System Memory Module

IP Version: v2.5.0

User Guide

FPGA-IPUG-02073-2.4

December 2025

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

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AHB-L	Advanced High-Performance Bus – Lite
AMBA	Advanced Microcontroller Bus Architecture
AXI	Advanced eXtensible Interface
DUT	Device Under Test
EBR	Embedded Block RAM
ECC	Error Correction Code
FIFO	First In, First Out
FPGA	Field Programmable Grid Array
GUI	Graphical User Interface
HDL	Hardware Description Language
IP	Intellectual Property
JTAG	Joint Test Action Group
LRAM	Large Random Access Memory
LSE	Lattice Synthesis Engine
PDF	Physical Design Constraints
RAM	Random Access Memory
RISC-V	Reduced Instruction Set Computer – V
ROM	Read-Only Memory
RTL	Register Transfer Level
SDK	Software Development Kit
SoC	System on Chip

1. Introduction

This document provides technical information about System Memory Soft IP and aims to provide essential information for IP/system developers in verification, integration, testing, and validation.

1.1. Overview of the IP

The System Memory Module is designed to support external source programming, offering user-interface options such as AHBL and AXI for efficient instruction fetching of the microcontroller.

1.2. Quick Facts

Table 1.1. Summary of the System Memory Module IP

IP Requirements	Supported FPGA Family	iCE40 UltraPlus™, MachXO2™, MachXO3™, MachXO3D™, MachXO4™, ECP5™, ECP5-5G™, CrossLink™-NX, Certus™-NX, CertusPro™-NX, Mach™-NX, MachXO5™-NX, Lattice Avant™, Certus™-N2.
	IP Changes[1]	For a list of changes to the IP, refer to the System Memory Module IP Release Notes (FPGA-RN-02065).
Resource Utilization	Targeted Devices	Refer to Table A.1
	Supported User Interface	AXI, AHBL
Design Tool Support	Lattice Implementation	IP Core v2.5.0 – Lattice Radiant™ Software 2025.2 IP Core v2.5.0 – Lattice Propel™ Builder Software 2025.2 IP Core v2.5.0 – Lattice Diamond™ Software 3.13
	Synthesis	Lattice Synthesis Engine (LSE) Synopsys® Synplify Pro for Lattice
	Simulation	For a list of supported simulators, see the Lattice Radiant Software User Guide

Note:

1. In some instances, the IP may be updated without changes to the user guide. This user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.

1.3. IP Support Summary

Table 1.2. System Memory Module IP Support Readiness

Device Family	IP	User-Interface	Data Width	Memory Type	Data Rate (Mbps)	Radiant Timing Model	Hardware Validated
Lattice Avant[1]	System Memory Module	AHBL	X8	EBR	1400	Preliminary	Yes
			X16	EBR	2800	Preliminary	Yes
			X32	EBR	5800	Preliminary	Yes
		AXI4	X8	EBR	1950	Preliminary	Yes
			X16	EBR	3800	Preliminary	Yes
			X32	EBR	7700	Preliminary	Yes
			X64	EBR	15600	Preliminary	Yes
CertusPro-NX[2]	System Memory Module	AHBL	X8	EBR	700	Preliminary	Yes
			X16	EBR	1400	Preliminary	Yes
			X32	EBR	2900	Preliminary	Yes
				LRAM	2900	Preliminary	Yes
		AXI4	X8	EBR	975	Preliminary	Yes
			X16	EBR	1950	Preliminary	Yes
			X32	EBR	3800	Preliminary	Yes

Device Family	IP	User-Interface	Data Width	Memory Type	Data Rate (Mbps)	Radiant Timing Model	Hardware Validated
				LRAM	3800	Preliminary	Yes
			X64	EBR	7700	Preliminary	Yes
Certus-NX[2]	System Memory Module	AHBL	X8	EBR	700	Preliminary	No
			X16	EBR	1400	Preliminary	No
			X32	EBR	2900	Preliminary	No
				LRAM	2900	Preliminary	No
		AXI4	X8	EBR	975	Preliminary	No
			X16	EBR	1950	Preliminary	No
			X32	EBR	3800	Preliminary	No
				LRAM	3800	Preliminary	No
			X64	EBR	7700	Preliminary	No
Mach-NX[2]	System Memory Module	AHBL	X8	EBR	700	Preliminary	Yes
			X16	EBR	1400	Preliminary	Yes
			X32	EBR	2900	Preliminary	Yes
				LRAM	2900	Preliminary	Yes
		AXI4	X8	EBR	975	Preliminary	Yes
			X16	EBR	1950	Preliminary	Yes
			X32	EBR	3800	Preliminary	Yes
				LRAM	3800	Preliminary	Yes
			X64	EBR	7700	Preliminary	Yes
ECP5[2]	System Memory Module	AHBL	X8	EBR	700	Preliminary	No
			X16	EBR	1400	Preliminary	No
			X32	EBR	2900	Preliminary	No
		AXI4	X8	EBR	975	Preliminary	No
			X16	EBR	1950	Preliminary	No
			X32	EBR	3800	Preliminary	No
			X64	EBR	7700	Preliminary	No

Notes:

1. The Lattice Avant data rate was tested using 200 MHz clock frequency.
2. The CertusPro-NX, Certus-NX, ECP5, Mach-NX data rate was tested using 100 MHz clock frequency.

1.4. Features

Key features of the System Memory Module IP include:

- Compliant with AMBA 3 AHB-Lite Protocol v1.0
- Compliant with AMBA AXI4 Protocol
- Supports AXI4 atomic access
- Configurable as single or dual port memory, utilizing 1 or 2 AHB-Lite or AXI4 Interfaces
- Core memory can be implemented as EBR, Distributed RAM, or Large RAM
- Supports ROM and RAM mode
- Supports byte writes when used with compatible hardware
- Supports up to 1 Mb maximum memory (maximum varies per device and per memory implementation)
- Supports 8, 16, or 32-bit data word transfers
- Supports 64-bit data word transfers for AXI interface
- Uses Little-endian bit structure
- Has a dedicated high-speed interface for fast memory initialization using either FIFO Interface or AXI4-Stream

1.5. Licensing and Ordering Information

The System Memory Module IP is provided at no additional cost with the Lattice Radiant software.

1.6. Hardware Support

Refer to the Example Design section for more information on the boards used.

1.7. Minimum Device Requirements

There is no speed grade limitation for using the System Memory IP. However, the maximum clock frequency of the IP depends on the device used.

1.8. Naming Conventions

1.8.1. Nomenclature

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

1.8.2. Signal Names

- _*n* is active low (asserted when value is logic 0)
- _*i* are input signals
- _*o* are output signals

2. Functional Description

2.1. IP Architecture Overview

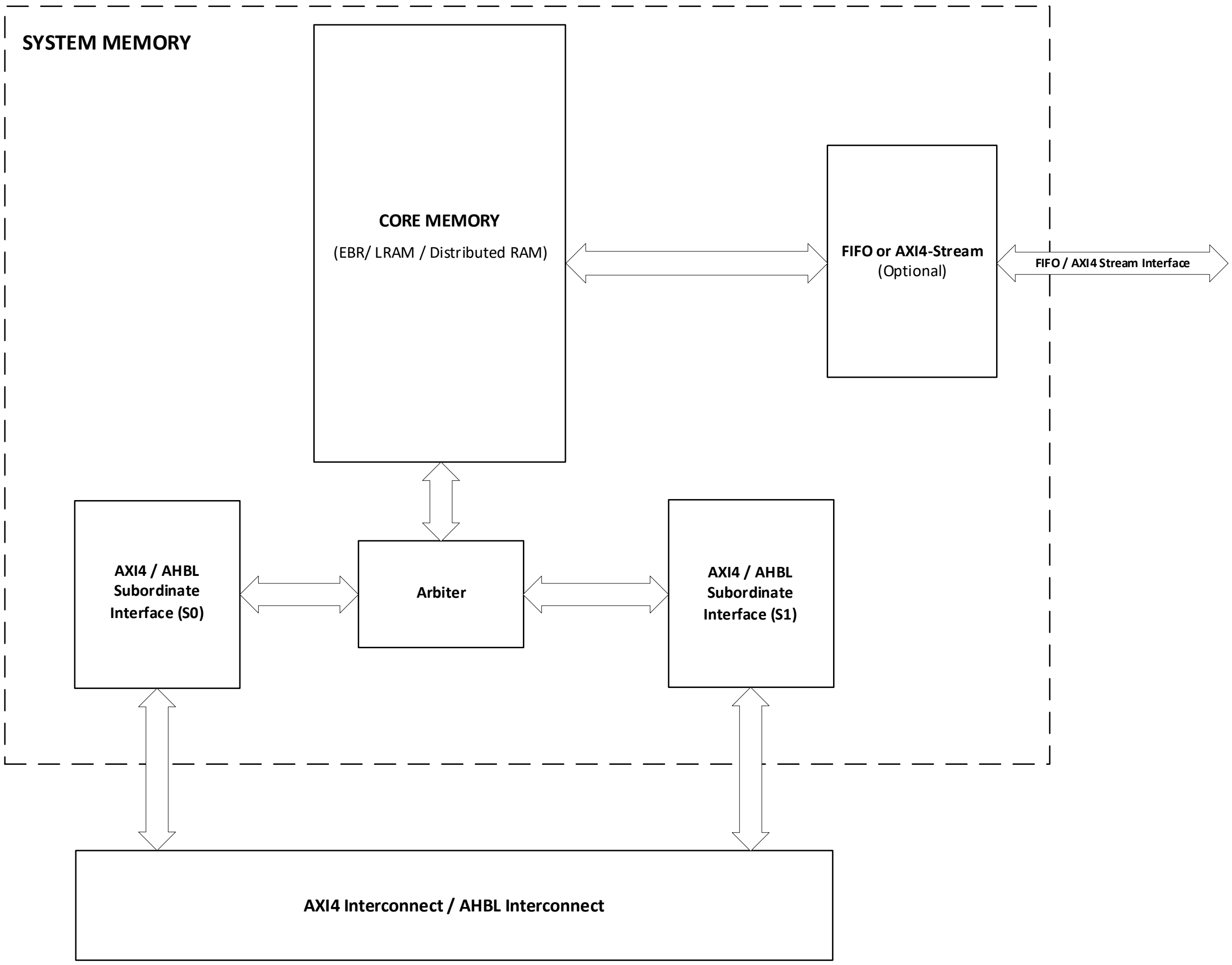

Figure 2.1. System Memory Module IP Core Block Diagram

The System Memory Module IP includes the following blocks:

- Core Memory
- AXI4/AHBL Subordinate Interface
- Arbiter
- FIFO or AXI4-Stream (Optional)

2.2. Clocking

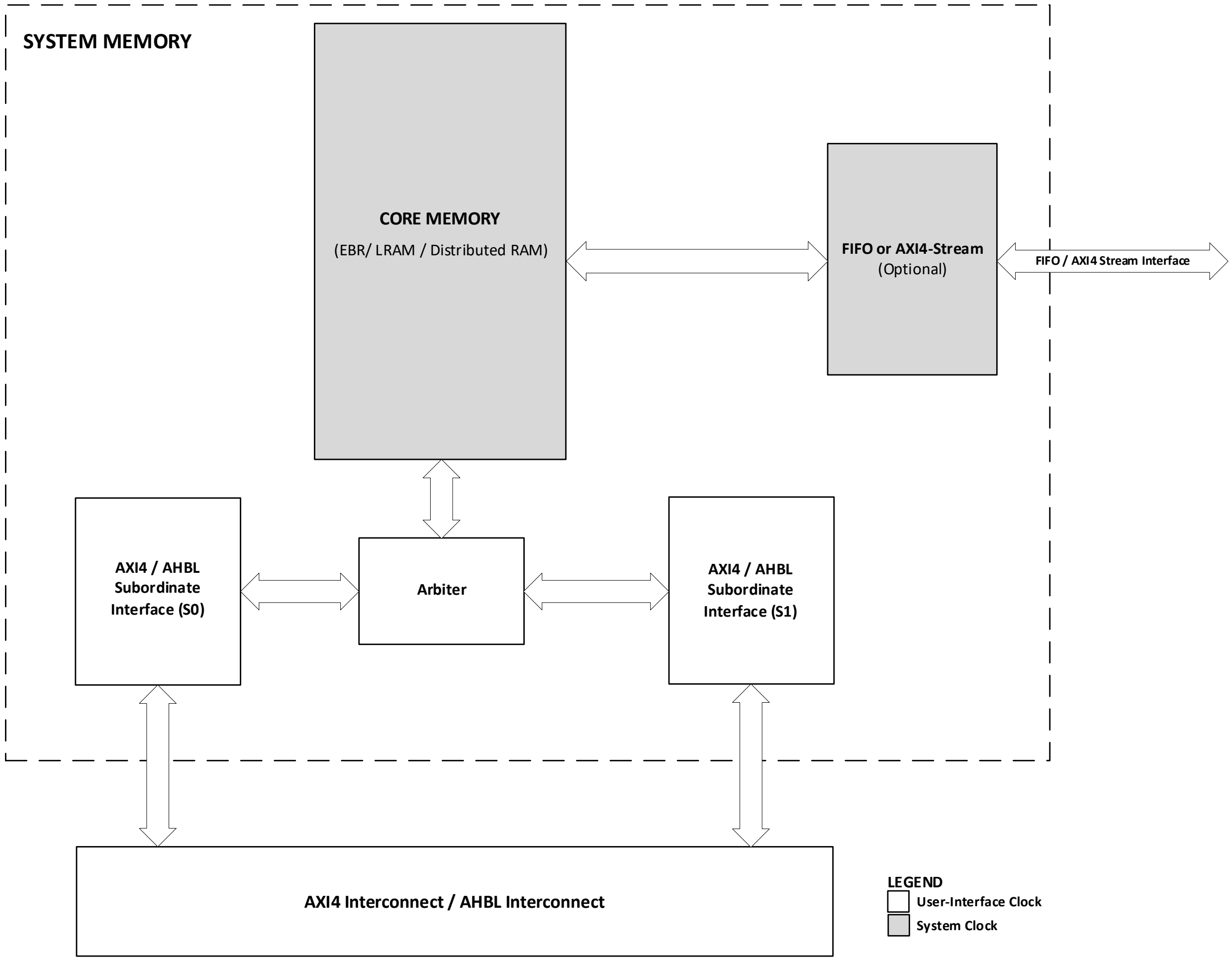

Figure 2.2. System Memory Module IP Clock Domain Block Diagram

2.2.1. Clocking Overview

- User-Interface Clock: ahbl_clk_i/ axi_aclk_i
    - For AXI4, the supported frequency ranges from 1 MHz to 125 MHz.
    - For AHBL, the supported frequency ranges from 1 MHz to 125 MHz.
    - For Lattice Avant devices, the supported frequency range for both AHBL and AXI4 is up to 200 MHz.
- If you enable the data streamer clock bypass, the system clock uses the user interface clock.

2.3. Reset

2.3.1. Reset Overview

The System Memory IP Core has only one active-low reset. When you use the AXI interface, the active-low reset is named axi_resetn_i. When you use the AHBL interface, the active-low reset is named ahbl_hresetn_i.

The reset input signal affects only the registers in the System Memory IP. It does not affect the registers in the HARD IP (memory implementation). You must wait for at least two clock cycles before initiating transactions after the reset sequence. This allows the IP to complete its reset sequence.

Below are the steps in the reset sequence for the System Memory IP Core:

1. De-assert the active-low reset using axi_resetn_i for the AXI interface and ahbl_hresetn_i for the AHBL interface.
2. Wait for two or more system clock cycles. The clock name is axi_aclk_i for the AXI interface and ahbl_hclk_i for the AHBL interface.
3. Initiate the next transaction.

2.3.2. Reset Timing Diagram

Figure 2.3 illustrates the timing for the start of the next transaction.

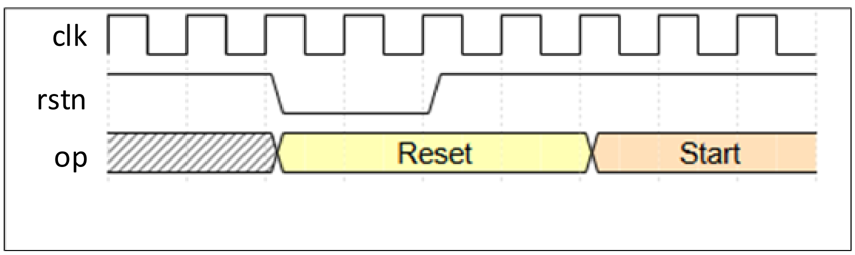

Figure 2.3. Reset Timing Diagram

2.4. User Interfaces

Table 2.1. User Interfaces and Supported Protocols

User Interface	Supported Protocols	Description
Control	AHB-Lite	The System Memory Module is fully compatible with the AHB-Lite standard. You can configure it as single or dual AHB-Lite interfaces, depending on whether you need single or dual port memory.
Control	AXI	The System Memory Module is fully compatible with the AXI4 standard. Unlike the AHB-Lite interface, you always implement AXI4 as a dual port memory. You assign one port of the memory to AXI4 Write Channels and the other port to AXI4 Read Channels. Similar to the AHB-Lite interface, AXI4 transactions translate into memory-compatible signals that the core memory directly interprets.
Control	FIFO	The AHB-L port S1 shares a dedicated FIFO interface. You can use this interface to inject data from a FIFO stream. LIFCL, LFCPNX, LFD2NX, and LFMXO5 devices support this feature.
Control	AXI4-Streamer	When you set the interface for System Memory to AXI4, you can set the data streamer interface to AXI4-Stream.

2.4.1. AHB-Lite

The AHBL interface for System Memory IP supports the INCR burst type for *write* and *read* operations. The WRAP burst type is currently unsupported.

For LFMXO4 devices, unaligned address and narrow transactions are only supported in 32-bit data width.

2.4.2. AXI4

The AXI4 interface for the System Memory IP supports the INCR burst type for both *write* and *read* operations. The WRAP burst type is not supported.

For LFMXO4 devices, unaligned address and narrow transactions are only supported in 32-bit data width.

2.4.3. FIFO

You typically use this to upload firmware values to the core memory. The FIFO starts writing at the designated first byte-addressable data and writes up to the maximum depth of the implemented memory.

2.4.4. AXI4 Stream

This interface is fully compatible with the AXI4-Stream standard and is implemented to prioritize other AXI4 write transactions when a valid data stream is given. Like the FIFO interface, the data starts at the designated first byte-addressable data and writes up to the maximum memory depth. The You can use the TLAST signal to indicate the end of the data stream, then the following AXI4-Stream transaction starts again at the designated first address. The AXI4-Stream write strobe is ignored when using Distributed RAM.

2.5. Memory Implementation

The System Memory Module uses Embedded Block RAMs (EBR) or Distributed RAM in the MachXO3D family devices, as well as Large RAM in LIFCL, LFCPNX, LFD2NX, and LFMXO5 family devices. You can configure the memory implementation as true-dual port, pseudo dual port, single port, or read-only memory. The number of ports and read/write configuration of the System Memory Module automatically select the best type of memory for the user-selected application.

Remember that the memory is not affected by the reset. All written data is stored even after the reset sequence.

Table 2.2. System Core Memory Type – AHB-Lite

Memory Type	AHB-Lite Configuration Used[1]	LatticeECP3, ECP5, ECP5-5G, MachXO2, MachXO3, MachXO3D, MachXO4[2], Lattice Avant, Certus-N2	Crosslink-NX, CertusPro-NX, Certus-NX, MachXO5-NX	iCE40 UltraPlus
LRAM	1 port: “R/W, R/O”	—	Yes	—
	2 ports: “R/W” + “R/W” “R/O” + “R/W”			
EBR[1]	1 port: “R/W, R/O”	Yes	Yes	Yes
	2 ports: “R/W” + “R/W” “R/O” + “R/W”			—
Distributed RAM	1 port:	Yes	Yes	—

Memory Type	AHB-Lite Configuration Used[1]	LatticeECP3, ECP5, ECP5-5G, MachXO2, MachXO3, MachXO3D, MachXO4[2], Lattice Avant, Certus-N2	Crosslink-NX, CertusPro-NX, Certus-NX, MachXO5-NX	iCE40 UltraPlus
	“R/W, R/O”			

Notes:

1. The EBR ECC is limited to single port only.
2. ECC is not supported.

Table 2.3. System Core Memory Type – AXI

Memory Type	AXI Configuration Used	LatticeECP3, ECP5, ECP5-5G, MachXO2, MachXO3, MachXO3D, MachXO4[2], Lattice Avant[2], Certus-N2	Crosslink-NX, CertusPro-NX, Certus-NX, MachXO5-NX	iCE40 UltraPlus[2]
LRAM	1 port: “R/W, R/O”	—	Yes	—
	2 ports: “R/W” + “R/W” “R/O” + “R/W”			
EBR[1]	1 port: “R/W, R/O”	Yes	Yes	—
	2 ports: “R/W” + “R/W” “R/O” + “R/W”			
Distributed RAM	1 port: “R/W, R/O”	Yes	Yes	—

Notes:

1. The EBR ECC is limited to single port only.
2. ECC is not supported.

Table 2.4. System Core Memory Implementation

Memory Type	User Interface	Access Type	Memory Implementation
LRAM	AHBL	1 port: "R/W", "R/O"	Single Port
		2 ports: "R/W" + "R/W" "R/O" + "R/W"	Dual Port
	AXI	1 port: "R/W", "R/O"	Pseudo Dual Port
		2 ports: "R/W" + "R/W" "R/O" + "R/W"	Pseudo Dual Port
EBR	AHBL	1 port: "R/W", "R/O"	Single Port
		2 ports: "R/W" + "R/W" "R/O" + "R/W"	Dual Port
	AXI	1 port: "R/W", "R/O"	Pseudo Dual Port
		2 ports: "R/W" + "R/W" "R/O" + "R/W"	Pseudo Dual Port
Distributed RAM	AHBL	1 port: "R/W", "R/O"	Single Port
	AXI	1 port: "R/W", "R/O"	Pseudo Dual Port

Table 2.5. Features Supported per Memory Block

Device	LRAM	EBR	Distributed RAM
ECC[1]	Yes[2]	Yes[2]	No
Memory Initialization	Yes	Yes	Yes
Registered Output	Yes	Yes	Yes
Dual Port Configuration	Yes	Yes	Yes
Byte-Enable	Yes[2]	Yes[2]	No
Unaligned Read Access	Yes[3]	Yes	No

Notes:

1. Lattice Avant devices do not support ECC.
2. You can use Byte-enable with ECC.
3. You cannot use unaligned read access with Byte-enable.

Table 2.6. ECC Implementation per Memory Block

Memory Type	ECC Implementation
LRAM	Hard IP[1,4]
EBR	Hard IP[1,2,3,4,5,6]
Distributed RAM	No

Notes:

1. Available in CrossLink-NX, CertusPro-NX, Certus-NX, and MachXO5-NX devices.
2. Available in LatticeECP3, ECP5, ECP5-5G, MachXO2, MachXO3, MachXO3D, Lattice Avant, and Certus-N2 devices.
3. Available in iCE40 UltraPlus devices.
4. You cannot use the ECC function in AXI4 because Byte-enable always supports AXI4 write strobes.
5. You cannot use the ECC function in AXI4 because it always implements dual port RAM in EBR.
6. The EBR ECC function is available only when the port count is equals one.

Table 2.7. Allowable Combination of Features for System Memory when INTERFACE = AHBL

Device	Byte-Enable	ECC	Unaligned Read Access	FIFO	Maximum Supported Port Count
LRAM	×	×	×	×	2
	×	×	×	✓	2
	×	×	✓	×	2
	×	×	✓	✓	2
	×	✓	×	×	2
	×	✓	×	✓	Not Supported
	×	✓	✓	×	2
	×	✓	✓	✓	Not Supported
	✓	×	×	×	2
	✓	×	×	✓	2
	✓	×	✓	×	2
	✓	×	✓	✓	2
	✓	✓	×	×	Not Supported
	✓	✓	×	✓	Not Supported
	✓	✓	✓	×	Not Supported
	✓	✓	✓	✓	Not Supported
EBR	×	×	N/A	×	2
	×	×	N/A	✓	2
	×	✓	N/A	×	1[1]
	×	✓	N/A	✓	Not Supported
	✓	×	N/A	×	2
	✓	×	N/A	✓	2
Distributed RAM	N/A	N/A	N/A	×	1[2]
	N/A	N/A	N/A	✓	1[2]

Notes:

1. EBR ECC supports only a single port.
2. Distributed RAM supports only a single port.

2.6. System Memory Error Information

Table 2.8. System Memory Error

Error	Description	AHB-Lite Behavior	AXI4 Behavior
Dual Write	Occurs when two ports attempt to write to the same address of the memory.	The system prioritizes port S0 and ignores port S1 transaction without generating an error.	The system does not generate an error. An arbiter decides which AXI Interface (S0 or S1) grants access.
Illegal Access	Occurs when a manager attempts to access an address outside the bounds of the START_ADDRESS or END_ADDRESS parameter.	The system generates a bus error through the hresp_o port.	The system generates a bus error through the bresp_o or rresp_o port.
Illegal Transaction (W/R)	Occurs when a manager attempts to write to a read-only port, or read from a write-only port.	The system generates a bus error through the hresp_o port.	The system generates a bus error through the bresp_o or rresp_o port.

Error	Description	AHB-Lite Behavior	AXI4 Behavior
Unaligned Error	Occurs when a manager attempts to access a wider data bit without providing appropriate pads for the lower address bits. Example: A 32-bit data access with ahbl_addr[1:0] != 2'b00.	The system generates a bus error through the hresp_o port.	The system does not generate an error. The AXI4 Interface supports unaligned access.
ECC error	Occurs when an ECC error is generated during a read attempt. • ecc_sec – a single error is detected and corrected. • ecc_dec – two errors are detected and cannot be corrected.	The system generates a bus error through the hresp_o port.	The AXI4 Interface does not support ECC.
Multiple AXI4 Exclusive Read	Occurs when the AXI4 manager issues an exclusive read when there is already a standing exclusive transaction with different transaction ID in the subordinate.	—	The AXI4 subordinate generates a subordinate error in the rresp port.

2.7. Arbitration

When the System Memory Module has two ports (S0 and S1), both ports can access the memory at the same time only if the ports target different addresses or if both ports are performing read operations.

If both ports access the same address and at least one is a write, the module enforces arbitration to prevent data corruption. In such cases, only one port is granted access, and port S0 always has priority over port S1.

In the System Memory Module, port S0 is always prioritized in every arbitration.

2.7.1. Arbitration in the AXI4 Interface

To manage the separate write and read channels of the AXI4 Interface, the implemented memory is always pseudo-dual port, with port A in the primitive for write transactions and port B in the primitive for read transactions. When the port count is two, an arbiter block manages the arbitration between the write channels of port S0 and port S1 using WREADY signals. Separate arbitration manages the read channels using the ARREADY signal.

For example, port S0 is doing a write transaction in address 0x0200 and port S1 attempts to request a read transaction to the same address. The RREADY signal for port S1 is delayed until the transaction in port S0 finishes. Refer to Figure 2.4 for the illustration of the example scenario.

Figure 2.4. Dual Port AXI Interface (Read Arbitration)

For the scenario when two ports are sending the same transaction (read/write), port S0 is doing a write transaction in address 0x0200 and port S1 attempts to request a write transaction. The WREADY signal for port S1 is delayed until the transaction in port S0 finishes. Refer to Figure 2.5 for the illustration of the example scenario.

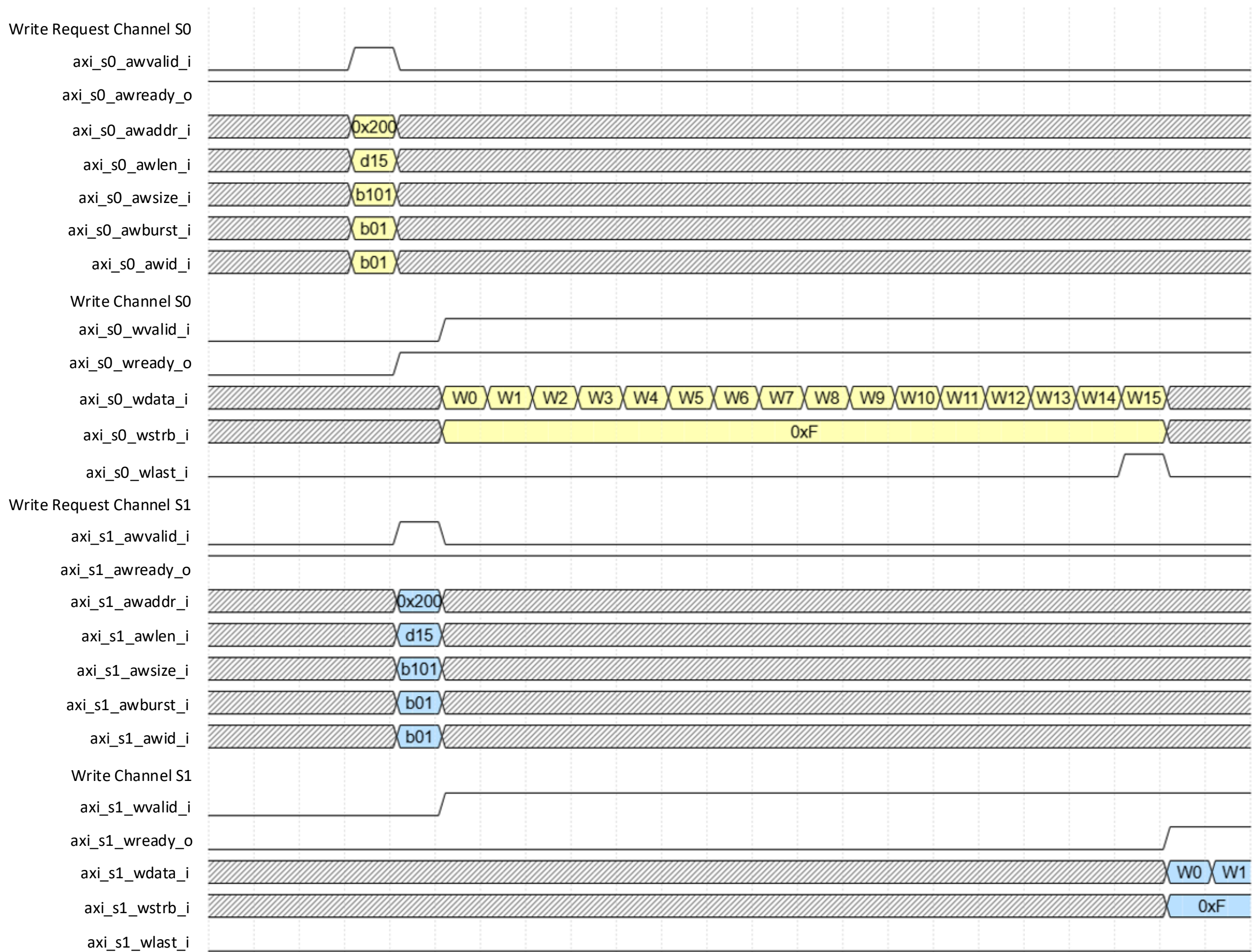

Figure 2.5. Dual Port AXI Interface (Write Arbitration)

2.7.2. Arbitration in the AHBL Interface

To manage the arbitration in the AHBL interface, the HREADY signal of the port that loses the arbitration is delayed indicating that the subordinate cannot handle transactions as the other port has an ongoing write transaction at the same address. Refer to

Figure 2.6 and Figure 2.7 for the illustration of the example scenarios.

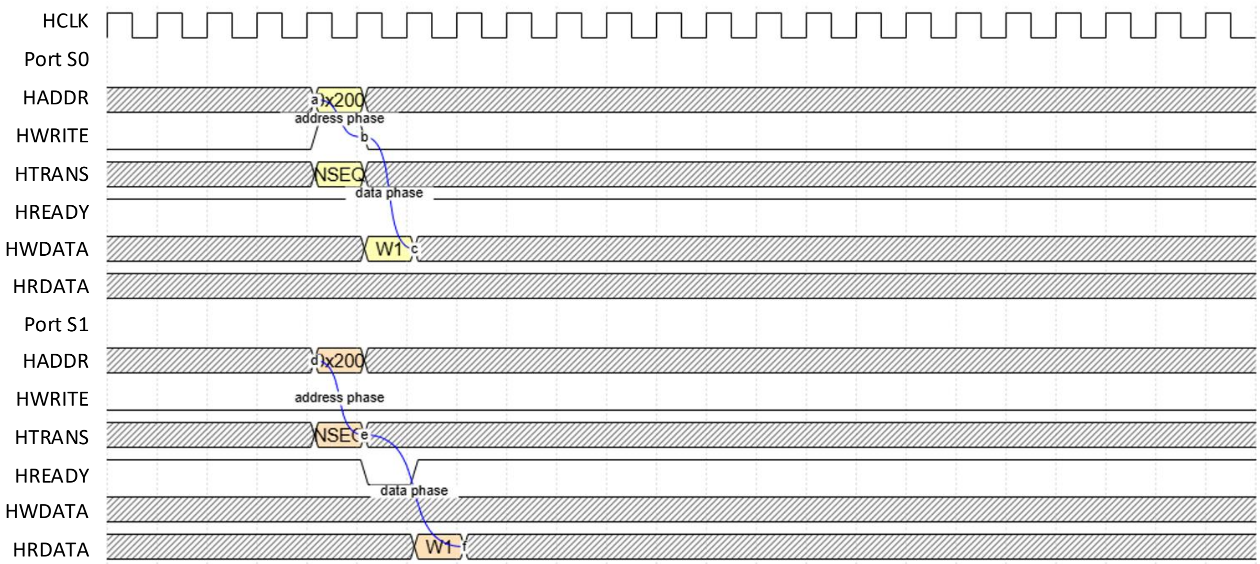

Figure 2.6. Dual Port AHBL Interface Arbitration (Read Transaction)

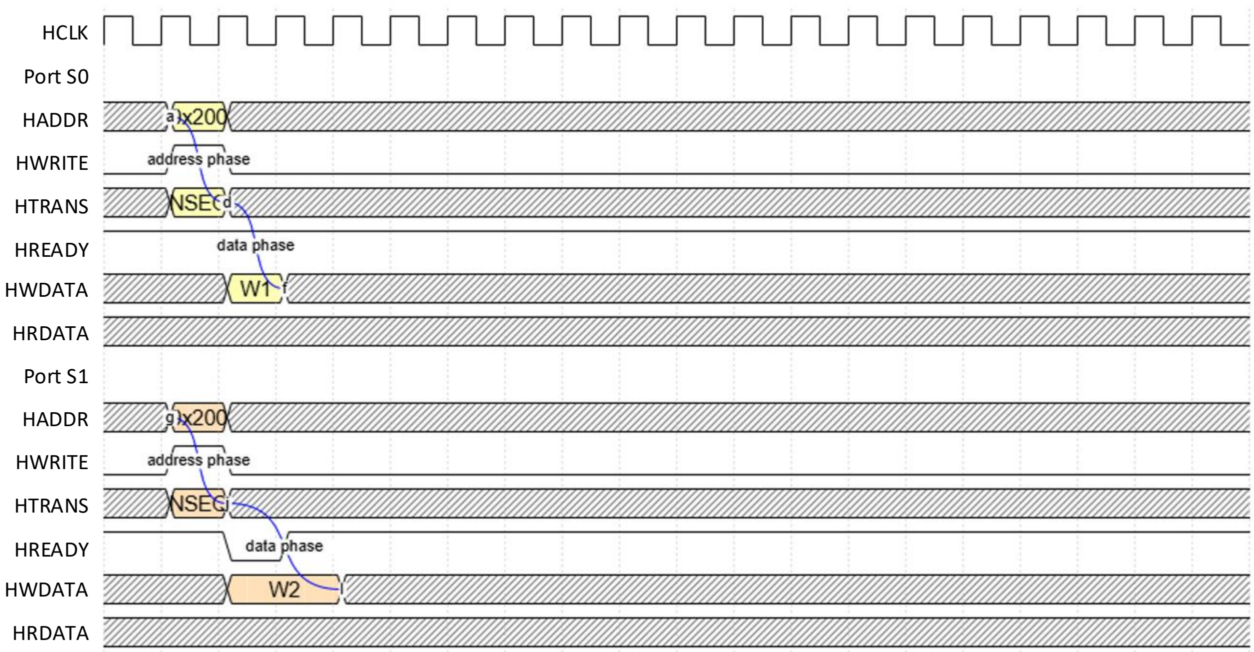

Figure 2.7. Dual Port AHBL Interface Arbitration (Write Transaction)

2.8. Initialization Format

You can create or edit the initialization file, an ASCII file, using any ASCII editor. The Module/IP Block Wizard supports the following memory file formats:

- Binary file
- Hex File

The memory initialization file is *.mem (<file_name>.mem). Each row stores the value for a specific memory location. The number of characters (or columns) represents the number of bits for each address (or the memory module width).

The memory initialization can be static or dynamic. For static initialization, the memory values are stored in the bitstream. Dynamic memory initialization involves storing memory values in the external flash which user logic can update knowing the EBR address locations. The bitstream (bit or rbt file) size is larger due to stored static values.

The initialization file is used when the System Memory is configured as a ROM. In RAM configuration, you can also use the initialization file to preload memory contents.

Binary File

The binary file contains 0s and 1s. The rows represent the number of words, and the columns represent the memory width.

Memory Size 20 x 32

```
00100000001000000010000001000000
00000001000000010000000100000001
00000010000000100000001000000010
00000011000000110000001100000011
00000100000001000000010000000100
00000101000001010000010100000101
00000110000001100000011000000110
00000111000001110000011100000111
00001000010010000000100001001000
00001001010010010000100101001001
00001010010010100000101001001010
00001011010010110000101101001011
00001100000011000000110000001100
00001101001011010000110100101101
00001110001111100000111000111110
00001111001111110000111100111111
00010000000100000001000000010000
00010001000100010001000100010001
00010010000100100001001000010010
00010011000100110001001100010011
```

Hex File

The hex file contains hexadecimal characters arranged in a similar row-column format. The number of rows matches the number of address locations, with each row representing the content of the memory location.

Memory Size 8 x 16

```
A001
0B03
1004
CE06
0007
040A
0017
02A4
```

2.9. AXI4 Atomic Access

The System Memory Module supports AXI4 atomic access. This section provides the details on how to use the atomic access feature.

2.9.1. Exclusive Access Restrictions

The System Memory Module has the following restrictions:

- The address must be aligned with the total number of bytes in the transaction (AxSIZE x *transaction length*).
- The number of bytes must be power of 2.
- The maximum number of bytes in a transfer is 128.

- The length must not exceed 16.
- Exclusive read pipelining is not supported. Subordinate error response is sent if attempted.
- AxSIZE must match the data bus width.
- WSTRB must be fully asserted.
- Fixed burst is not supported.

Note that not following these restrictions may result in unexpected behavior from the System Memory Module.

To complete an exclusive transaction, the exclusive read and exclusive write must have the same request information. Table 2.9 lists the supported signals that must be the same in an exclusive sequence.

Table 2.9. Request Information Atomic Access

Signal Name	Supported Signal
AxID	Yes
AxLEN	Yes
AxADDR	Yes
AxSIZE	Yes
AxBURST	Yes
AxLOCK	Yes (v2.5.0)

2.9.2. Exclusive Access Sequence

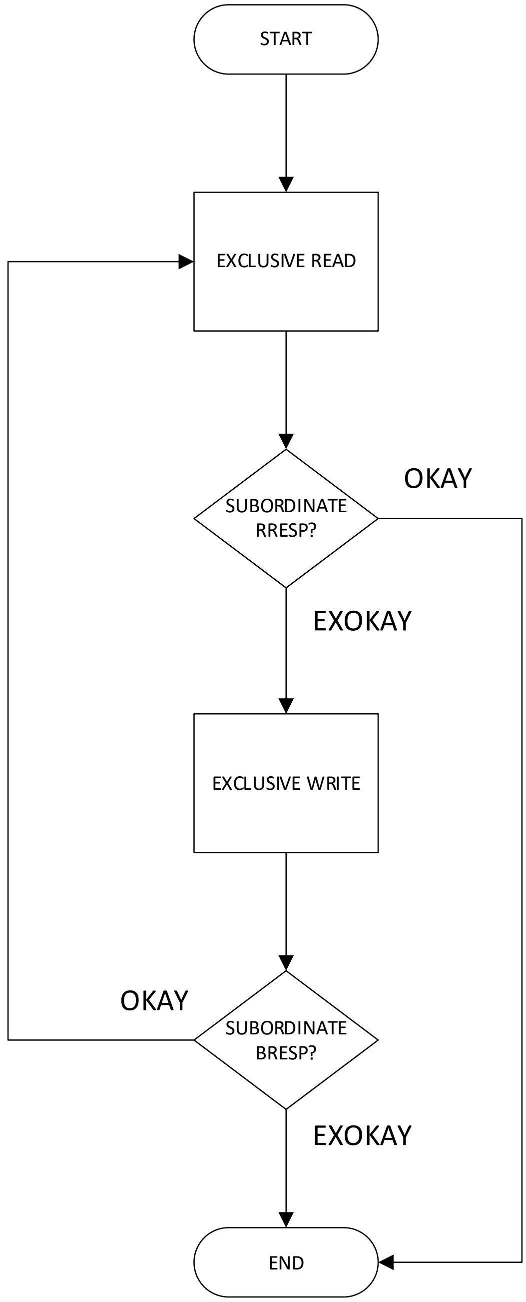

Figure 2.8. Exclusive Access Sequence

1. A manager issues an exclusive read request. The manager can resend another exclusive read using the same ARID to reset the monitored address for exclusive access.
2. At some later time, the manager attempts to complete the exclusive operation by issuing an exclusive write request to the same address, with an AWID that matches the ARID used for the exclusive read.
   - Successful, if no other manager has written to that location since the exclusive read access. In this case, the exclusive write updates memory.
   - Fails, if another manager has written to that location since the exclusive read access or the monitored address is updated. In this case, the memory location is not updated.

2.9.3. Exclusive Access Timing Diagram

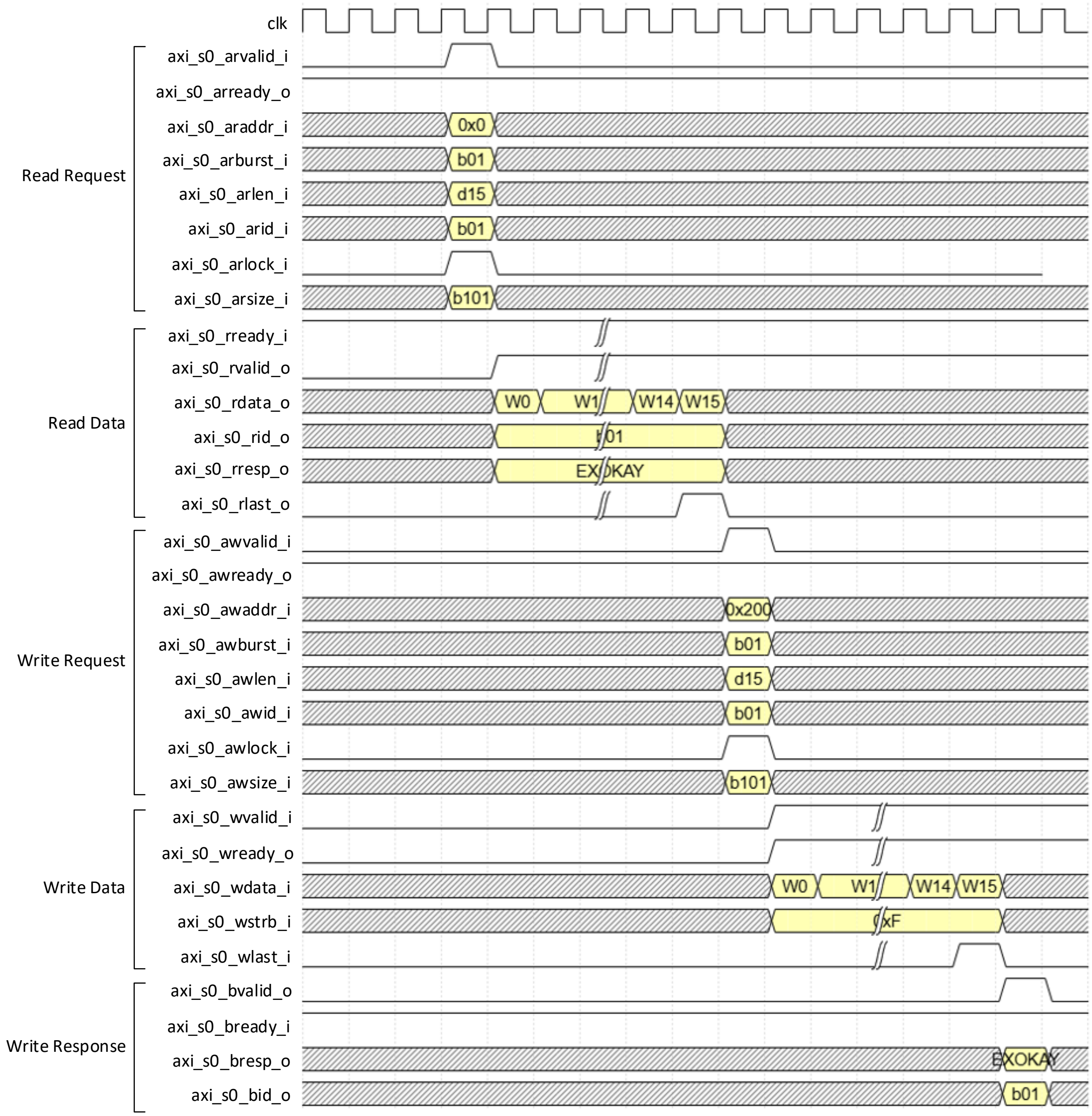

Figure 2.9. Successful Atomic Access Transaction

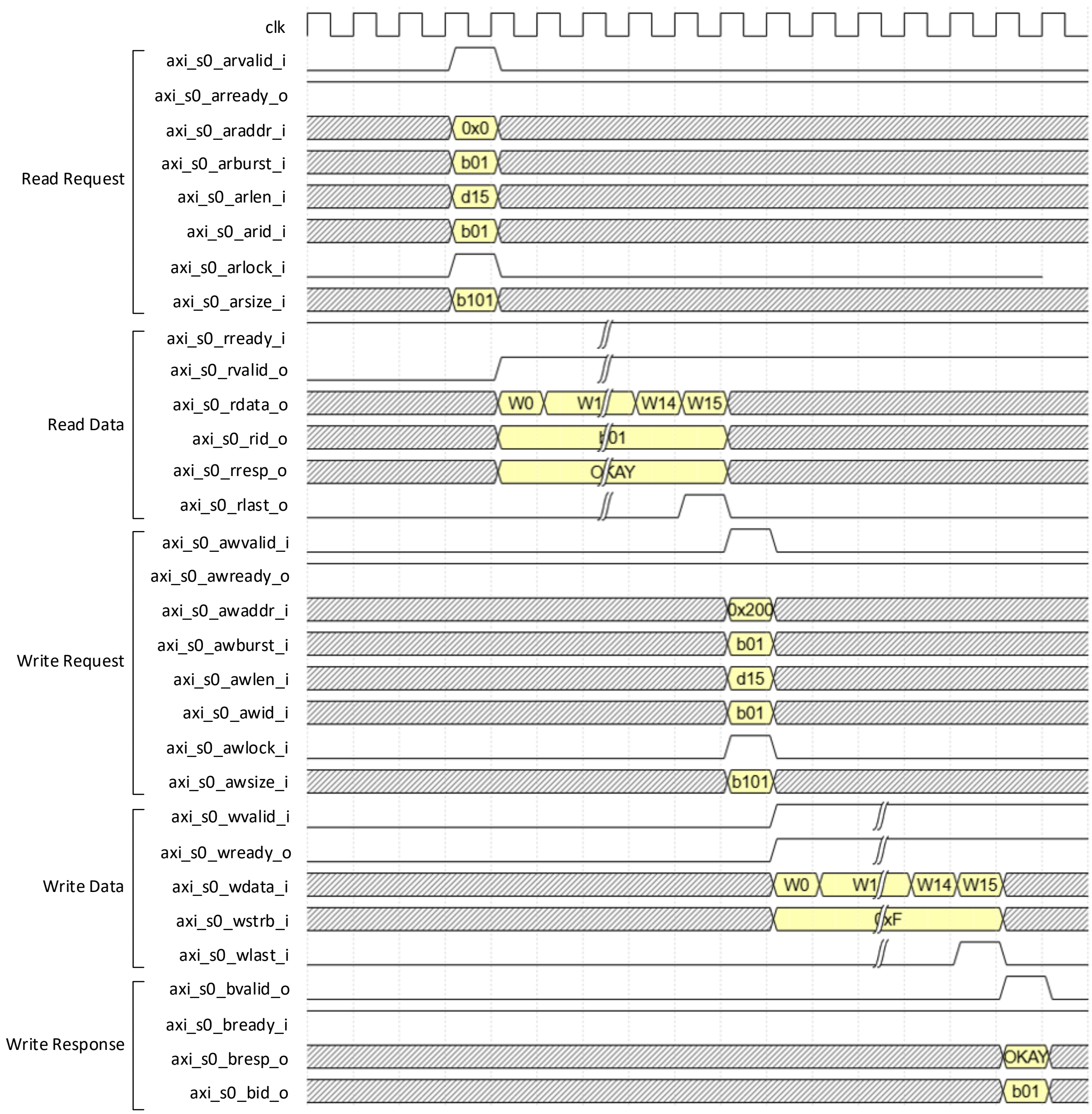

Figure 2.10. Failed Atomic Access Transaction

2.9.4. Exclusivity Granularity

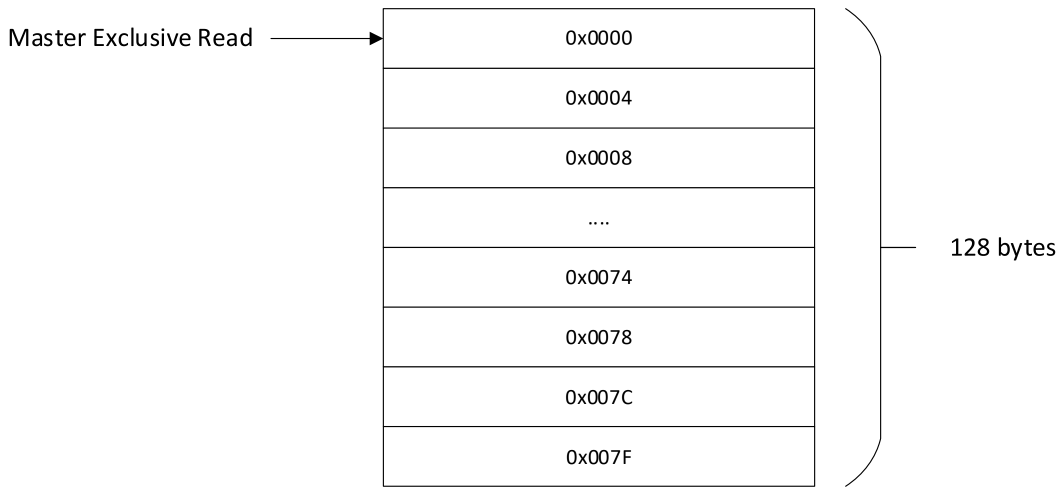

Figure 2.11. Atomic Access Exclusivity Granularity

In Figure 2.11, the AXI4 manager sends an exclusive read that starts in address 0x0000 and has a length of 128 bytes. All addresses from 0x0000 to 0x007F are now locked in exclusive access to that specific AXI4 manager. Any write transaction that modifies these addresses causes the exclusive write transaction to fail.

3. IP Parameter Description

The tables below show the configurable attributes of the System Memory IP. You can configure the IP by setting the attributes in the IP Catalog's Module/IP wizard within the Lattice Radiant software.

Default values, where applicable, are highlighted in bold.

3.1. General

Table 3.1. General Attributes

Attribute	Selectable Values	Description
Interface	**AHBL** AXI4	Select the subordinate bus interface.
Memory Address Depth	1 – 456,000	Measure the depth of the memory address in units of DATA_WIDTH. The maximum ADDR_DEPTH depends on DATA_WIDTH, memory type, and device used.
Data Bus Width(bits)	8, 16, **32,** 64	The data width of the memory measured in bits. The value of 64 bits is available only if the interface is AXI4 and the target device has two or more LRAM blocks available.
Memory Type	**EBR** Distributed_RAM LRAM	Select the type of memory implemented for this instance of system memory.
Port Count [1 -2]	**1,** 2	Determine whether the generated system memory uses one or two AHB-Lite/AXI Interfaces.
ECC Enable[1,2,3]	True, **False**	Determines whether ECC is used, which applies to both ports. Refer to Table 2.5 for details. Note that the data width must be 32 bits when using ECC.
Enable Arbiter	True, **False**	Enable the ARBITER function. Editable only when *PORT_COUNT* is two.
AXI4 ID Width	1 - 15	The width of the AXI4 ID for all channels. Available only when the interface is *AXI4*.
Data Streamer		
Enable Data Streamer	True, **False**	Enable the data streamer interface.
Enable Data Streamer == TRUE		
Data Streamer Interface	**Generic FIFO** AXI4 Stream	Select the interface for the data streamer. The AXI4-Stream is available only when the selected bus interface is AXI4.
Data Streamer Write Start Address	**0** – (ADDR_DEPTH-(Data Bus Width/8))	The starting address where the data streamer begins writing.
Data Streamer Interface == AXI4 Stream		
Data Streamer Clock Bypass	True, **False**	When enabled, the data streamer uses the ahbl_clk_i or axi_aclk_i as its clock source. Otherwise, it uses its own dedicated clock, which is fifo_clk_i for Generic FIFO or axis_rx_aclk_i for AXI4-Stream.
Initialization		
Initialize Memory	True, **False**	Enable the initialization of the system memory by providing an initialization file.
Initialize Memory == TRUE		
Initialization File Format	**Hex,** Binary	Determine the file format of the initialization file.
Initialization File	<string>	Select the initialization file for the system memory.

Notes:

1. Apply this only to LRAM and EBR when the INTERFACE is AHBL. Limit the EBR ECC to a single port.
2. ECC is always disabled when data initializes the memory.
3. The *ECC Enable* attribute is not supported in LFMXO4 devices.

3.2. Port S0 Settings

Table 3.2. Port S0 Settings

Attribute	Selectable Values	Description
Enable Port S0 Memory Core Output Register	True, **False**	Apply a registered output from the memory core to subordinate S0 to improve timing in Place and Route.
Interface == AXI4		
Enable Atomic Access S0	True, **False**	Enable AXI4 atomic access in port S0.
Interface == AHBL and Memory Type == LRAM and Enable Port S0 Memory Core Output Register == true		
Enable Port S0 Read Pipeline[1]	True, **False**	Implement an additional register to pipeline AHBL read transactions for port S0.
Port S0 setting that are always visible		
Reset behavior for Port S0	Async, **Sync**	Set the reset mode for port S0 to *Sync* when the Memory Type is *LRAM*.
Byte Enable for Port S0[2,3,4]	**True,** False	Determine whether Byte Enable is used for port S0. Set it to True when the Access Type for Port S0 is R/O, the data width is not 8, or the interface is AHB-Lite. This enables narrow transactions in the AHBL interface.
Unaligned Access for Port S0[5]	True, **False**	Enable the unaligned read functionality. Set it to False when the data width is not 32 or the memory is not LRAM.
Unaligned access for Port S0 == True		
Unaligned Access Shift Direction (S0)[6]	Right, Left, **None**	Shift the direction of the unaligned read access.
Edit Address Range Port S0[7]	True, **False**	Enable the memory address offset for port S0. Set it to False when the port count is one.
Edit address range Port S0 == True		
Start Address Port S0 (hex)[7]	**0** – ADDR_DEPTH*DATA_WIDTH-1	Set the starting memory address offset for port S0.
End Address Port S0 (hex)2[7]	0 – **ADDR_DEPTH*DATA_WIDTH-1**	Set the ending memory address offset for port S0. It must be greater than the Start Address for Port S0.
Access Type for Port S0	**R/W,** R/O	Determine the access for port S0.

Notes:

1. This feature is available only when the Interface is AHBL, the Memory Type is LRAM, and the output register is enabled.
2. The byte-enable function is unavailable when ECC is enabled or when DATA_WIDTH is eight.
3. The byte-enable function is always enabled by default when the Interface is AXI4.
4. The byte-enable function is disabled when the interface is AHBL, and the Memory Type is Distributed RAM.
5. This feature is available only for LRAM and disables the BYTE_ENABLE function.
6. Edit this only if ENABLE_UNALIGNED_ACCESS is enabled.
7. Edit this only when PORT_COUNT is two.

3.3. Port S1 Settings

Table 3.3. Port S1 Settings

Port Count == 2		
Attribute	**Selectable Values**	**Description**
Enable Port S1 Memory Core Output Register	True, **False**	Apply a registered output from the memory core to subordinate S1 to improve timing in Place and Route.
Interface == AXI4		
Enable Atomic Access S1	True, **False**	Enable AXI4 atomic access in Port S1
Interface == AHBL and Memory Type == LRAM and Enable Port S1 Memory Core Output Register == true		
Enable Port S1 Read Pipeline[1]	True, **False**	Implement an additional register to pipeline AHBL read transactions for port S1.
Port S1 Setting that are always visible		
Reset behavior for Port S1	Async, **Sync**	Set the reset mode for port S1 to *Sync* when the Memory Type is *LRAM*.
Byte Enable for Port S1[2,3,4,8]	**True,** False	Determine whether Byte Enable is used for port S1. Set it to True when the Access Type for port S1 is R/O, the data width is not 8, or the interface is AHB-Lite. This enables narrow transactions in the AHBL interface.
Unaligned Access for Port S1[5]	True, **False**	Enable the unaligned read functionality. Set it to *False* when the data width is not 32 or the memory is not LRAM.
Unaligned Access for Port S1 == True		
Unaligned Access Shift Direction (S1)[6]	Right, Left, **None**	Shift the direction of the unaligned read access.
Edit Address Range Port S1[7]	True, **False**	Enable the memory address offset for port S1. Set it to False when the port count is 1.
Edit Address Range Port S1 == True		
Start Address Port S1 (hex)[7]	**0** – ADDR_DEPTH*DATA_WIDTH-1	Set the starting memory address offset for port S1.
End Address Port S1 (hex)2[7]	0 – **ADDR_DEPTH*DATA_WIDTH-1**	Set the ending memory address offset for port S1. It must be greater than the Start Address for Port S1.
Access Type for Port S1	**R/W**	Determine the access for port S1.

Notes:

1. This feature is available only when the Interface is AHBL, the Memory Type is LRAM, and the output register is enabled.
2. The byte-enable function is unavailable when ECC is enabled or when DATA_WIDTH is eight.
3. The byte-enable function is always enabled by default when the Interface is AXI4.
4. The byte-enable function is disabled when the interface is AHBL ,and the Memory Type is Distributed RAM.
5. This feature is available only for LRAM and disables the BYTE_ENABLE function.
6. Edit this only if ENABLE_UNALIGNED_ACCESS is enabled.
7. Edit this only when PORT_COUNT is two.
8. For LFMXO4 devices, the byte-enable function is always equal to the *Byte Enable for Port S0* attribute when the interface is AHB-Lite.

3.4. IP Parameter Settings for Example Use Cases

Table 3.4 shows the parameter settings for System Memory IP used for memory storage.

Table 3.4. IP Parameter Settings for Example Use Cases

Attribute	Value
General Attribute	
Interface	AXI4
Memory Address Depth	57344
Data Bus Width(bits)	64
Memory Type	LRAM
Port Count [1 -2]	2
ECC Enable	—
Enable Arbiter	✓
AXI4 ID Width	6
Enable Data Streamer	False
Data Streamer Interface	—
Data Streamer Write Start Address	(ADDR_DEPTH- Data Bus Width/8)
Data Streamer Clock Bypass	False
Initialize Memory	False
Initialization File Format	Hex
Initialization File	—
The attributes of Port S0 and Port S1 are set to *default*.	

4. Signal Description

This section describes the System Memory IP ports.

4.1. Clock Interface

Table 4.1. Clock Ports

Port	Type	Description
AHBL Interface		
ahbl_hclk_i	Input	• This is the input clock for the user clock domain • The frequency range for this is from 1 MHz to 125 MHz • For Lattice Avant devices, the frequency range extends up to 200 MHz
AXI Interface		
axi_aclk_i	Input	• This is the input clock for the user clock domain • The frequency range for this is 1 MHz to 125 MHz • For Lattice Avant devices, the frequency range extends up to 200 MHz

4.2. Reset Interface

Table 4.2. Reset Ports

Port	Type	Description
AHBL Interface		
ahbl_hresetn_i	Input	This input signal provides an active-low reset. De-asserting this signal resets the entire System Memory IP.
AXI Interface		
axi_resetn_i	Input	This input signal provides an active-low reset. De-asserting this signal resets the entire System Memory IP.

4.3. User-Interface

Table 4.3. AHBL Ports

Port	Type	Width	Description
ahbl_s0_hsel_i	Input	1	The device functions when the active HIGH signal asserts.
ahbl_s0_hready_i	Input	1	A HIGH signal indicates that no transfers are currently executing.
ahbl_s0_haddr_i	Input	32	It contains the address of the data to write or read.
ahbl_s0_hburst_i	Input	3	It determines whether the transfer is a single transfer or part of a burst.
ahbl_s0_hsize_i	Input	3	It indicates the transfer size.
ahbl_s0_hmastlock_i	Input	1	It indicates if the transfer is part of a locked sequence.
ahbl_s0_hprot_i	Input	4	It determines the type of burst used in the transfer.
ahbl_s0_htrans_i	Input	2	It indicates the current transfer type.
ahbl_s0_hwrite_i	Input	1	It indicates the transfer direction.
ahbl_s0_hwdata_i	Input	Data bus width	Input data for the system memory.
ahbl_s0_hreadyout_o	Output	1	A HIGH signal indicates that the system memory is busy.
ahbl_s0_hresp_o	Output	1	A HIGH signal indicates an error in the transfer.
ahbl_s0_hrdata_o	Output	Data bus width	Output data for the system memory.
ahbl_s1_hsel_i	Input	1	The device functions when the active HIGH signal asserts.

Port	Type	Width	Description
ahbl_s1_hready_i	Input	1	A HIGH signal indicates that no transfers are currently executing.
ahbl_s1_haddr_i	Input	32	It contains the address of the data to write or read.
ahbl_s1_hburst_i	Input	3	It determines whether the transfer is a single transfer or part of a burst.
ahbl_s1_hsize_i	Input	3	Indicates the transfer size.
ahbl_s1_hmastlock_i	Input	1	It indicates if the transfer is part of a locked sequence.
ahbl_s1_hprot_i	Input	4	It determines the type of burst used in the transfer.
ahbl_s1_htrans_i	Input	2	It indicates the current transfer type.
ahbl_s1_hwrite_i	Input	1	It indicates the transfer direction.
ahbl_s1_hwdata_i	Input	Data bus width	Input data for the system memory.
ahbl_s1_hreadyout_o	Output	1	A HIGH signal indicates that the system memory is busy.
ahbl_s1_hresp_o	Output	1	A HIGH signal indicates an error in the transfer.
ahbl_s1_hrdata_o	Output	Data bus width	Output data for the system memory.

Table 4.4. AXI Ports

Port	Type	Width	Description
AXI Subordinate Interface Port1			
axi_s0_awid_i	IN	AXI4 ID Width	The AXI4 write address ID indicates the identification tag for a write transaction.
axi_s0_awaddr_i	IN	32	The AXI4 write address indicates the address of the first transfer in a write transaction.
axi_s0_awlen_i	IN	8	The AXI4 write address length indicates the exact number of data transfers in a write transaction.
axi_s0_awsize_i	IN	3	The AXI4 write address size indicates the number of bytes in each data transfer in a write transaction.
axi_s0_awburst_i	IN	2	The AXI4 write address burst indicates how the address changes between each transfer in a write transaction.
axi_s0_awlock_i	IN	1	AXI4 write address lock This provides information about the atomic characteristics of a write transaction.
axi_s0_awcache_i	IN	4	The AXI4 write address cache indicates how a write transaction is required to progress through a system. This signal is unused.
axi_s0_awprot_i	IN	3	The AXI4 write address protect indicates the protection attributes of a write transaction such as, privilege, security level, and access type. This signal is unused.
axi_s0_awvalid_i	IN	1	The AXI4 write address valid indicates that the write address channel signals are valid.
axi_s0_awready_o	OUT	1	The AXI4 write address ready indicates that a transfer on the write address channel can be accepted.
axi_s0_wid_i	IN	AXI4 ID Width	The AXI4 write data ID indicates the ID tag of the write data transfer.
axi_s0_wdata_i	IN	Data Bus Width	AXI4 write data.
axi_s0_wstrb_i	IN	Data Bus Width / 8	The AXI4 write data strobe indicates which byte lanes hold valid data.
axi_s0_wlast_i	IN	1	The AXI4 write data last indicates whether this is the last data transfer in a write transaction.
axi_s0_wvalid_i	IN	1	The AXI4 write data valid indicates that the write data channel signals are valid.
axi_s0_wready_o	OUT	1	The AXI4 write data ready indicates that a transfer on the write data channel can be accepted.
axi_s0_bid_o	OUT	AXI4 ID Width	The AXI4 write response ID indicates the identification tag for a write response.

Port	Type	Width	Description
axi_s0_bresp_o	OUT	2	The AXI4 write response indicates the status of a write transaction. EXOKAY and DECERR error responses are not supported.
axi_s0_bvalid_o	OUT	1	The AXI4 write response valid indicates that the write response channel signals are valid.
axi_s0_bready_i	IN	1	The AXI4 write response ready indicates that a transfer on the write response channel can be accepted.
axi_s0_arid_i	IN	AXI4 ID Width	The AXI4 read address ID indicates the identification tag for a read transaction.
axi_s0_araddr_i	IN	32	The AXI4 read address indicates the address of the first transfer in a read transaction.
axi_s0_arlen_i	IN	8	The AXI4 read address length indicates the exact number of data transfers in a read transaction.
axi_s0_arsize_i	IN	3	The AXI4 read address size indicates the number of bytes in each data transfer in a read transaction.
axi_s0_arburst_i	IN	2	The AXI4 read address burst indicates how the address changes between each transfer in a read transaction.
axi_s0_arlock_i	IN	1	AXI4 read address lock This provides information about the atomic characteristics of a read transaction.
axi_s0_arcache_i	IN	4	The AXI4 read address cache indicates how a read transaction is required to progress through a system. This signal is unused.
axi_s0_arprot_i	IN	3	The AXI4 read address protect indicates the protection attributes of a read transaction such as, privilege, security level, and access type. This signal is unused.
axi_s0_arvalid_i	IN	1	The AXI4 read address valid indicates that the read address channel signals are valid.
axi_s0_arready_o	OUT	1	The AXI4 read address ready indicates that a transfer on the read address channel can be accepted.
axi_s0_rid_o	OUT	AXI4 ID Width	The AXI4 read data ID indicates the ID tag of the read data transfer.
axi_s0_rdata_o	OUT	Data Bus Width	AXI4 read data.
axi_s0_rresp_o	OUT	2	The AXI4 read data response indicates the status of a read transfer. EXOKAY and DECERR error responses are not supported.
axi_s0_rlast_o	OUT	1	The AXI4 read data last indicates whether this is the last data transfer in a read transaction.
axi_s0_rvalid_o	OUT	1	The AXI4 read data valid indicates that the read data channel signals are valid.
axi_s0_rready_i	IN	1	The AXI4 read data ready indicates that the receiver is ready to accept read data.
AXI Subordinate Interface Port1			
axi_s1_awid_i	IN	AXI4 ID Width	The AXI4 write address ID indicates the identification tag for a write transaction.
axi_s1_awaddr_i	IN	32	The AXI4 write address indicates the address of the first transfer in a write transaction.
axi_s1_awlen_i	IN	8	The AXI4 write address length indicates the exact number of data transfers in a write transaction.
axi_s1_awsize_i	IN	3	The AXI4 write address size indicates the number of bytes in each data transfer in a write transaction. If the address size is greater than the data bus width, it is set to the data bus width.
axi_s1_awburst_i	IN	2	The AXI4 write address burst indicates how the address changes between each transfer in a write transaction.
axi_s1_awlock_i	IN	1	AXI4 write address lock This provides information about the atomic characteristics of a write transaction.
axi_s1_awcache_i	IN	4	The AXI4 write address cache indicates how a write transaction is required to progress through a system. This signal is unused.
axi_s1_awprot_i	IN	3	The AXI4 write address protect indicates the protection attributes of a write transaction such as, privilege, security level, and access type. This signal is unused.

Port	Type	Width	Description
axi_s1_awvalid_i	IN	1	The AXI4 write address valid indicates that the write address channel signals are valid.
axi_s1_awready_o	OUT	1	The AXI4 write address ready indicates that a transfer on the write address channel can be accepted.
axi_s1_wid_i	IN	AXI4 ID Width	The AXI4 write data ID indicates the ID tag of the write data transfer.
axi_s1_wdata_i	IN	Data Bus Width	AXI4 write data.
axi_s1_wstrb_i	IN	Data Bus Width / 8	The AXI4 write data strobe indicates which byte lanes hold valid data.
axi_s1_wlast_i	IN	1	The AXI4 write data last indicates whether this is the last data transfer in a write transaction.
axi_s1_wvalid_i	IN	1	The AXI4 write address valid indicates that the sender has valid write data available for transfer.
axi_s1_wready_o	OUT	1	The AXI4 write data ready indicates that a transfer on the write data channel can be accepted.
axi_s1_bid_o	OUT	AXI4 ID Width	The AXI4 write response ID indicates the identification tag for a write response.
axi_s1_bresp_o	OUT	2	The AXI4 write response indicates the status of a write transaction. EXOKAY and DECERR error responses are not supported.
axi_s1_bvalid_o	OUT	1	The AXI4 write response valid indicates that the write response channel signals are valid.
axi_s1_bready_i	IN	1	The AXI4 write response ready indicates that a transfer on the write response channel can be accepted.
axi_s1_arid_i	IN	AXI4 ID Width	The AXI4 read address ID indicates the identification tag for a read transaction.
axi_s1_araddr_i	IN	32	The AXI4 read address indicates the address of the first transfer in a read transaction.
axi_s1_arlen_i	IN	8	The AXI4 read address length indicates the exact number of data transfers in a read transaction.
axi_s1_arsize_i	IN	3	The AXI4 read address size indicates the number of bytes in each data transfer in a read transaction.
axi_s1_arburst_i	IN	2	The AXI4 read address burst indicates how the address changes between each transfer in a read transaction.
axi_s1_arlock_i	IN	1	AXI4 read address lock This provides information about the atomic characteristics of a read transaction.
axi_s1_arcache_i	IN	4	The AXI4 read address cache indicates how a read transaction is required to progress through a system. This signal is unused.
axi_s1_arprot_i	IN	3	The AXI4 read address protect indicates the protection attributes of a read transaction such as, privilege, security level, and access type. This signal is unused.
axi_s1_arvalid_i	IN	1	The AXI4 read address valid indicates that the read address channel signals are valid.
axi_s1_arready_o	OUT	1	The AXI4 read address ready indicates that a transfer on the read address channel can be accepted.
axi_s1_rid_o	OUT	AXI4 ID Width	The AXI4 read data ID indicates the ID tag of the read data transfer.
axi_s1_rdata_o	OUT	Data Bus Width	AXI4 read data.
axi_s1_rresp_o	OUT	2	The AXI4 read data response indicates the status of a read transfer. EXOKAY and DECERR error responses are not supported.
axi_s1_rlast_o	OUT	1	The AXI4 read data last indicates whether this is the last data transfer in a read transaction.
axi_s1_rvalid_o	OUT	1	The AXI4 read data valid indicates that the read data channel signals are valid.

Port	Type	Width	Description
axi_s1_rready_i	IN	1	The AXI4 read data ready indicates that a transfer on the read data channel can be accepted.

Table 4.5. FIFO Streamer Ports

Port	Type	Width	Description
fifo_clk_i	IN	1	User Input Clock
fifo_wr_en_i	IN	1	Write Enable bit
fifo_wr_data_i	IN	8	Write data
fifo_interface_en_i	IN	1	Enable the FIFO interface and assert to start the transaction in the FIFO stream.
fifo_address_rstn_i	IN	1	The active-low reset clears all data inside the FIFO and resets the FIFO streamer registers.
fifo_full_o	OUT	1	It indicates that the FIFO is full.

Table 4.6. AXI4 Streamer Ports

Port	Type	Width	Description
axis_rx_aclk_i	IN	1	User clock input.
axi_rx_tvalid_i	IN	1	It indicates that the transaction values are valid.
axi_rx_tdata_i	IN	DATA WIDTH	User input data.
axi_rx_tstrb_i	IN	DATA WIDTH/8	Byte write strobe.
axi_rx_tlast_i	IN	1	It indicates the end of data write transaction.
axi_rx_tready_o	OUT	1	It indicates that the AXI4 streamer is ready to receive data.

Table 4.7. ECC Ports

Port	Type	Width	Description
ecc_ded_s0_o	OUT	1	It indicates that one error is detected in port S0.
ecc_sec_s0_o	OUT	1	It indicates that two errors are detected in port S0.
ecc_ded_s1_o	OUT	1	It indicates that one error is detected in port S1.
ecc_sec_s1_o	OUT	1	It indicates that two errors are detected in port S1.

5. Register Description

The System Memory IP Core does not use dedicated readable or writable memory registers. Instead, it uses hard IP memory to store data. The IP uses the lower 19-bit (512 KB) address for address decoding, while all upper bits are expected to be the base address, which does not need to be decoded in the System Memory IP.

6. Example Design

The System Memory IP example design allows you to compile, simulate, and test the System Memory IP on the following Lattice evaluation boards:

- MachXO5-NX Development Board

6.1. Example Design Supported Configuration

Note: In the table below, ✓ refers to a checked option in the System Memory IP example design.

Table 6.1. System Memory IP Configuration Supported by the Example Design

Attribute	Value
General Attribute	
Interface	AXI4
Memory Address Depth	57,344
Data Bus Width(bits)	64
Memory Type	LRAM
Port Count [1 -2]	2
ECC Enable	—
Enable Arbiter	✓
AXI4 ID Width	6
Enable Data Streamer	False
Data Streamer Interface	—
Data Streamer Write Start Address	0 – (ADDR_DEPTH-(Data Bus Width/8))
Data Streamer Clock Bypass	False
Initialize Memory	False
Initialization File Format	Hex
Initialization File	—
Port S0 and Port S1 attributes are set to default.	

6.2. Overview of the Example Design and Features

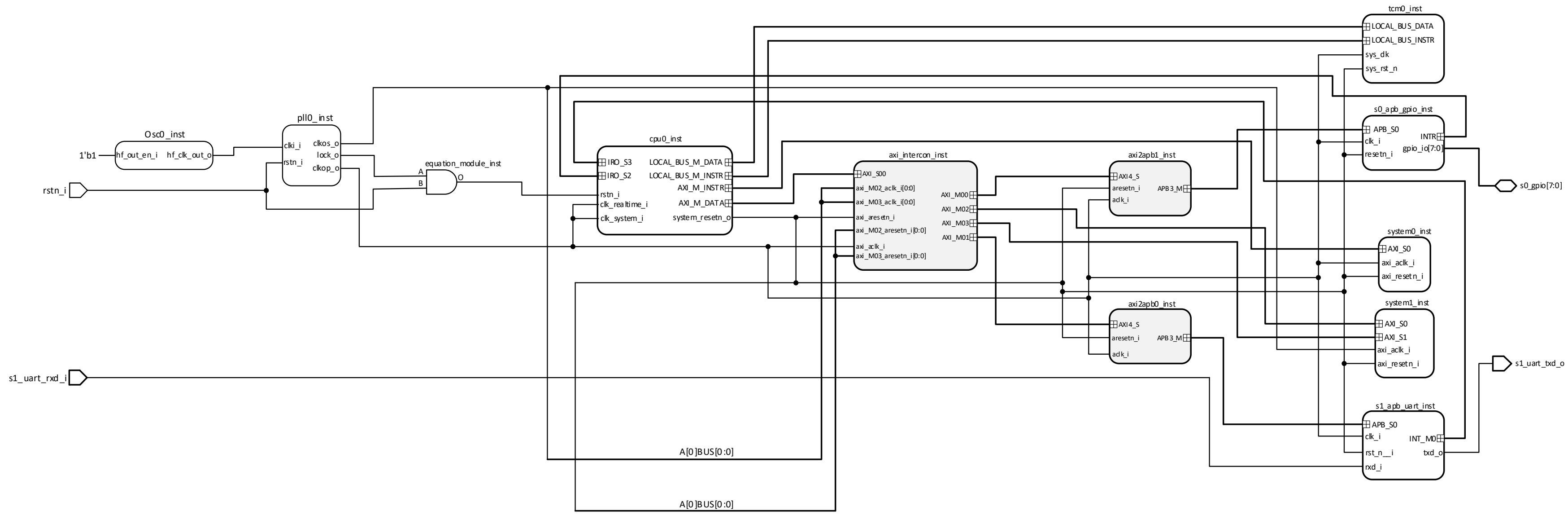

Figure 6.1. System Memory IP in Propel SoC Project

Key features of the example design include:

- Memory Access
  - The System Memory IP supports the use of available hard IP memory for data storage in the architecture.
  - Depending on the device, the System Memory IP uses distributed RAM, EBR, and LRAM.

6.3. Example Design Components

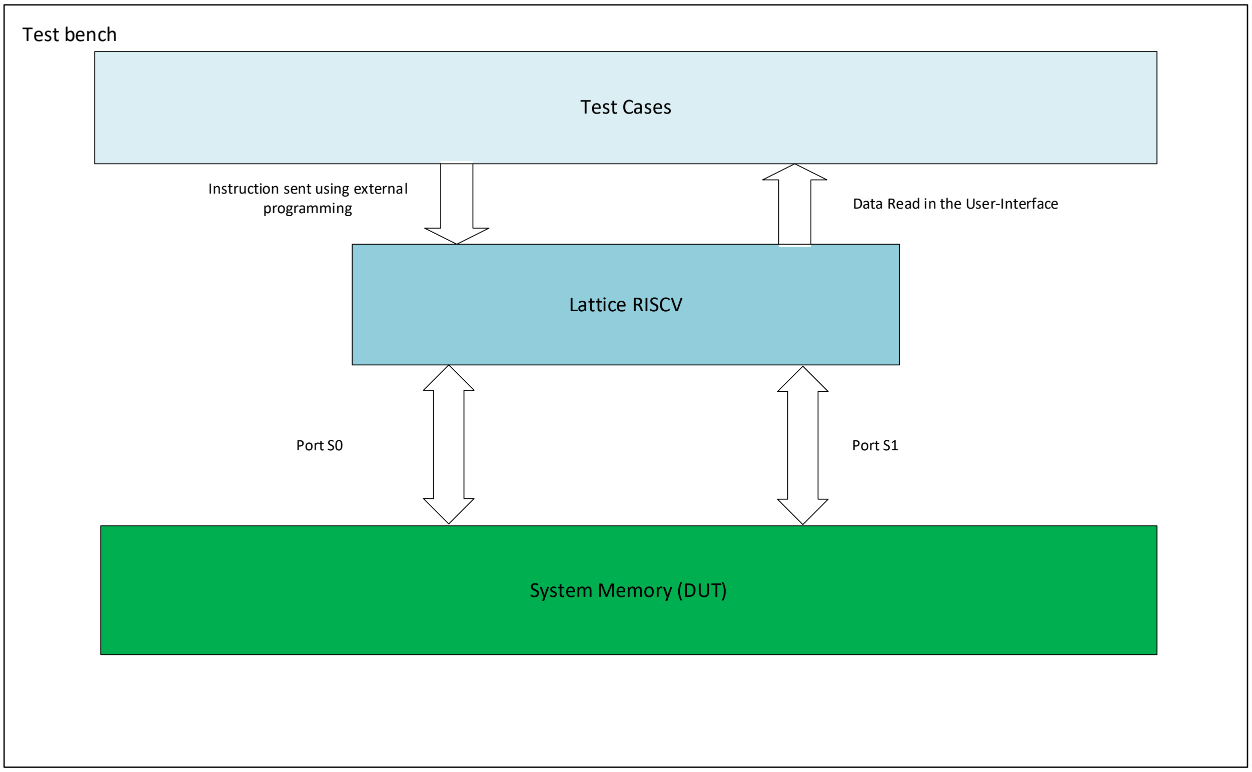

Figure 6.2. System Memory Example Design Block Diagram

The System Memory IP example design includes the following blocks:

- Test Cases
- Lattice RISC-V microcontroller
- System Memory IP

6.4. Test Cases

- The test cases are written in C code.
- These test cases are transmitted to the device via JTAG. The System Memory IP instantiation is used for external programming.
- Compare the read data with the expected data.

6.5. Lattice RISC-V Microcontroller

- The microcontroller receives the test cases as instructions and initiates read and write transactions to the System Memory IP.
- The AXI Interconnect IP is used in the data port of the microcontroller.

6.6. System Memory IP

- The design is under test.
- External programming is not included in test scope.
- Hardware validation includes read and write access to both ports of the System Memory DUT instantiation.

6.7. Simulation the Example Design

Refer to the Lattice Propel SDK User Guide in the Lattice Propel Design Environment web page for more details on the Lattice Propel design environment.

1. Launch the Lattice Propel SDK and set your workspace directory.
2. In the Lattice Propel SDK, create a new Lattice SoC Design Project by clicking **File > New > Lattice SoC Design Project**.
3. The Create SoC Project window opens.
   - In the **Device Select** section, specify the correct details of the device or board that you are using. In Figure 6.4, the device is set to LFMXO5-100T-9BBG400C since the MachXO5-NX Evaluation Board is used in the hardware testing.
   - In the **Template Design** section, choose **RISC-V RX SoC Project**. Click **Finish**.

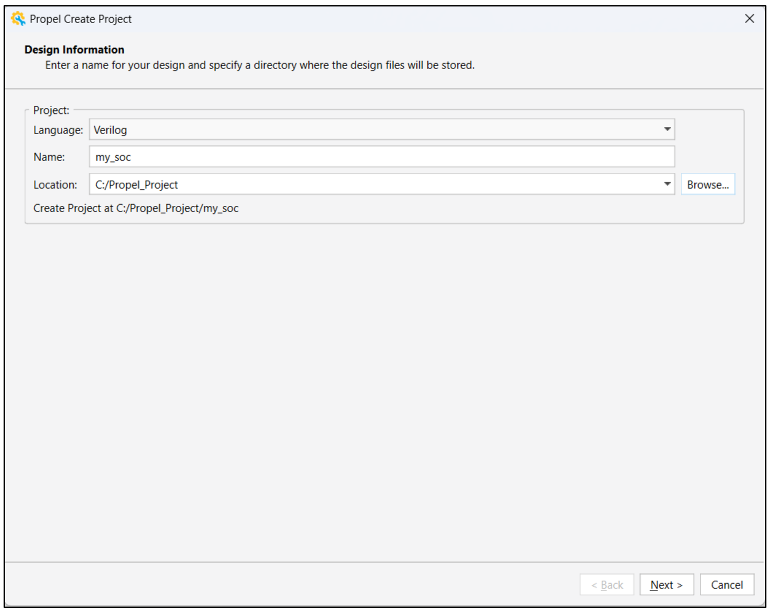

Figure 6.3. Create SoC Project

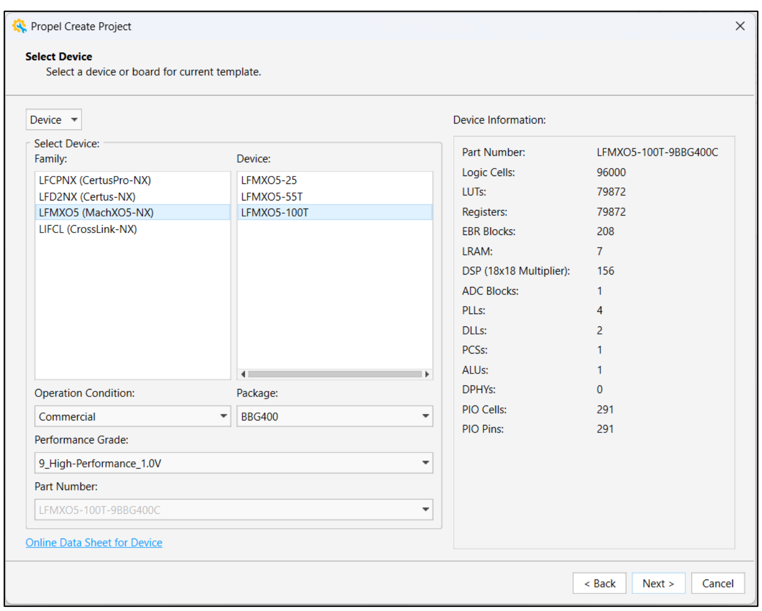

Figure 6.4. Create SoC Project

4. Run the Lattice Propel Builder by clicking the  icon or selecting **Lattice Tools** > **Open Design** in Lattice Propel Builder. The Propel Builder software opens and loads the design template.
5. In the **IP Catalog** tab, instantiate the System Memory IP. Refer to the Generating and instantiating the IP section for more details.

   See the Example Design Supported Configuration section for the corresponding parameter settings.

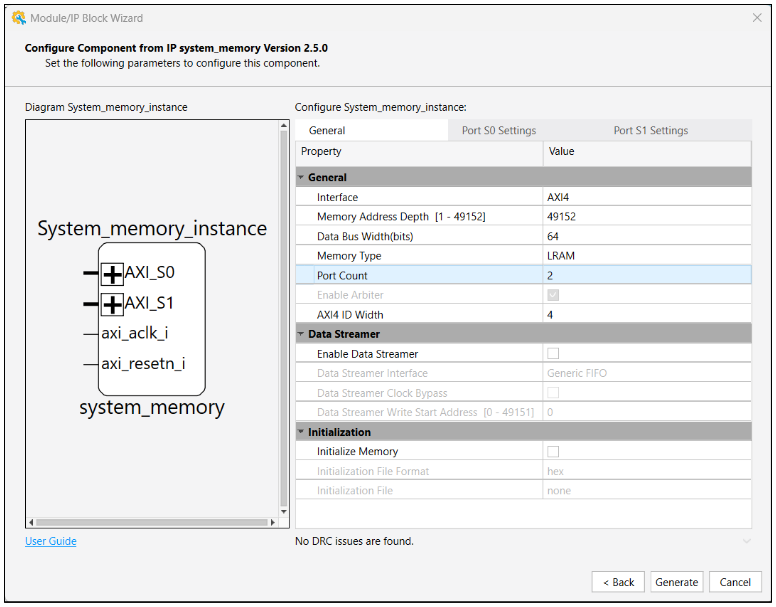

Figure 6.5. Instantiating System Memory IP Module

6. After generating the IP, the **Define Instance** window opens. Modify the instance name if needed, then, click **OK.**

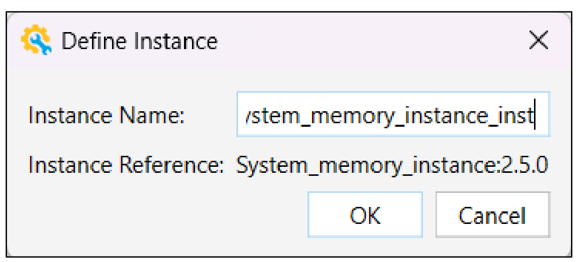

Figure 6.6. Defining Instances

7. Connect the instantiated IPs to the system. Refer to Figure 6.1 for the connections used in this IP. Update other components of the system for clock and reset sources, interrupt, and bus interface.
8. Click the  icon or select **Design** > **Run Radiant** to launch the Lattice Radiant Software.
9. Update your constraints file accordingly and generate the programming file.
10. In the Lattice Propel SDK, build your SoC project to generate the system environment needed for the embedded C/C++ project. Select your SoC project, then click **Project** > **Build Project**.
11. Check the build result from the **Console** view.
12. Generate a new Lattice C/C++ project by clicking **File** > **New** > **Lattice C/C++ Project**. Update your **Project name**, click **Next**, and then click **Finish**.
13. Select your C/C++ project, then select **Project** > **Build**.
14. Check the build result from the **Console** view.

15. This environment is now ready for running your tests on the device. Refer to the MachXO5-NX Development Board User Guide (FPGA-EB-02052) for a step-by-step guide.

7. Designing with the IP

This section explains how to generate the IP Core using the Lattice Radiant software and run simulation and synthesis. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide in the Lattice Radiant Software web page.

Note: The screenshots provided are for reference only. Details may vary depending on the version of the IP or software being used. If there have been no significant changes to the GUI, a screenshot may reflect an earlier version of the IP.

7.1. Generating and instantiating the IP

You can use the Lattice Radiant software to generate IP modules and integrate them into the device architecture. The steps below describe how to generate the System Memory IP in the Lattice Radiant software.

To generate the System Memory IP:

1. Create a new project in Lattice Radiant software or open an existing one.
2. In the **IP Catalog** tab, double-click **System Memory** under **IP, Processors_Controllers_and_Peripherals** category. The **Module/IP Block Wizard** opens as shown in Figure 7.1 . Enter values in the **Component name** and **Create in** fields, then click **Next**.

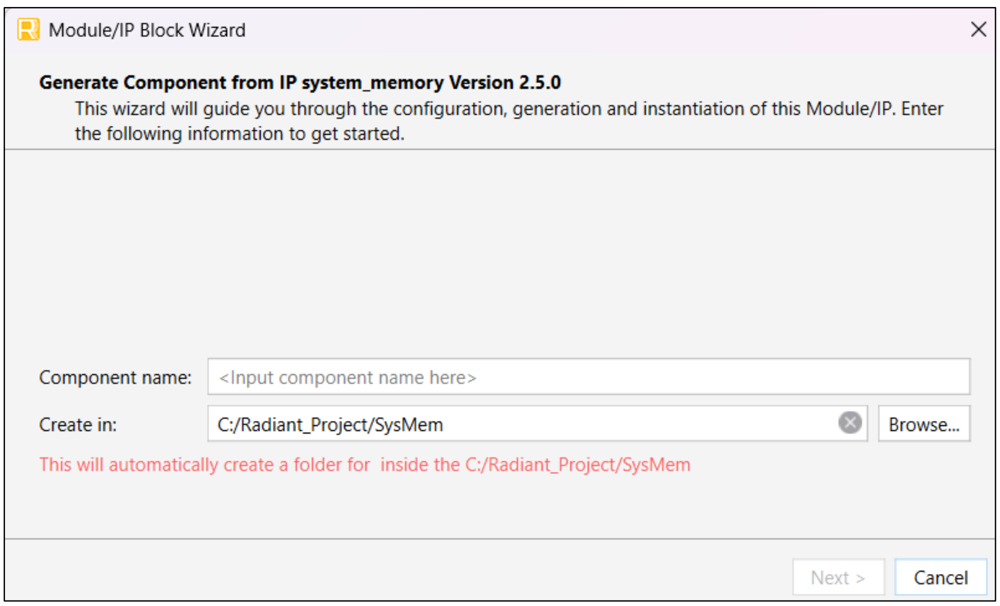

Figure 7.1. Module/IP Block Wizard

3. In the next **Module/IP Block Wizard** window, customize the selected System Memory IP using the drop-down lists and checkboxes. Figure 7.2 shows an example configuration of the System Memory IP. Refer to the IP Parameter Description section for details on the configuration options.

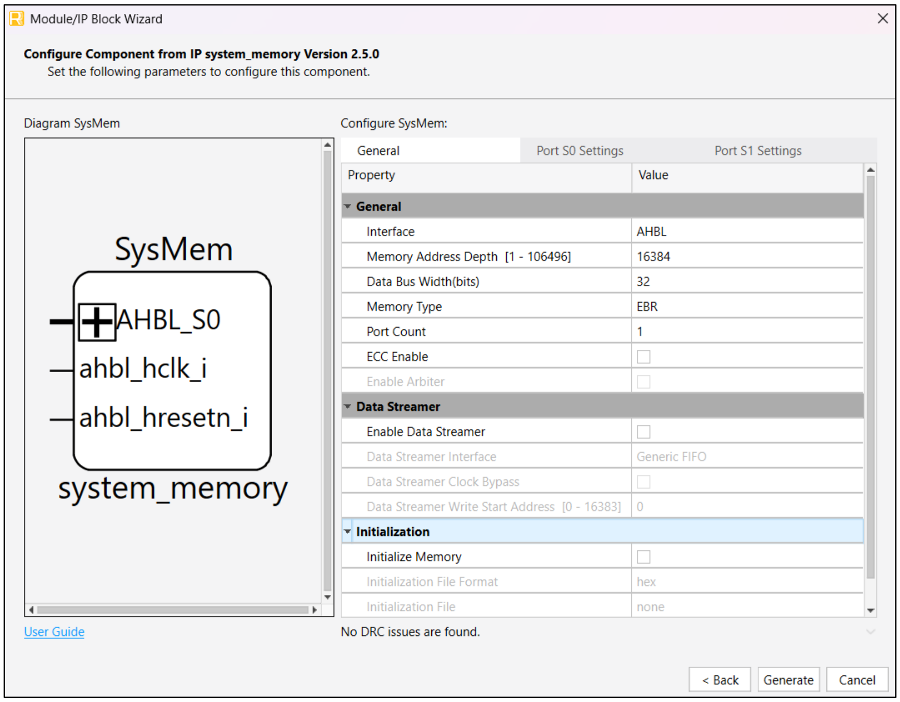

Figure 7.2. IP Configuration

4. Click **Generate**. The **Check Generated Result** dialog box opens, displaying the design block messages and results, as shown in Figure 7.3.

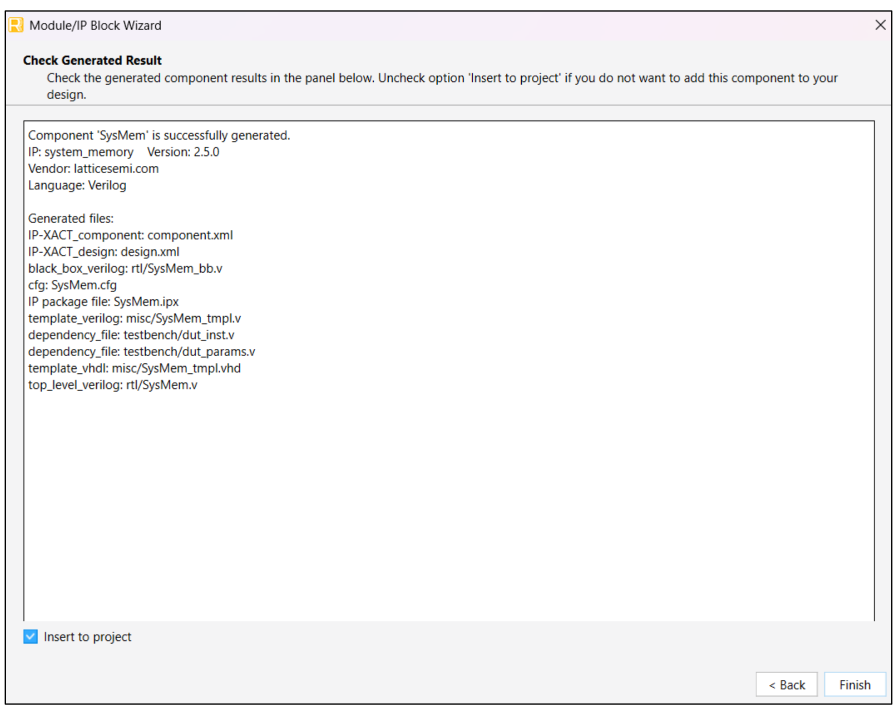

Figure 7.3. Check Generated Result

5. Click **Finish**. The generated files are placed under the directory paths specified in the **Create in** and the **Component name** fields, as shown in Figure 7.1.

7.1.1. Generated Files and File Structure

The generated System Memory module package includes the closed-box (<Component name>_bb.v) and instance templates (<Component name>_tmpl.v/vhd) for instantiating the core in a top-level design. An example RTL top-level reference source file (<Component name>.v) is also provided, which you can use as an instantiation template for the module or as the starting template for the top-level design. Table 7.1 lists the generated files.

Table 7.1. Generated File List

Attribute	Description
<Component name>.ipx	This file contains information about the files associated with the generated IP.
<Component name>.cfg	This file contains the parameter values used for IP configuration.
component.xml	This contains the ipxact component information of the IP.
design.xml	This document specifies the configuration parameters of the IP in the IP-XACT 2014 format.
rtl/<Component name>.v	This file provides an example RTL top file that instantiates the module.
rtl/<Component name>_bb.v	This file provides the synthesis closed box.
misc/<Component name>_tmpl.v misc /<Component name>_tmpl.vhd	These files provide instance templates for the module.

7.2. Design Implementation

Completing your design involves specifying 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 constraint .pdc source files for storing logical timing/physical constraints. Constraints added using the device constraint editor are saved to the active .pdc file which 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.

7.3. Timing Constraints

The timing constraints depend on the clock frequency. The relevant constraint files define the timing constraints for the IP. The example below shows the IP timing constraints generated for the System Memory IP.

```
create_clock -name {axi_aclk_i} -period 8 -waveform {0 4} [get_ports axi_aclk_i]
```

Figure 7.4. Timing Constraint File (.pdc) for the System Memory IP

7.4. Physical Constraints

The System Memory IP has no specific physical constraints.

7.5. Specifying the Strategy

The Lattice Radiant software provides two predefined strategies like Area and Timing. It also allows you to create customized strategies. Refer to the Strategies section of the Lattice Radiant Software User Guide for details on creating a new strategy.

7.6. Running Functional Simulation

Run the functional simulation after generating the IP.

1. Click the button on the **Toolbar** to initiate the **Simulation Wizard**, as shown in Figure 7.5.

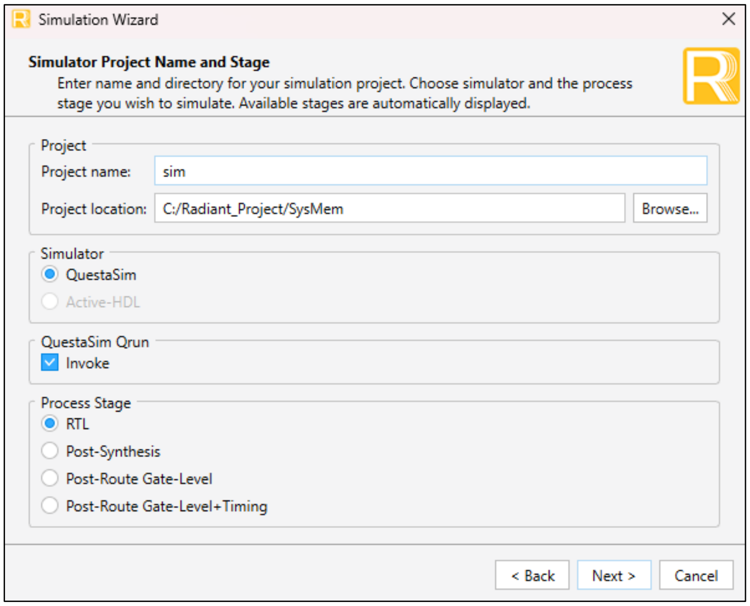

Figure 7.5. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window, as shown in Figure 7.6.

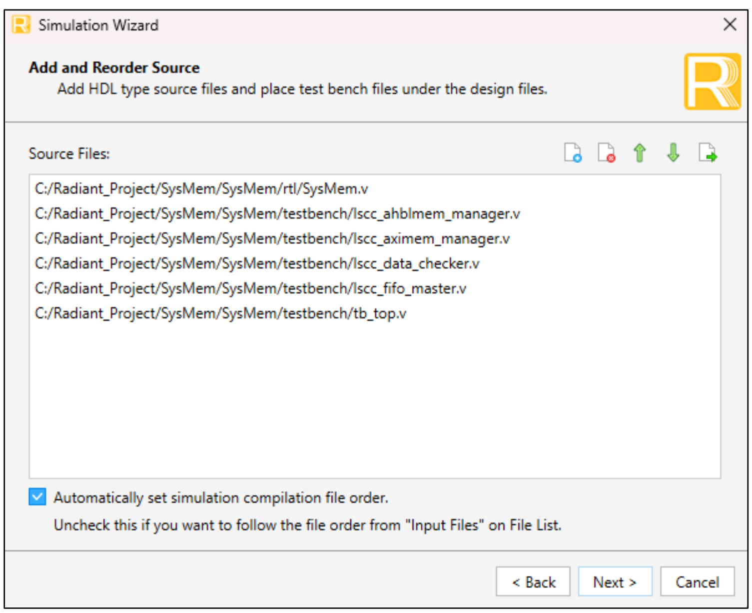

Figure 7.6. Add and Reorder Source

3. Click **Next** to open the **Summary** window.
4. Click **Finish** to run the simulation.

The waveform in Figure 7.7 illustrates an example simulation result.

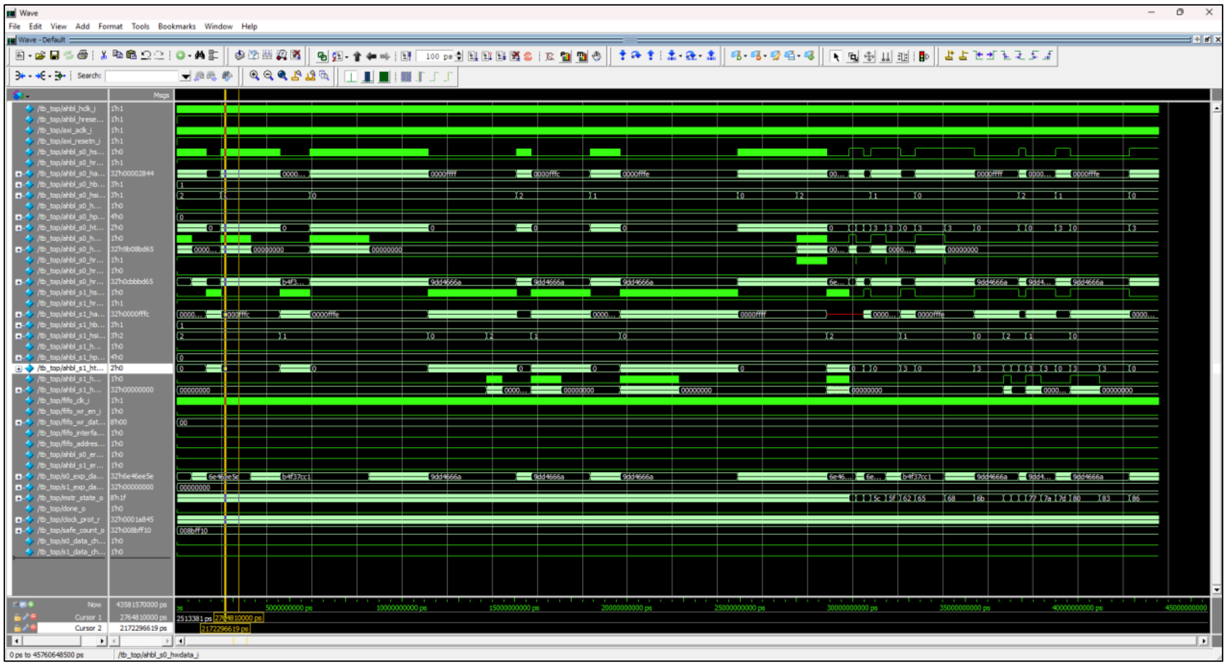

Figure 7.7. Simulation Waveform

7.6.1. Simulation Results

The simulation results show write and read transactions in the AHBL interface of the System Memory IP. The customer testbench passes when the write transaction data (expected data) matches the read data.

8. Debugging

This section lists possible issues and suggests troubleshooting steps to follow.

8.1. Debug Methods

8.1.1. Hardware Detection Failure

Follow the steps in the flow diagram below if the system does not detect the hardware.

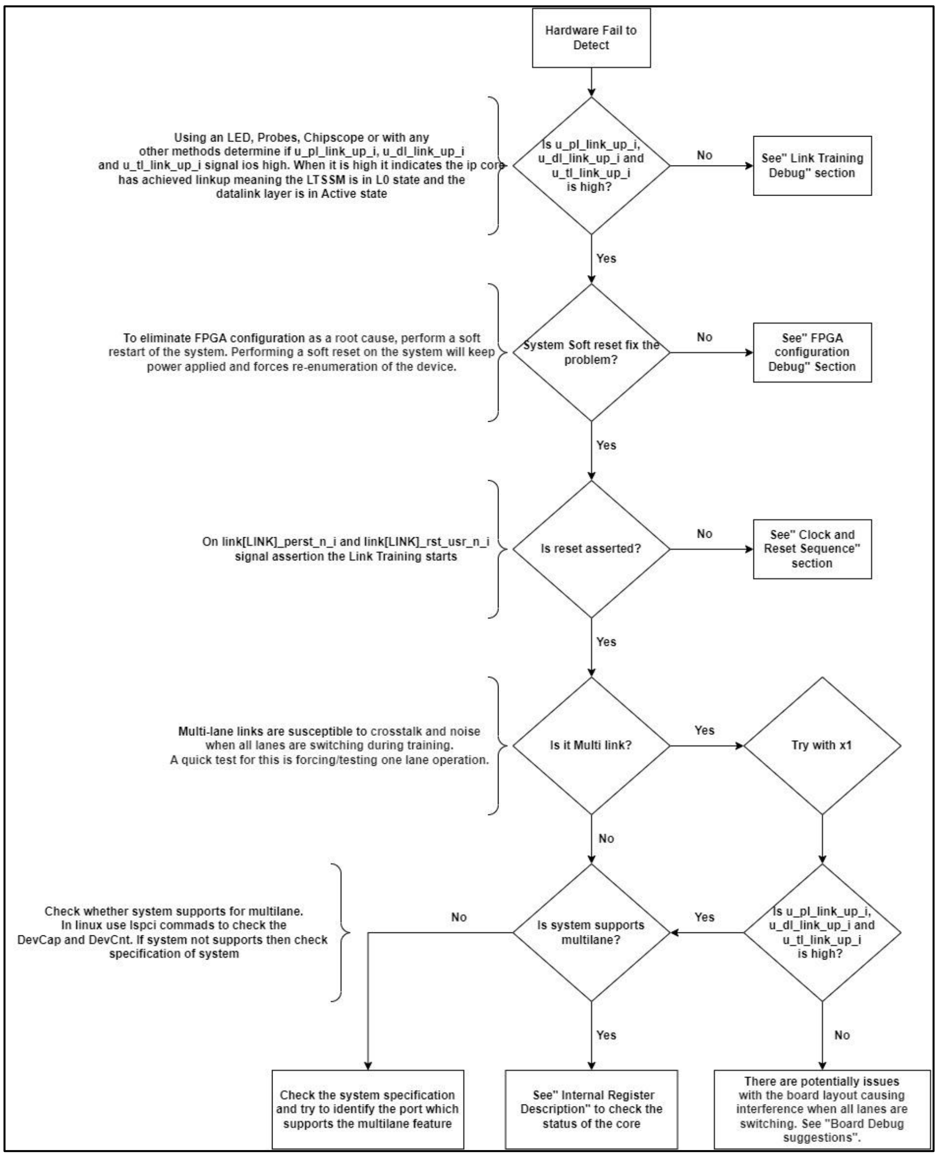

Figure 8.1. Hardware Detection Failure Debugging Flow

8.2. Debug Tools

You can use various tools to debug System Memory IP design issues.

9. Design Considerations

9.1. Design Considerations for External Programming

- Select the appropriate clocking architecture based on the System Memory IP configuration.
- Select the appropriate user interface. For ease of use, choose the same user interface as the microcontroller.
- Connect the microcontroller's instruction port to Subordinate/Port S0. The System Memory IP prioritizes port S0 when both ports access the same address simultaneously. For faster access, connect instruction ports to port S0.

9.2. Design Considerations for Memory Storage

- Select the appropriate clocking architecture based on the System Memory IP configuration.
- Select the appropriate user interface. For ease of use, choose the same user interface as the microcontroller.
- By default, both ports access all memory addresses. If your application requires these ports to access different memory addresses, fill in the appropriate parameters in the graphic user interface when instantiating the IP.

Appendix A. Resource Utilization

Table A.1 shows a sample resource utilization of the System Memory IP Core on LFCPNX-100-7CBG256C.

Table A.1. Resource Utilization

IP Configuration	Slices	LUTs	Registers	EBR	LRAM
AHB-Lite Data Interface, Default Address Depth, 32-bit Data Width, EBR Memory Type, 2 Port Count	486/79,872	999/79,872	486/80,349	64/208	0/7
AXI4 Data Interface, Default Address Depth, 64-bit Data Width, EBR Memory Type, 1 Port Count	557/79,872	1306/79,872	573/80,349	64/208	0/7
AHB-Lite Data Interface, Default Address Depth, 32-bit Data Width, EBR Memory Type, 2 Port Count, Data Streaming True, FIFO Data Streamer	445/79,872	1081/79,872	445/80,349	64/208	0/7
AXI4 Data Interface, Default Address Depth, 32-bit Data Width, EBR Memory Type, 2 Port Count, Data Streaming True, AXI4 Streamer	990/79,872	2254/79,872	1023/80,349	64/208	0/7
AHB-Lite Data Interface, Default Address Depth, 32-bit Data Width, LRAM Memory Type, 2 Port Count	418/79,872	1011/79,872	418/80,349	0/208	1/7
AXI4 Data Interface, Default Address Depth, 64-bit Data Width, LRAM Memory Type, 1 Port Count	558/79,872	1292/79,872	574/80,349	0/208	1/7
AHB-Lite Data Interface, Default Address Depth, 32-bit Data Width, LRAM Memory Type, 2 Port Count, Data Streaming True, FIFO Data Streamer	443/79,872	1091/79,872	443/80,349	0/208	1/7
AXI4 Data Interface, Default Address Depth, 32-bit Data Width, LRAM Memory Type, 2 Port Count, Data Streaming True, AXI4 Streamer	986/79,872	2256/79,872	1019/80,349	0/208	1/7
AXI4 Data Interface, Default Address Depth, 32-bit Data Width, EBR Memory Type, 2 Port Count, Atomic Enable S0, Atomic Enable S1	1038/79,872	2689/79,872	1072/80,349	64/208	0/7

References

- Arm web page for the AMBA 3 AHB-Lite Protocol Specification, AMBA AXI Protocol Specification, and AMBA AXI-Stream Protocol Specification
- System Memory Module IP Release Notes (FPGA-RN-02065)
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- iCE40 UltraPlus web page
- ECP5 / ECP5-5G web page
- CrossLink-NX web page
- Certus-NX web page
- Certus-N2 web page
- CertusPro-NX web page
- Mach-NX web page
- MachXO2 web page
- MachXO3 web page
- MachXO3D web page
- MachXO4 web page
- MachXO5-NX web page
- Avant-E web page
- Avant-G web page
- Avant-X web page
- System Memory Module IP Core web page
- Lattice Radiant Software web page
- Lattice Propel Design Environment web page
- Lattice Diamond Software 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

Note: In some instances, the IP may be updated without changes to the user guide. The user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.

Revision 2.4, IP v2.5.0, December 2025

Section	Change Summary
All	• Added a note on IP version in Quick Facts and *Revision History* sections. • Performed minor formatting and editorial edits.
Abbreviations in This Document	Updated list of abbreviations.

Introduction	• Updated Table 1.1. Summary of the System Memory Module IP as follows:
	  • Added MachXO4 devices.
	  • Updated IP version.
	• Updated Table 1.2. System Memory Module IP Support Readiness as follows:
	  • Updated the hardware validation status for Lattice Avant, CertusPro-NX, and Mach-NX devices.
	  • Updated data rate for all devices.
	• Added the AXI4 atomic access feature in the Features section.
	• Updated the Licensing and Ordering Information section.

Functional Description	• Added description on LFMXO4 devices in AHB-Lite and AXI4 sections.
	• Updated the Memory Implementation section as follows:
	  • Renamed and updated Table 2.2. System Core Memory Type – AHB-Lite.
	  • Added Table 2.3. System Core Memory Type – AXI.
	  • Updated memory implementation for LRAM – AXI and EBR – AXI in Table 2.4. System Core Memory Implementation.
	  • Updated notes for LRAM and EBR memory type in Table 2.6. ECC Implementation per Memory Block.
	  • Removed the note on FIFO interface for distributed RAM devices in Table 2.7. Allowable Combination of Features for System Memory when INTERFACE = AHBL.
	• Added the *Multiple AXI4 Exclusive Read* error in Table 2.8. System Memory Error.
	• Updated the Arbitration section and added the following subsections:
	  • Arbitration in the AXI4 Interface
	  • Arbitration in the AHBL Interface
	• Added the AXI4 Atomic Access section.

IP Parameter Description	• Added a note to the *ECC Enable* attribute in Table 3.1. General Attributes. • Added the *Enable Atomic Access S0* attribute in Table 3.2. Port S0 Settings. • Added the *Enable Atomic Access S1* attribute in Table 3.3. Port S1 Settings.
Signal Description	Updated the description for axi_s0_awlock_i, axi_s0_arlock_i, axi_s1_awlock_i, and axi_s1_arlock_i in Table 4.4. AXI Ports.

Example Design	• Updated the reference to the Lattice Propel SDK User Guide in the Simulation the Example Design section.
	• Updated the following figures:
	  • Figure 6.5. Instantiating System Memory IP Module
	  • Figure 6.6. Defining Instances

Designing with the IP	• Updated the Designing with the IP section as follows:
	  • Updated the reference to the Lattice Radiant Software User Guide.
	  • Added a note on IP version in GUI.
	• Updated the following figures:
	  • Figure 7.1. Module/IP Block Wizard
	  • Figure 7.2. IP Configuration
	  • Figure 7.3. Check Generated Result

Appendix A. Resource Utilization	Updated Table A.1. Resource Utilization.

Section	Change Summary
References	Updated references.

Revision 2.3, IP v2.4.0, June 2025

Section	Change Summary
All	• Updated the IP version from *2.3.0* to *2.4.0*. • Minor editorial fixes.

Introduction	• Reworked section contents.
	• Added the following subsection:
	  • Overview of the IP
	  • Quick Facts
	  • IP Support Summary
	  • Licensing and Ordering Information
	  • Hardware Support
	  • Minimum Device Requirements

Functional Description	• Reworked section contents.
	• Added the following subsection:
	  • IP Architecture Overview
	  • Clocking
	  • Reset
	  • User Interfaces
	  • Memory Implementation
	  • Arbitration
	• Removed the following subsection:
	  • Block Diagram – replaced by IP Architecture Overview
	  • Functional Overview – replaced by User Interface
	  • Attribute Summary
	  • Signal Description

IP Parameter Description	Added this section.
Signal Description	Added this section.
Register Description	Added this section.
Example Design	Added this section.
Designing with the IP	Added this section.
Debugging	Added this section.
Design Considerations	Added this section.
Appendix A. Resource Utilization	Reworked section contents.
References	Updated section contents.

Revision 2.2, IP v2.3.0, December 2024

Section	Change Summary
All	Minor editorial fixes.
Abbreviations in This Document	Replaced *Acronyms* with *Abbreviations*.
Introduction	Added Certus-N2 in Table 1.1. FPGA Software for IP Configuration, Generation, and Implementation.
Appendix A. Resource Utilization	• Updated the Clock Fmax, Registers, and LUTs values in Table A.1. Resource Utilization Using LFCPNX-100-9LFG672C. • Updated the Clock Fmax, Registers, and LUTs values in Table A.2. Resource Utilization Using LAV-AT-E70-2LFG1156C.
References	Added System Memory Module IP Release Notes (FPGA-RN-02065) in this section.

Revision 2.1, January 2024

Section	Change Summary
All	Updated the document title to *System Memory Module*.
Disclaimers	Updated this section.
Functional Description	In Table 2.6. System Memory Attribute Summary: • added the Enable Port S0 Read Pipeline attribute under Port S0 Settings; • added the Enable Port S1 Read Pipeline attribute under Port S1 Read Pipeline; • added the following table note for newly added attributes: Available only when Interface=AHBL, Memory Type=LRAM, and output register enabled.
Resource Utilization	• Updated the resource utilization of LFCPNX-100-9LFG672C in Table A.1. Resource Utilization Using LFCPNX-100-9LFG672C; • Updated the resource utilization of LAV-AT-E70-2LFG1156C in Table A.1. Resource Utilization Using LFCPNX-100-9LFG672C;
References	Newly added the link to Lattice Insights for Lattice Semiconductor training series and learning plans.

Revision 2.0, June 2023

Section	Change Summary
All	Minor adjustments in formatting across the document.
Inclusive Language	Added this section.

Introduction	• Updated Table 1.1. FPGA Software for IP Configuration, Generation, and Implementation to add support for Avant.
	• Updated the following in Features section:
	  • Added bullet point for AMBA AXI4 protocol.
	  • Updated single or dual port memory bullet point to add AXI4 interface.
	  • Updated 32-bit data bullet point to add 8 and 16-bit values.

Functional Description	• Updated overall diagram of Figure 2.1. Generic System Memory Block Diagram.
	• Updated AHB-L bus information in AHB-Lite Interface.
	• Added AXI4 Interface and AXI4-Stream Interface sections.
	• Updated the following in Table 2.1. System Core Memory Implementation:
	  • Added Avant and AXI4 Configuration Used columns.
	  • Updated values and changed column name from Configuration Used to *AHB-Lite Configuration Used*.
	  • Changed LIFCL, LFD2NX column name to *LIFCL, LFCPNX, LFD2NX*.
	  • Changed memory type from ram_dp to *EBR*.
	  • Added LRAM table note and removed Byte enable note.
	• Updated Table 2.2. Features Supported per Memory Block to change RISC-V device row to *Unaligned Read Access*, EBR value to *Yes*, and table note to *Unaligned Read Access cannot be used in conjunction with Byte-enable*.
	• Updated Table 2.3. ECC Implementation per Memory Block to change ram_dp memory type to EBR, add LFCPNX in ECC Implementation column, and add table notes for ECC function.
	• Updated Table 2.4. Allowable Combination of Features for System Memory when INTERFACE = AHBL to add *when INTERFACE = AHBL* in table name, add AHBL information in table note, and remove Unaligned read access information in table note.
	• Updated overall content of Table 2.6. System Memory Attribute Summary to add and change Attribute, Values, Default, and Description columns, as well as the table notes.
	• Updated the following in Table 2.7. System Memory Ports:
	  • Applied inclusive language.
	  • Added axi_aclk_i and axi_resetn_i row.
	  • Changed DATA_WIDTH to *Data Bus Width* across the table.

Section	Change Summary
	• Added AXI4 Subordinate Interface for Port 0 and AXI4 Subordinate Interface for Port 1 groups and values. • Added ECC Outputs for Port 0 and ECC Outputs for Port 1 groups and values. • Added fifo_full_o in FIFO Interface group. • Added AXI4 Stream Interface group and values. • Added table notes for Data Streamer and ECC Enable. • Changed System Memory Timing Information section name to Timing Information for AHB-Lite Interface. • Added Timing Information for AXI4 Interface and Timing Diagrams for AXI4 Interface sections.
Appendix A. Resource Utilization	Added this section.
References	Added reference links for AMBA AXI4, AMBA AXI, CertusPro-NX, and Avant and updated webpage listing structure.
Technical Support Assistance	Added reference to the Lattice Answer Database on the Lattice website.

Revision 1.3, April 2022

Section	Change Summary
Introduction	Updated Table 1.1. FPGA Software for IP Configuration, Generation, and Implementation to add support for CertusPro-NX.
Functional Description	• Updated FIFO Interface content to add information that this only supports LIFCL, LFCPNX, and LFD2NX devices. • Updated Memory Implementation content to add LFCPNX. • Updated ADDR_DEPTH value and description, and removed reference to previous table note 1 in Table 2.6. System Memory Attribute Summary.
References	Updated content to add web page reference for CertusPro-NX and corrected web page link for Certus-NX.

Revision 1.2, May 2021

Section	Change Summary
Introduction	Updated Table 1.1 to add MachXO2 and MachXO3 as supported FPGA family.
References	Updated content to add reference for MachXO2 and MachXO3.

Revision 1.1, November 2020

Section	Change Summary
All	Added CrossLink-NX, Certus-NX, and FIFO interface support across the document.
Introduction	• Added Table 1.1. • Updated content in Features to change memory support to 1 Mb.
Functional Description	• Added FIFO Interface and Memory Implementation section. • Updated Table 2.1, Table 2.2, Table 2.3, Table 2.4, Table 2.5, and Table 2.6. • Added Figure 2.9.
References	Updated content to remove reference links for Lattice Propel and Lattice Diamond user guide; and to add reference links for Mach-NX, CrossLink-NX and Certus-NX web page.

Revision 1.0, May 2020

Section	Change Summary
All	Initial release

www.latticesemi.com