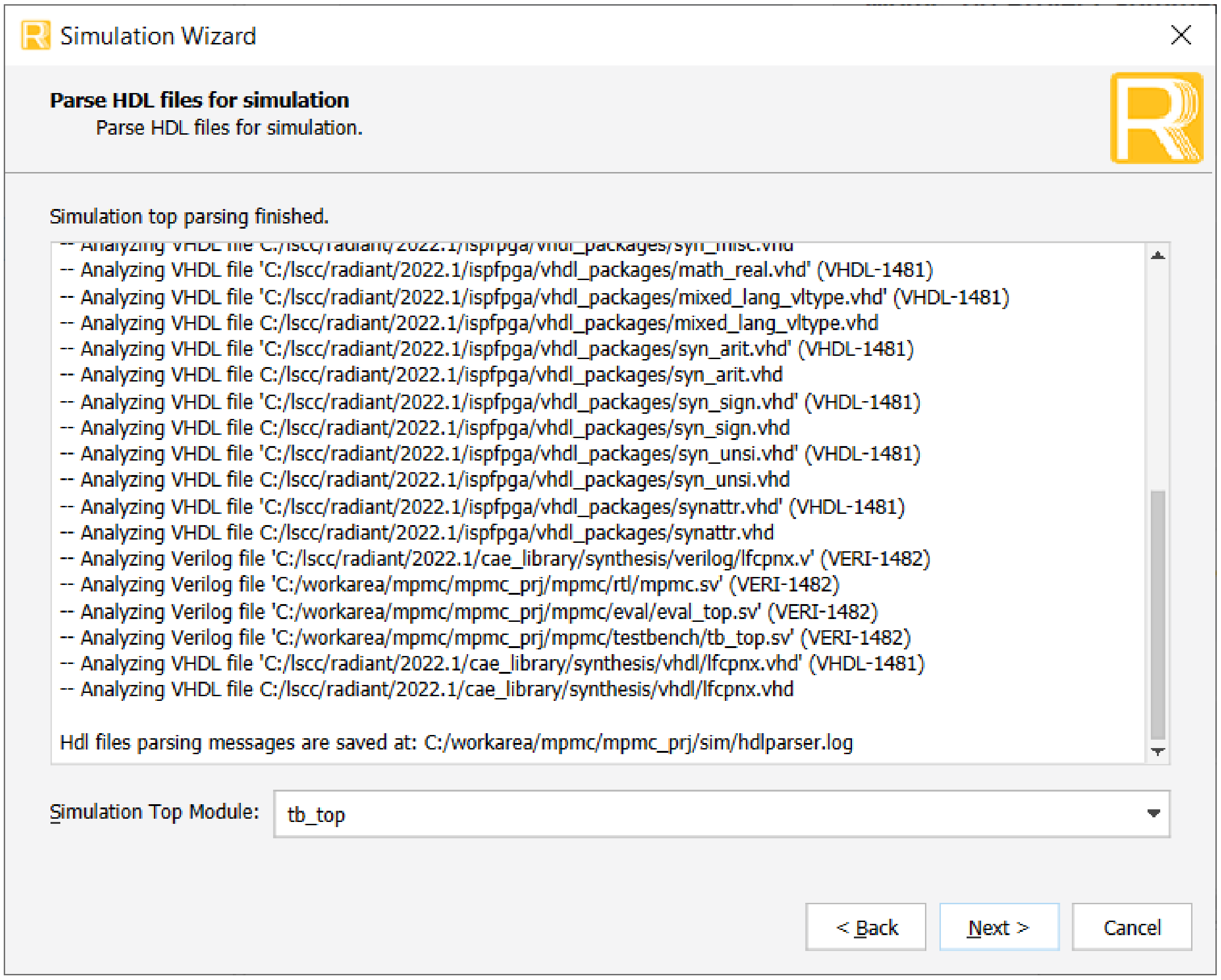
```
C:/workarea/mpmc/mpmc_prj/mpmc/rtl/mpmc.sv
C:/workarea/mpmc/mpmc_prj/mpmc/eval/eval_top.sv
C:/workarea/mpmc/mpmc_prj/mpmc/testbench/tb_top.sv
```

[✓] Automatically set simulation compilation file order.

Uncheck this if you want to follow the file order from "Input Files" on File List.

**Figure 6.12. Adding and Reordering Simulation Source Files**

3. In the **Parse HDL files for simulation** window, check that the **Simulation Top Module** is set to tb_top, as shown in Figure 6.13. Click **Next.**


Simulation Wizard
Parse HDL files for simulation
Parse HDL files for simulation.

Simulation top parsing finished.

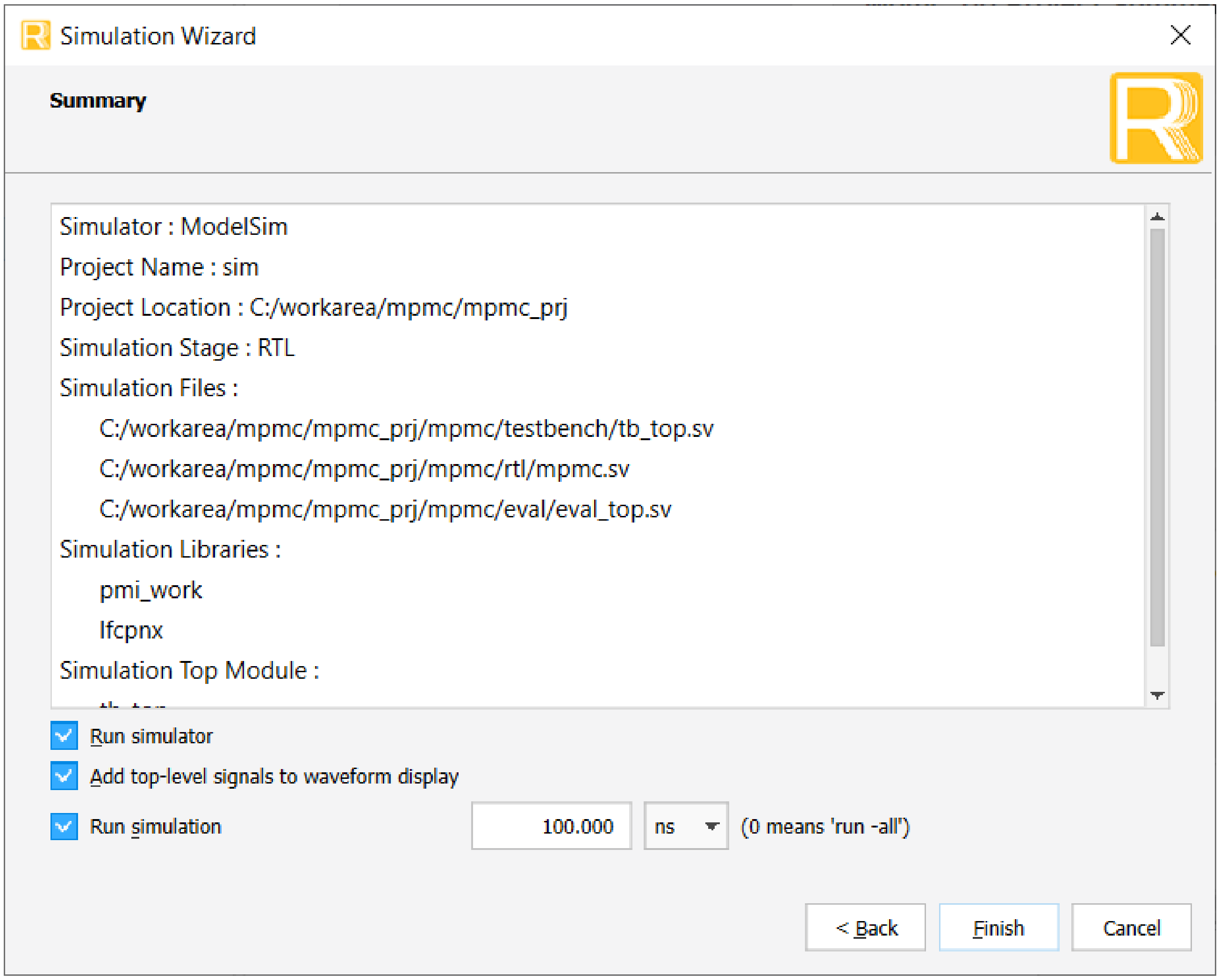
```
-- Analyzing VHDL file C:/lscc/radiant/2022.1/ispfpga/vhdl_packages/syn_misc.vhd
-- Analyzing VHDL file 'C:/lscc/radiant/2022.1/ispfpga/vhdl_packages/math_real.vhd' (VHDL-1481)
-- Analyzing VHDL file 'C:/lscc/radiant/2022.1/ispfpga/vhdl_packages/mixed_lang_vltype.vhd' (VHDL-1481)
-- Analyzing VHDL file C:/lscc/radiant/2022.1/ispfpga/vhdl_packages/mixed_lang_vltype.vhd
-- Analyzing VHDL file 'C:/lscc/radiant/2022.1/ispfpga/vhdl_packages/syn_arit.vhd' (VHDL-1481)
-- Analyzing VHDL file C:/lscc/radiant/2022.1/ispfpga/vhdl_packages/syn_arit.vhd
-- Analyzing VHDL file 'C:/lscc/radiant/2022.1/ispfpga/vhdl_packages/syn_sign.vhd' (VHDL-1481)
-- Analyzing VHDL file C:/lscc/radiant/2022.1/ispfpga/vhdl_packages/syn_sign.vhd
-- Analyzing VHDL file 'C:/lscc/radiant/2022.1/ispfpga/vhdl_packages/syn_unsi.vhd' (VHDL-1481)
-- Analyzing VHDL file C:/lscc/radiant/2022.1/ispfpga/vhdl_packages/syn_unsi.vhd
-- Analyzing VHDL file 'C:/lscc/radiant/2022.1/ispfpga/vhdl_packages/synattr.vhd' (VHDL-1481)
-- Analyzing VHDL file C:/lscc/radiant/2022.1/ispfpga/vhdl_packages/synattr.vhd
-- Analyzing Verilog file 'C:/lscc/radiant/2022.1/cae_library/synthesis/verilog/lfcpnx.v' (VERI-1482)
-- Analyzing Verilog file 'C:/workarea/mpmc/mpmc_prj/mpmc/rtl/mpmc.sv' (VERI-1482)
-- Analyzing Verilog file 'C:/workarea/mpmc/mpmc_prj/mpmc/eval/eval_top.sv' (VERI-1482)
-- Analyzing Verilog file 'C:/workarea/mpmc/mpmc_prj/mpmc/testbench/tb_top.sv' (VERI-1482)
-- Analyzing VHDL file 'C:/lscc/radiant/2022.1/cae_library/synthesis/vhdl/lfcpnx.vhd' (VHDL-1481)
-- Analyzing VHDL file C:/lscc/radiant/2022.1/cae_library/synthesis/vhdl/lfcpnx.vhd

Hdl files parsing messages are saved at: C:/workarea/mpmc/mpmc_prj/sim/hdlparser.log
```

Simulation Top Module: tb_top

**Figure 6.13. Parsing Simulation HDL Files**

4. The **Summary** window opens as shown in Figure 6.14. By default, the simulation runs for 100 µs. Before running the simulation, configure the waveform to log signals of interest in the Modelsim simulator. To run the simulation until completion, enter this tcl command in the Modelsim simulator: `run -all`. Alternatively, to run the simulation with the default top-level signals, change the value from 100 µs to 0 µs to execute the simulation completely.



Simulator : ModelSim
Project Name : sim
Project Location : C:/workarea/mpmc/mpmc_prj
Simulation Stage : RTL
Simulation Files :
C:/workarea/mpmc/mpmc_prj/mpmc/testbench/tb_top.sv
C:/workarea/mpmc/mpmc_prj/mpmc/rtl/mpmc.sv
C:/workarea/mpmc/mpmc_prj/mpmc/eval/eval_top.sv
Simulation Libraries :
pmi_work
lfcpnx
Simulation Top Module :

✓ Run simulator
✓ Add top-level signals to waveform display
✓ Run simulation 100.000 ns (0 means 'run -all')

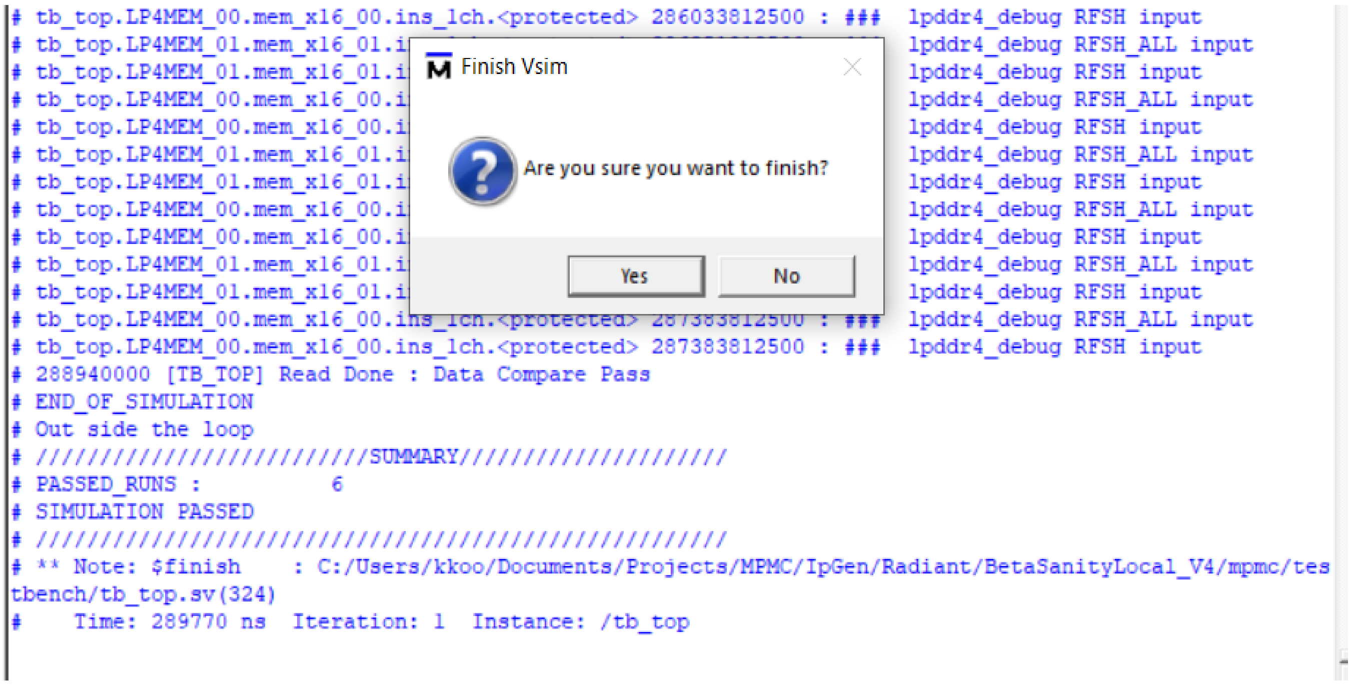
**Figure 6.14. Simulation Summary**

Shortening the reset and CKE initialization results in violation of timing requirements for the LPDDR4 simulation model. In the Modelsim simulation log, disregard the following error messages from the violation.

```
# tb_top.LP4MEM_00.mem_x16_00.ins_1ch.<protected> 26083125000 : ###  lpddr4_debug
RESET_n high input
# tb_top.LP4MEM_01.mem_x16_01.ins_1ch.<protected> 26083125000 : ###  lpddr4_debug
RESET_n high input
# Error: tb_top.LP4MEM_01.mem_x16_01.ins_1ch.<protected>.<protected> 26083125000  tINIT1
Error.
# Error: tb_top.LP4MEM_00.mem_x16_00.ins_1ch.<protected>.<protected> 26083125000  tINIT1
Error.
# tb_top.LP4MEM_00.mem_x16_00.ins_1ch.<protected> 32804075000 : ###  lpddr4_debug CKE
high input
# tb_top.LP4MEM_01.mem_x16_01.ins_1ch.<protected> 32804075000 : ###  lpddr4_debug CKE
high input
# Error: tb_top.LP4MEM_01.mem_x16_01.ins_1ch.<protected>.<protected> 32804075000  tINIT3
Error.
# Error: tb_top.LP4MEM_00.mem_x16_00.ins_1ch.<protected>.<protected> 32804075000  tINIT3
Error.
```

### 6.6.1. Simulation Results

When simulation is successfully complete, a "Simulation Passed" message is displayed, as shown in Figure 6.15.



```
# tb_top.LP4MEM_00.mem_x16_00.ins_lch.<protected> 286033812500 : ###  lpddr4_debug RFSH input
# tb_top.LP4MEM_01.mem_x16_01.i                                       lpddr4_debug RFSH_ALL input
# tb_top.LP4MEM_01.mem_x16_01.i                                       lpddr4_debug RFSH input
# tb_top.LP4MEM_00.mem_x16_00.i                                       lpddr4_debug RFSH_ALL input
# tb_top.LP4MEM_00.mem_x16_00.i                                       lpddr4_debug RFSH input
# tb_top.LP4MEM_01.mem_x16_01.i                                       lpddr4_debug RFSH_ALL input
# tb_top.LP4MEM_01.mem_x16_01.i                                       lpddr4_debug RFSH input
# tb_top.LP4MEM_00.mem_x16_00.i                                       lpddr4_debug RFSH_ALL input
# tb_top.LP4MEM_00.mem_x16_00.i                                       lpddr4_debug RFSH input
# tb_top.LP4MEM_01.mem_x16_01.i                                       lpddr4_debug RFSH_ALL input
# tb_top.LP4MEM_01.mem_x16_01.i                                       lpddr4_debug RFSH input
# tb_top.LP4MEM_00.mem_x16_00.ins_lch.<protected> 287383812500 : ###  lpddr4_debug RFSH_ALL input
# tb_top.LP4MEM_00.mem_x16_00.ins_lch.<protected> 287383812500 : ###  lpddr4_debug RFSH input
# 288940000 [TB_TOP] Read Done : Data Compare Pass
# END_OF_SIMULATION
# Out side the loop
# ///////////////////////////SUMMARY////////////////////
# PASSED_RUNS :           6
# SIMULATION PASSED
# ///////////////////////////////////////////////////////
# ** Note: $finish    : C:/Users/kkoo/Documents/Projects/MPMC/IpGen/Radiant/BetaSanityLocal_V4/mpmc/tes
tbench/tb_top.sv(324)
#    Time: 289770 ns  Iteration: 1  Instance: /tb_top
```

**Figure 6.15. Simulation Completes Successfully**

### 6.6.2. Simulation Waveforms

Figure 6.16 and Figure 6.17 show the simulation waveforms for write and read transactions.

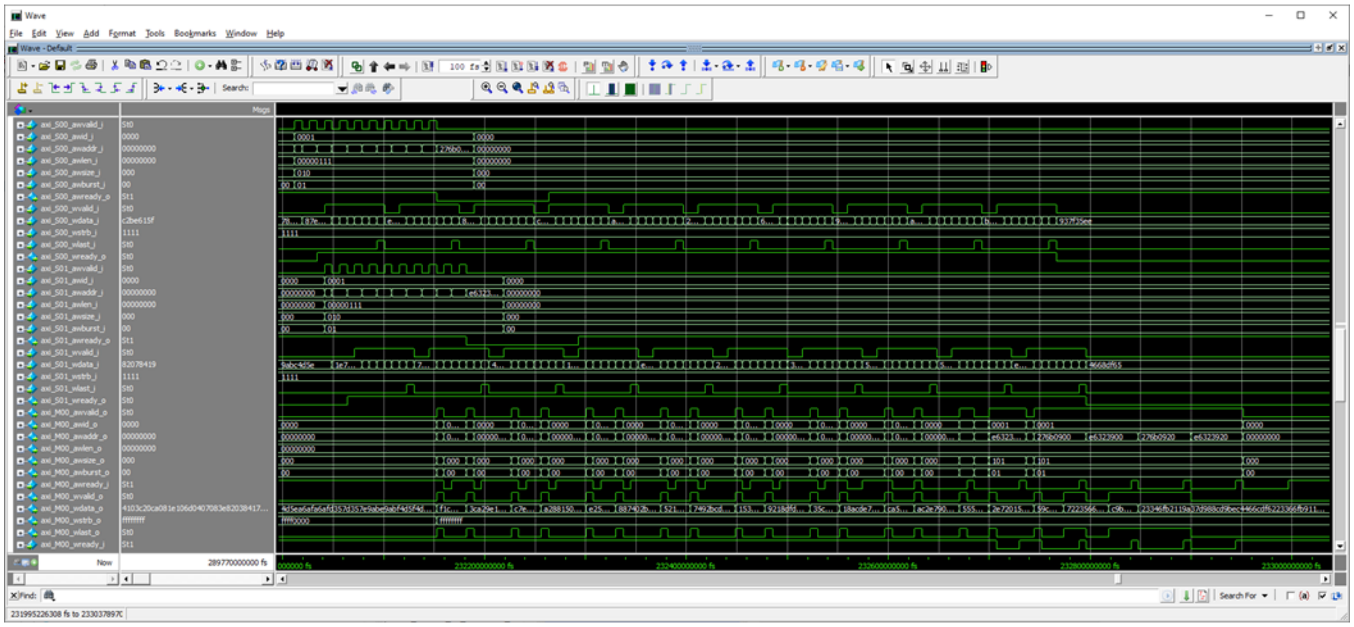
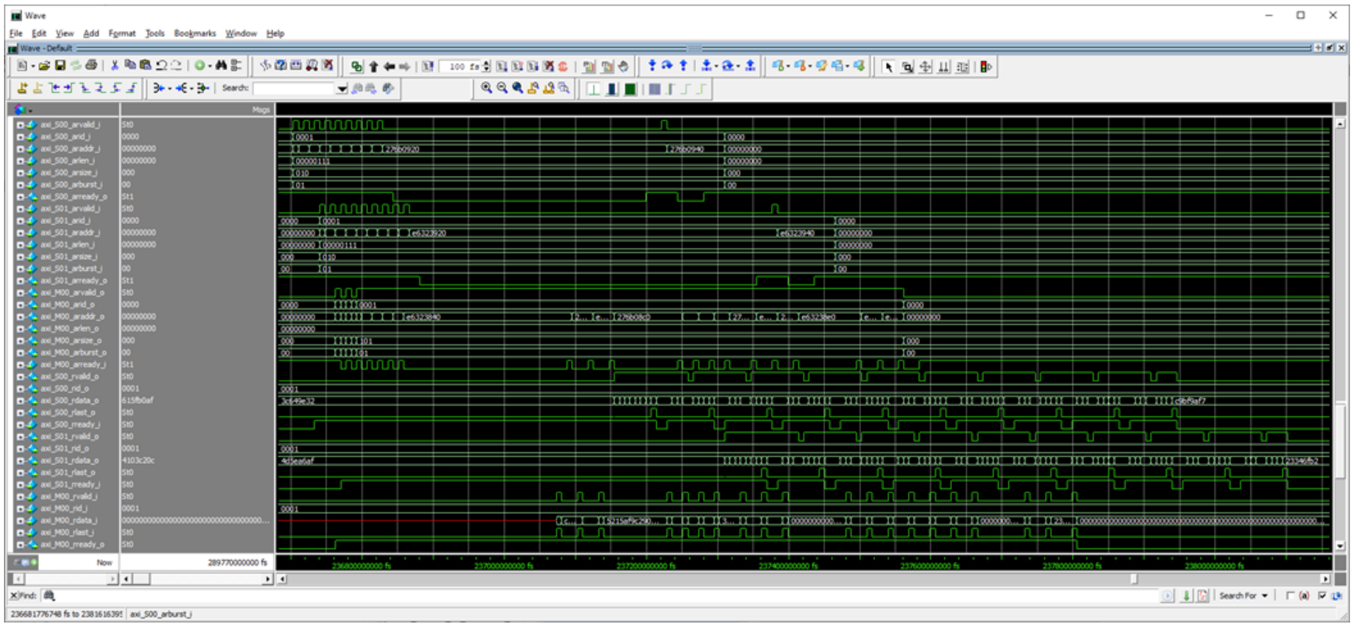


**Figure 6.16. Write Transactions**



| | Msgs |
|---|---|
| axi_S00_arvalid_i | St0 |
| axi_S00_arid_i | 0000 |
| axi_S00_araddr_i | 00000000 |
| axi_S00_arlen_i | 00000000 |
| axi_S00_arsize_i | 000 |
| axi_S00_arburst_i | 00 |
| axi_S00_arready_o | St1 |
| axi_S01_arvalid_i | St0 |
| axi_S01_arid_i | 0000 |
| axi_S01_araddr_i | 00000000 |
| axi_S01_arlen_i | 00000000 |
| axi_S01_arsize_i | 000 |
| axi_S01_arburst_i | 00 |
| axi_S01_arready_o | St1 |
| axi_M00_arvalid_o | St0 |
| axi_M00_arid_o | 0000 |
| axi_M00_araddr_o | 00000000 |
| axi_M00_arlen_o | 00000000 |
| axi_M00_arsize_o | 000 |
| axi_M00_arburst_o | 00 |
| axi_M00_arready_i | St1 |
| axi_S00_rvalid_o | St0 |
| axi_S00_rid_o | 0001 |
| axi_S00_rdata_o | 615fb0af |
| axi_S00_rlast_o | St0 |
| axi_S00_rready_i | St0 |
| axi_S01_rvalid_o | St0 |
| axi_S01_rid_o | 0001 |
| axi_S01_rdata_o | 4103c20c |
| axi_S01_rlast_o | St0 |
| axi_S01_rready_i | St0 |
| axi_M00_rvalid_i | St0 |
| axi_M00_rid_i | 0001 |
| axi_M00_rdata_i | 00000000000000000000000000000000... |
| axi_M00_rlast_i | St0 |
| axi_M00_rready_o | St0 |
| Now | 289770000000 fs |

236681776748 fs to 2381616395 | axi_S00_arburst_i

**Figure 6.17. Read Transactions**

# Appendix A. Resource Utilization

The values in this section are obtained from the Lattice Radiant software version 2023.2.0.15.0.

Table A.1 shows a sample resource utilization of the AXI4 MPMC IP Core on LFCPNX-100-9LFG672I device.

**Table A.1. Sample Resource Utilization of the AXI4 MPMC IP Core on LFCPNX-100-9LFG672I Device**

| Number of External Managers | CDC Enabled | Virtual I/O | Slices | LUTs | Registers | EBR |
|---|---|---|---|---|---|---|
| 2 | Yes | 1184 | 7753 | 5944 | 4055 | 34 |
| 3 | | 1420 | 11169 | 8662 | 5972 | 51 |
| 4 | | 1656 | 14146 | 11084 | 7882 | 68 |
| 2 | No | 1184 | 7867 | 5771 | 4271 | 34 |
| 3 | | 1420 | 11357 | 8410 | 6296 | 51 |
| 4 | | 1656 | 14418 | 10782 | 8314 | 68 |

Table A.2 shows a sample FMAX of the AXI4 MPMC IP Core on LFCPNX-100-9LFG672I device.

**Table A.2. Sample FMAX of the AXI4 MPMC IP Core on LFCPNX-100-9LFG672I Device**

| Number of External Managers | CDC Enabled | SI FMAX (MHz) | MI FMAX (MHz) | Minimum Slack (ns) |
|---|---|---|---|---|
| 2 | Yes | 142.86 | 181.82 | 0.139 |
| 3 | | 136.99 | 181.82 | 0.136 |
| 4 | | 135.14 | 181.82 | 0.026 |
| 2 | No | 136.99 | 136.99 | 0.263 |
| 3 | | 125.00 | 125.00 | 0.215 |
| 4 | | 125.00 | 125.00 | 0.537 |

Table A.3 shows a sample resource utilization of the AXI4 MPMC IP Core on LAV-AT-E70-3LFG1156I device.

**Table A.3. Sample Resource Utilization of the AXI4 MPMC IP Core on LAV-AT-E70-3LFG1156I Device**

| Number of External Managers | CDC Enabled | Virtual I/O | Slices | LUTs | Registers | EBR |
|---|---|---|---|---|---|---|
| 2 | Yes | 1184 | 4559 | 5781 | 4129 | 18 |
| 3 | | 1420 | 6517 | 8389 | 6091 | 27 |
| 4 | | 1656 | 8615 | 10734 | 8001 | 36 |
| 2 | No | 1184 | 4541 | 5547 | 4337 | 18 |
| 3 | | 1420 | 6531 | 8179 | 6413 | 27 |
| 4 | | 1656 | 8418 | 10484 | 8457 | 36 |

Table A.4 shows a sample FMAX of the AXI4 MPMC IP Core on LAV-AT-E70-3LFG1156I device.

**Table A.4. Sample FMAX of the AXI4 MPMC IP Core on LAV-AT-E70-3LFG1156I Device**

| Number of External Managers | CDC Enabled | SI FMAX (MHz) | MI FMAX (MHz) | Minimum Slack (ns) |
|---|---|---|---|---|
| 2 | Yes | 259.74 | 303.03 | 0.219 |
| 3 | | 250.00 | 303.03 | 0.094 |
| 4 | | 250.00 | 285.71 | 0.128 |
| 2 | No | 227.27 | 227.27 | 0.059 |
| 3 | | 219.78 | 219.78 | 0.023 |
| 4 | | 217.39 | 217.39 | 0.021 |

# References

- LPDDR4 Memory Controller for Nexus Devices User Guide (FPGA-IPUG-02127)
- CertusPro-NX web page
- Avant-E web page
- Avant-G web page
- Avant-X web page
- Lattice Radiant Software web page
- Lattice Propel Design Environment web page
- Lattice Insights for Lattice Semiconductor training courses and learning plans

## Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

# Revision History

**Revision 1.0, December 2023**

| Section | Change Summary |
|---|---|
| All | Initial release. |

www.latticesemi.com