MDIO Leader IP

IP Version: v1.3.0

User Guide

FPGA-IPUG-02223-1.3

December 2025

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

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
AHBL	AHB Lite
APB	Advanced Peripheral Bus
AXI	Advanced Extensible Interface
FPGA	Field Programmable Gate Array
IP	Intellectual Property
MAC	Media Access Control
MDC	Management Data Clock
MIIM	Media Independent Interface Management
MDIO	Management Data Input Output
PCS	Physical Coding Sublayer
RTL	Register Transfer Level
SMI	Structure of Management Information

1. Introduction

Management Data Input/Output (MDIO), or Media Independent Interface Management (MIIM) is a serial bus protocol defined for the IEEE 802.3 standard Ethernet series of Media Independent Interface (MII). MII connects media access control (MAC) devices to Ethernet physical layer (PHY) circuits. The SMI/MDIO protocol is a simple two-wire serial interface that connects the management unit to the managed PHY to control the PHY and capture the status of the PHY. The Management Data Input/Output (MDIO) component can be used to read and write the PHY control register. Each PHY can be monitored before operation and the connection status can be monitored during operation. These registers provide status and control information such as: link status, speed ability and selection, power down for low power consumption, duplex mode (full or half), auto-negotiation, fault signaling, and loopback. The main purpose of using MDIO protocol is to configure the PHY layer transceiver parameters. For example, the PHY devices can perform pre-emphasis or de-emphasis in the Physical Coding Sublayer (PCS), programming the control status registers in the PHY layer.

MDIO is a bidirectional shared bus structure that can provide a connection from the MAC (leader) up to 32 PHY (follower) devices. All data is synchronously transmitted with respect to the Management Data Clock (MDC), which is provided by the MAC and sent to all receiving devices. The data line is a tri-state shared bus that is MAC controlled for a write transaction or PHY controlled during a read transaction. The MDIO interface clock (MDC) supports frequency up to 2.5 MHz. The host processor, which is responsible for system configuration and monitoring, usually uses the MDIO host to perform individual access to various devices. MDIO was originally defined in Clause 22 of IEEE 802.3. To meet the growing needs of 10 Gigabit Ethernet devices, Clause 45 of the 802.3ae specification is introduced.

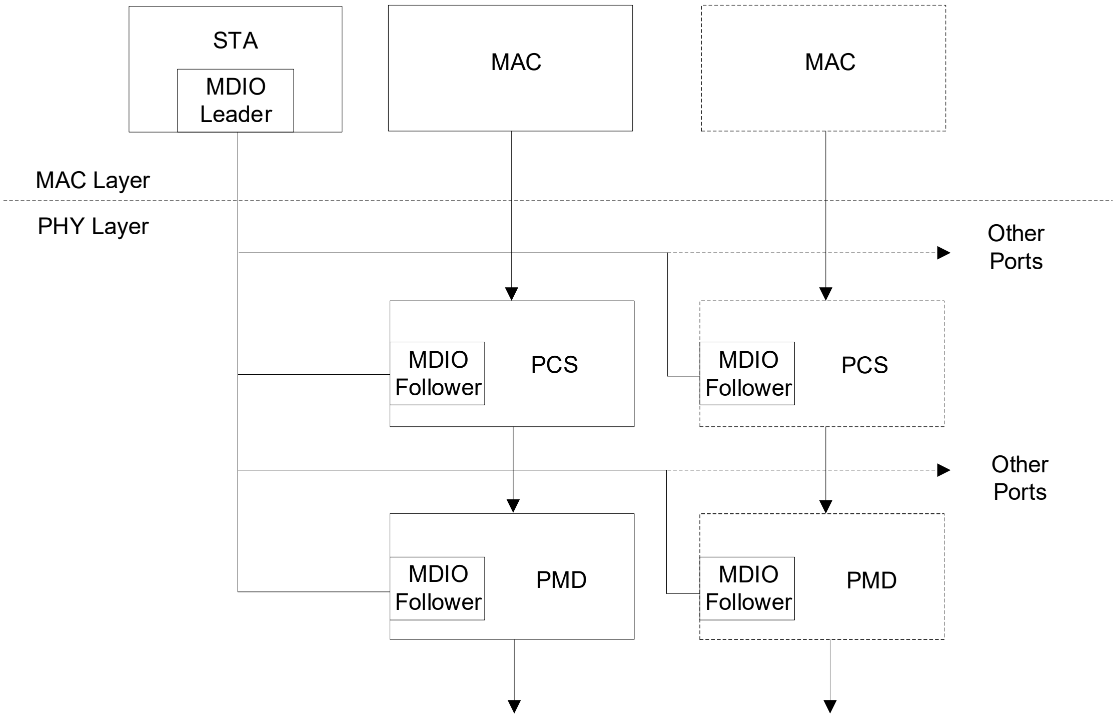


Figure 1.1. Generic Application Environment of MDIO Interface

1.1. Quick Facts

Table 1.1. Summary of the MDIO Leader IP

IP Requirements	Supported Devices	Lattice Avant™, CertusPro™-NX, Certus™-NX (LFD2NX-35, LFD2NX-65), Certus™-N2, and MachXO5™-NX (LFMXO5-35, LFMXO5-35T, LFMXO5-65, LFMXO5-65T).
	IP Changes[1]	For a list of changes to the IP, refer to the MDIO Leader IP Release Notes (FPGA-RN-02029).
Resource Utilization	Supported User Interface	Host: AHB-Lite, AXI-Lite, and APB.
	Resource Usage	Refer to Appendix A. Resource Utilization.
Design Tool Support	Lattice Implementation	IP core v1.3.0 – Lattice Radiant™ software 2025.2 or later.
	Synthesis	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.

1.2. Features

The key features of the MDIO Leader IP include:

- Implements the IEEE 802.3 Standard, Clause 22 and Clause 45 Leader interface.
- Three different standard interfaces for accessing the control and status signals of Leader: APB, AHB-L, and AXI-L. The interface selection is controlled using parameter.
- User control for selection between Clause 22 and Clause 45 protocols.
- Dynamic selection for Preamble pattern generation in MDIO frames.
- User control for MDC clock divider settings.

1.3. Licensing Information

The MDIO Leader IP is provided at no additional cost with the Lattice Radiant software.

1.4. Conventions

1.4.1. Nomenclature

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

1.4.2. Signal Names

Signal names that end with:

- _*n* are active low
- _*i* are input signals
- _*o* are output signals
- _*io* are bi-directional input/output signals

Signal names that begin with:

- *r*_ are registers
- *w*_ are wires
- *i*_ are input signals
- *o*_ are output signals
- *io*_ are bi-directional input/output signals

1.4.3. Attribute

The names of attributes in this document are formatted in title case and italicized (*Attribute Name*).

2. Functional Description

2.1. Overview

The MDIO Leader IP core is provided as an encrypted intellectual property (IP) core. Figure 2.1 shows the MDIO Leader IP core functional block diagram. The user interface is provided as a simple APB/AHB/AXI-L compliant interface configurable by the Host Interface Selection parameter. Table 2.1 provides the details of these I/O of the MDIO Leader IP.

The leader core generates the clock MDC using input clock *clk_i* from the top input interface. The register controlled CLKDIV setting (*CLK_CONTROL_REG*) can be used to define the frequency relationship between input *clk_i* and output MDC. The *MDC* clock frequency is half the frequency of *clk_i* divided by the CLKDIV settings value. For an example of 100 MHz of *clk_i* clock, CLKDIV setting of *20* gives the MDC clock of 2.5 MHz (100/20 = 5 and half of 5 = 2.5). The MDIO Leader IP core implements indirect addressing mechanism to access PHY registers across the MDIO interface. It implements four 32-bit wide registers that can be read and written from the interface selected by you using the Host Interface Selection parameter. The field details of these four registers are shown in Table 2.3 to Table 2.7. The registers are from byte addresses 0x00 to 0x0C of the REGIF Controller address space and can be written with different fields, which are used for building the MDIO frames. Refer to Operation Details and Programming Flow sections for detailed information on how frames are generated. The MDIO Leader IP core can be used to generate both Clause 22 and Clause 45 frame formats.

CLK_CONTROL_REG is used to control the clock divider settings for the clock generation from the input clock.
All registers support the four-byte Dword-based addressing. This means that the registers are accessed at the boundary of four bytes. For example, a request to access the 0x01 address will result in accessing the 0x0 address.

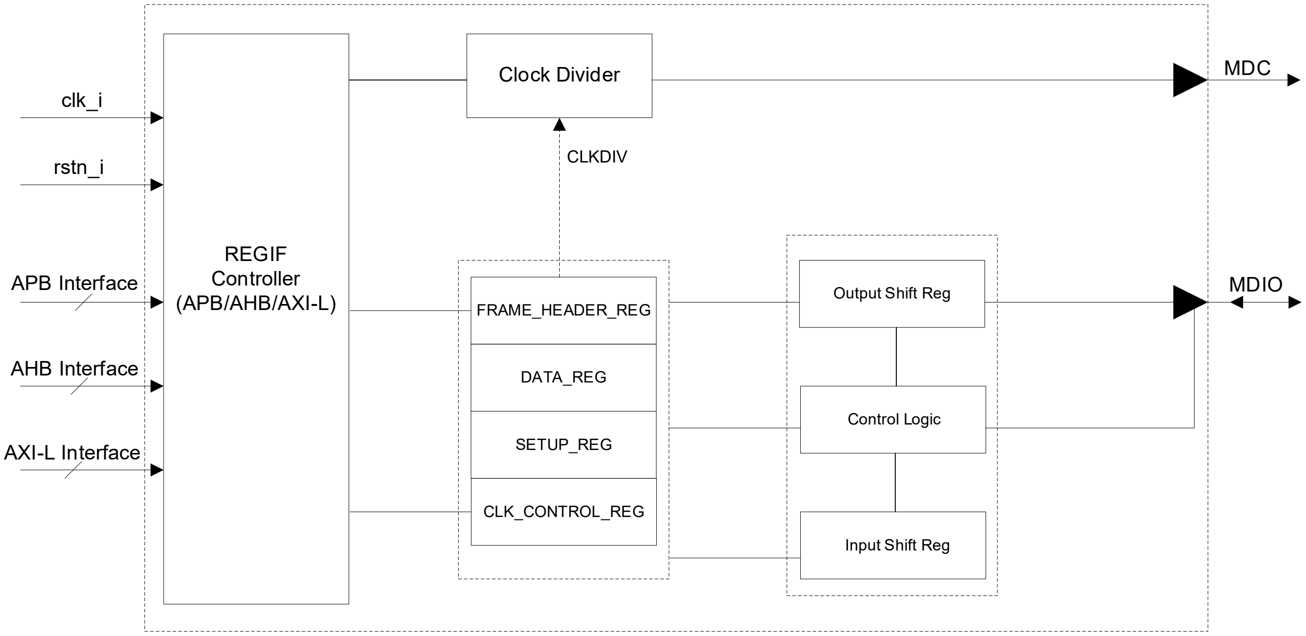


Figure 2.1.MDIO Leader Functional Block Diagram

2.1.1. Clock Divider Functional Overview

The following lists the functions of the Clock Divider (CLKDIV):

- You can configure the CLKDIV settings through the register interface. If not configured, a parameterized control value is used.
- The MDC clock signal is driven high for the reference count cycles (CLKDIV) with respect to *clk_i* and it is made low reference count cycles with respect to *clk_i*.
- This gives you the MDC signal frequency of clk_i/CLKDIVx2.
- The valid range for CLKDIV is 1 to 63.
- In case the value of CLKDIV is set to 0, there is no clock division and the MDC clock is the same as the *clk_i*.

Equation 1: MDC Clock Frequency Equation

$$fMDC \ = \ fclk\_i/[(Clock\ Divider\ Settings[5:0]) \times 2]$$

2.2. Signal Description

Table 2.1 shows the list of MDIO Leader ports.

Table 2.1. MDIO Leader IP Core Signal Description

Port Name	I/O	Width	Description
Common Clock Reset Signals			
clk_i	clk_i	clk_i	System clock input to drive the MDIO Leader IP.
rstn_i	rstn_i	rstn_i	Active-low asynchronous reset signal. Resets internal logic of the MDIO Leader IP core when asserted.
MDIO Interface (When MDIO Buffer Enable Selection= 1 Tristate Buffer is inside the FPGA Core Logic)			
MDC	Output	1	MDIO interface clock and has always the frequency as per equation 1. Make sure that CLKDIV is selected in such a manner that MDC is always restricted to a max frequency of 2.5 MHz.
MDIO	Inout	1	Bidirectional MDIO interface data.
MDIO Interface (When MDIO Buffer Enable = 0 Tristate Buffer is Outside the FPGA Core Logic)			
MDIO_IN	Input	1	MDIO Interface Input to FPGA.
MDIO_OUT	Output	1	MDIO Interface Output from FPGA.
MDIO_OEN	Output	1	MDIO Interface Output Enable from FPGA for External Tristate Buffer.
MDC	Output	1	MDIO interface clock and has always the frequency as per equation 1. Make sure that CLKDIV is selected in such a manner that MDC is always restricted to a maximum frequency of 2.5 MHz.
AHB Interface Signals (These signals are only present when Host Interface Selection = 0)			
ahbl_hsel_slv_i	Input	1	AHBL Follower Select Signal.
ahbl_adr_i [4:0]	Input	32	Register Address bus to the core.
ahbl_dat_i [31:0]	Input	32	Register Interface Data towards the core.
ahbl_rdat_o [31:0]	Output	32	Read Data from the core.
ahbl_htrans_slv_i [1:0]	Input	2	AHBL Transfer type of the current transfer (IDLE/BUSY/NSEQ/SEQ) 00 – Idle, 01 – Busy, 10 – NSEQ, 11 – SEQ In the current design, there is a requirement to support Single Transfer. The value of this signal must always be “10”. Other types are not supported.

Port Name	I/O	Width	Description
ahbl_hwrite_slv_i	Input	1	Interface Command Type 0 – Read, 1 – Write
ahbl_hready_slv_i	Input	1	AHBL indicates transfer completion.
ahbl_hreadyout_slv_o	Output	1	When HIGH, the HREADYOUT signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
ahbl_hresp_slv_o	Output	1	The transfer response, after passing through the multiplexor, provides the leader with additional information on the status of a transfer. 0 – OKAY 1 – ERROR
APB Interface Signals (These signals are only present when Host Interface Selection = 1)			
apb_psel_i	Input	1	It indicates that the completer device is selected and that a data transfer is required.
apb_adr_i [4:0]	Input	32	Register Address bus to the core.
apb_dat_i [31:0]	Input	32	Register Interface Data towards the core.
apb_rdat_o [31:0]	Output	32	Read Data from the core.
apb_pwrite_i	Input	1	This signal indicates an APB write access when HIGH and an APB read access when LOW. 0 – Read 1 – Write
apb_penable_i	Input	1	This signal indicates the second and subsequent cycles of an APB transfer.
apb_pready_o	Output	1	The completer device uses this signal to extend an APB transfer.
apb_slv_err_o	Output	1	Indicates transaction is send to the wrong Offset.
AXI-L Interface Signals (These signals are only present when Host Interface Selection = 2)			
axi_awvalid_i	Input	1	Write address valid. The Leader generates this signal when Write Address and control signals are valid.
axi_awready_o	Output	1	Write address ready. The AXI follower of the MDIO leader generates this signal when it can accept Write Address and control signals.
axi_awprot_i [2:0]	Input	3	Protection type. The AXI follower of the MDIO leader IP usually ignores, and Leader IP generates transactions with Normal, Secure and Data attributes.
axi_wvalid_i	Input	1	Write address valid. The Leader generates this signal when Write Address and control signals are valid.
axi_aw_addr_i	Input	32	Write Register Address.
axi_wready_o	Output	1	Write address ready. The AXI follower of the MDIO Leader generates this signal when it can accept Write Address and control signals.
axi_wstrb_i [3:0]	Input	4	Write Strobe per Byte When HIGH, specify the byte lanes of the data bus that contain valid information. There is one write strobe for each eight bits of the write data bus. The value of this signal must always be 4’b1111 as partial byte enable is not supported.
axi_bvalid_o	Output	1	Write response valid. The AXI follower of the MDIO Leader generates this signal when the write response on the bus is valid.
axi_bready_i	Input	1	Response ready. The Leader generates this signal when it can accept a write response.

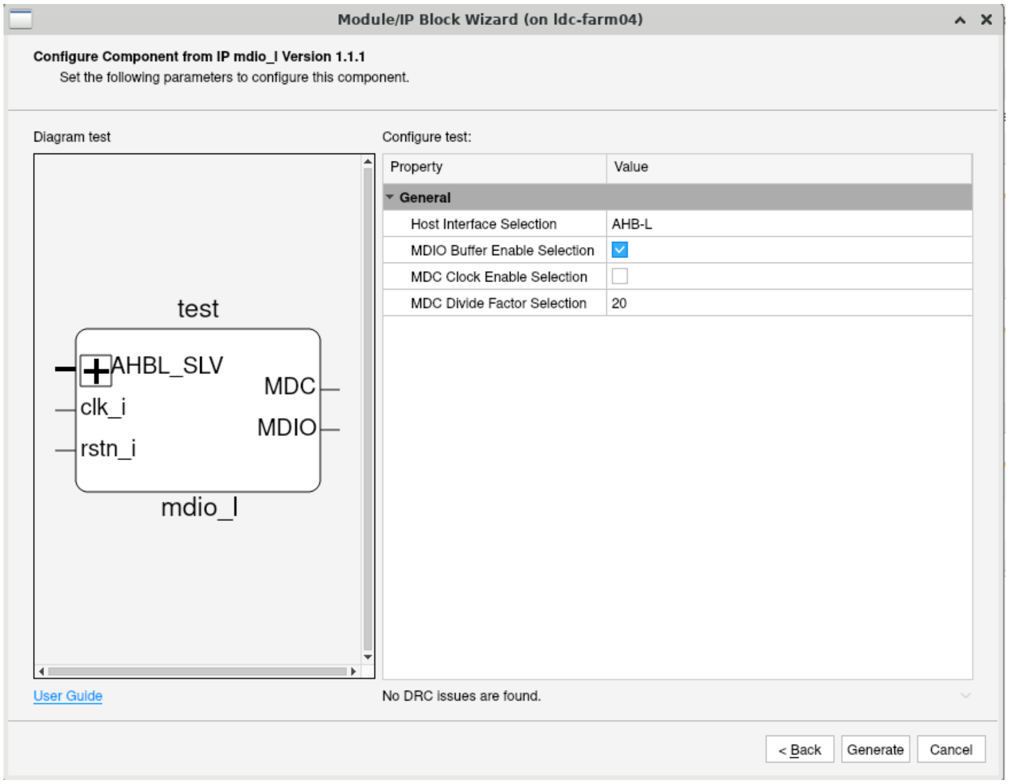
Port Name	I/O	Width	Description
axi_bresp_o [1:0]	Output	2	Write response. This signal indicates the status of the write transaction.
axi_arvalid_i	Input	1	Read address valid. The Leader generates this signal when Read Address and the control signals are valid.
axi_arready_o	Output	1	Read address ready. The AXI follower of the MDIO leader generates this signal when it can accept the read address and control signals.
axi_arprot_i [2:0]	Input	3	Protection type.
axi_rvalid_o	Output	1	Read address valid. The Leader generates this signal when Read Address and the control signals are valid.
axi_rready_i	Input	1	Read address ready. The AXI follower of the MDIO leader generates this signal when it can accept the read address and control signals.
axi_rresp_o [1:0]	Output	2	Read response. This signal indicates the status of data transfer.
axi_ar_addr_i [4:0]	Input	32	Read Register Address bus to the core.
axi_dat_i [31:0]	Input	32	Register Interface Data towards the core.
axi_rdat_o [31:0]	Output	32	Read Data from the core.

2.3. Attribute Summary

The configurable attributes of the MDIO Leader IP core are shown in table below.
The attributes can be configured through the IP Catalog's Module/IP wizard of the Lattice Propel Builder.

Table 2.2. Attributes Table

Attribute	Selectable Values	Default	Allowable Values
General			
MDIO Buffer Enable Selection	Editable	1	0 and 1.
Host Interface Selection	Editable	AHB-L	AHB-L, APB, and AXI-L.
MDC Clock Enable Selection	Editable	0	The Parameterized control for the default value of MDC Clock Enable. Allowable values of 0 and 1.
MDC Divide Factor Selection	Editable	20	The Parameterized Default Value of the MDC Clock Divider Factor to make a 2.5 MHz clock using the input clock clk_i. Allowed values are: • 5 to 25 MHz input clk • 10 to 50 MHz input clk • 15 to 75 MHz input clk • 20 to 100 MHz input clk • 25 to 125 MHz input clk • 30 to 150 MHz Input clk • 40 to 200 MHz input clk



Configure Component from IP mdio_l Version 1.1.1

Set the following parameters to configure this component.

Property	Value
General	
Host Interface Selection	AHB-L
MDIO Buffer Enable Selection	☑
MDC Clock Enable Selection	□
MDC Divide Factor Selection	20

No DRC issues are found.

Figure 2.2. Lattice Radiant Software IP Wizard Reference

2.4. Register Description

Table 2.3. MDIO Registers

Address (Hex)	Name
0x0000	MDIO Frame Header Register (*FRAME_HEADER_REG*)
0x0004	MDIO Write Data and Read Data Register (*DATA_REG*)
0x0008	MDIO Frame Setup and Status Register (*SETUP_REG*)
0x000C	MDIO Clocking Control Register (*CLK_CONTROL_REG*)

There are four registers in the IP core. These are configured by any of the three interfaces through APB, AHB, or AXI-L.

Table 2.4. Register Details for MDIO Frame Header Register (0x0000)

Bits	Default Value	Type	Description
31:21	0x0	R/W	Phy_Reg_Address [15:5] Higher [15:5] bits of PHY register address field, used only for Clause 45.
20:16	0x0	R/W	Phy_Reg_Address [4:0] Lower [4:0] bits of PHY register address field, used for both Clause 22 and 45.
15	0x0	R/W	Clause_Select Select between Clause 22 and Clause 45. 1'b0 – Clause 45 1'b1 – Clause 22
14	0x0	R/W	IS_NO_PRE Used only for Clause 22. Selects if preamble is required in generated MDIO frames. Preamble is always generated for Clause 45 frames. 1'b0 – Preamble is required. 1'b1 – Preamble is not required
13:10	0x0	RESERVED	—
9:5	0x0	R/W	Phy_Address 5-bit PHY Address field, used for both Clause 22 and 45.
4:0	0x0	R/W	Device_Type 5-bit Device Type field, used only for Clause 45.

Note: Number of registers is set to 4.

Table 2.5. Register Details for MDIO Write Data and Read Data Register (0x0004)

Bits	Default Value	Type	Description
31:16	0x0	R/O	MDIO PHY Reg Read Data This is 16 bits Data Read from the PHY device using the read frames.
15:0	0x0	R/W	MDIO PHY Reg Write Data Data to be written using the write frame.

Note: Number of registers is set to 4.

Table 2.6. Register Details for MDIO Frame Setup and Status Register (0x0008)

Bits	Default Value	Type	Description
31:17	0x0	RESERVED	—
16	0x0	R/O	Transaction Done. This indicates Read or Write transaction is done. This bit is cleared after you set the Bit [2] of the register (Start_Transaction).
15:3	0x0	RESERVED	—
2	0x0	Auto Clear	Start_Transaction
1:0	0x0	R/W	Transaction Opcode Definition. Bit [1] – This is used only when Clause_Select bit is 0; otherwise, it is Don't Care {Bit [15] at offset 0x04}. For Clause_Select = 1 [Clause_22], Bit [0] – '0' Means Read, Bit [0] – '1' Means Write. For Clause_Select = 0 [Clause_45] Definition of Bit [1:0] is as follows: 00 – Write access with Opcode 01 to address which was previously used. 01 – Read + Address increment access with Opcode 10. 10 – Address Frame with Opcode 00 is sent followed by Write with opcode 01 to address sent in address frame. 11 – Address Frame with Opcode 00 is sent followed by read with opcode 11 to address sent in address frame.

Table 2.7. Register Details for MDIO Clocking Control Register (0x000C)

Bits	Default Value	Type	Description
31:17	0x0	RESERVED	—
16	MDC Clock Enable Selection	R/W	Clocking Control Bit 0 – Disable MDIO Clock 1 – Enable MDIO Clock. MDC Clock starts only when this bit is set. If MDC_DIV_FACTOR is 0, then clock is disabled.
15:6	0x0	RESERVED	—
5:0	MDC_DIV_FACTOR	R/W	Clock Divider Settings If MDC_DIV_FACTOR is 0, the MDC clock is mapped directly to clk_i without the division.

2.5. Operation Details

2.5.1. Clock Divider Settings

The default value of CLKDIV settings are used to parameterize the control given to you in the Lattice Radiant software and Lattice Propel software user interface. You can also control whether to enable the clock by default or disable it either by default from parameter or later using the register settings bit [16] of clk_control register. You can also select the CLKDIV settings from the register interface clk_control [5:0] bits. You must select the clock settings in such a way that MDC frequency is less than 2.5 MHz.

Before going to further settings, the clk_control setting must be done initially to enable the MDC clock.

The relation between the input clock frequency and MDC frequency is given below:

$$fMDC \; = \; fclk\_i/[(Clock\ Divider\ Settings[5:0])x2].$$

The valid range for CLKDIV is 1 to 63. In case the value of CLKDIV is set to 0, then there is no clock division and the MDC clock is the same as the clk_i.

2.5.2. Clause 22 Leader

For Clause 22, bit [15] of the frame header register (0x8000 offset) is set to 1 and all other settings are done as per requirement. For setting up the write frame, bit [0] of the frame setup and status register (0x8008 offset) is set to 0. For setting up the read frame, bit [0] is set to value 1. For configuring the device and PHY address settings, the frame header register is used. In case of write frame, the write data is set up using the Data_Reg [15:0].

After doing the frame setup settings, you must write to bit [2] of Setup_Reg and the MDIO frame is generated. The transaction done status can be read using the Setup_Reg bit [16]. After this bit is set, the read data in case of MDIO read frame can be read from the register interface using Data_Reg [31:16].

2.5.3. Clause 45 Leader

For Clause 45, bit [15] of the frame header register (0x8000 offset) is set to 0 and all other settings are done as per requirement.

The following are the settings for different frame formats Setup_Reg [1:0]:

- 00 – Write access with Opcode 01 to address that was previously used
- 01 – Read + Address increment access with Opcode 10
- 10 – Address Frame with Opcode 00 is sent followed by write with opcode 01 to address sent in address frame
- 11 – Address Frame with Opcode 00 is sent followed by read with opcode 11 to address sent in address frame

In case of write frame, the write data is set up using the Data_Reg [15:0].

After doing the frame setup settings, you must write to bit [2] of Setup_Reg and the MDIO frame is generated. The transaction done status can be read using the Setup_Reg bit [16]. After this bit is set, the read data in case of MDIO read frame can be read from the register interface using Data_Reg [31:16].

2.6. Programming Flow

2.6.1. Reading and Writing Clause 22

To read and write Clause 22, perform the following steps:

1. Perform the clock divider settings using the clk_control register.
2. Perform the frame header configuration using the Frame_Header register. Unique bit for Clause 22 is bit[15], which is set as 1.
3. Perform the data configuration, if required, using Data_Reg.
4. Set the frame type to send using Setup_Reg.
5. Indicate transaction start using bit[2] of Setup_Reg.
6. Poll the status of transaction using the Transaction_Done bit [16] of Setup_Reg.
7. If required, read the data from the Data_Reg.

2.6.2. Reading and Writing Clause 45

To read and write Clause 45, perform the following steps:

1. Perform the clock divider settings using the clk_control register.
2. Perform the frame header configuration using the Frame_Header register. Unique bit for Clause 22 is bit[15], which is set as 0.
3. Perform the data configuration, if required, using Data_Reg.
4. Set the frame type to send using the Setup_Reg.
5. Indicate transaction start using bit[2] of Setup_Reg.
6. Poll the status of transaction using the Transaction_Done bit [16] of Setup_Reg.
7. If required, read the data from the Data_Reg.

2.7. Data Format

The APB, AXI-L, and AHBL Follower interfaces support standard data format as defined in Advanced Microcontroller Bus Architecture (AMBA) specification. Refer to the ARM AMBA User Guide for more information on the protocol. The MDIO interface is based on the IEEE 802.3 Specification.

2.8. Timing Diagrams

Figure 2.3 and Figure 2.4 show the timing diagrams for APB Write and Read Transfer.

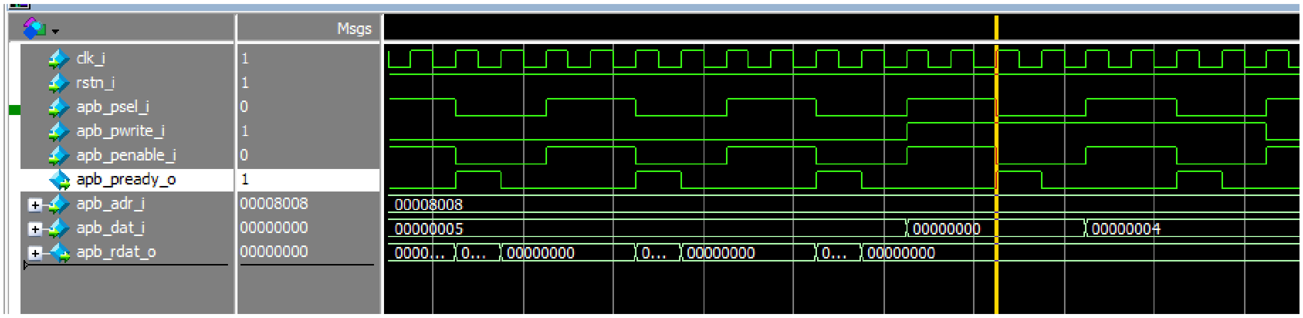


Figure 2.3. Basic APB Write Transfer

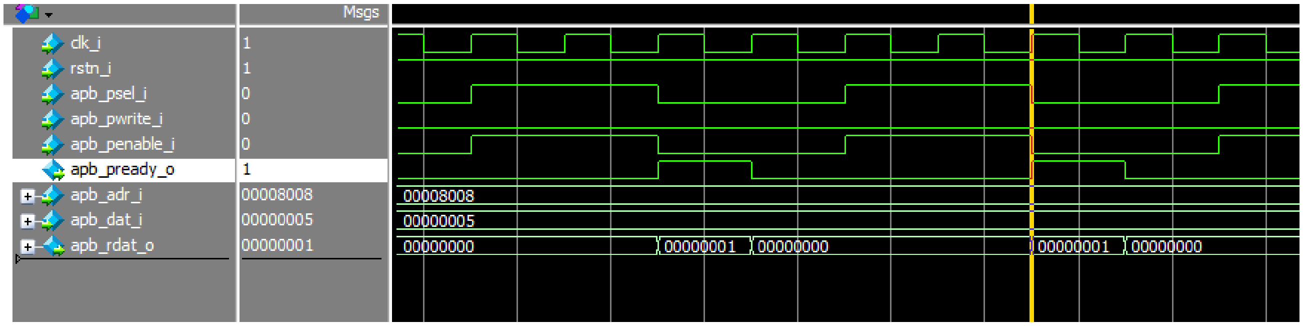
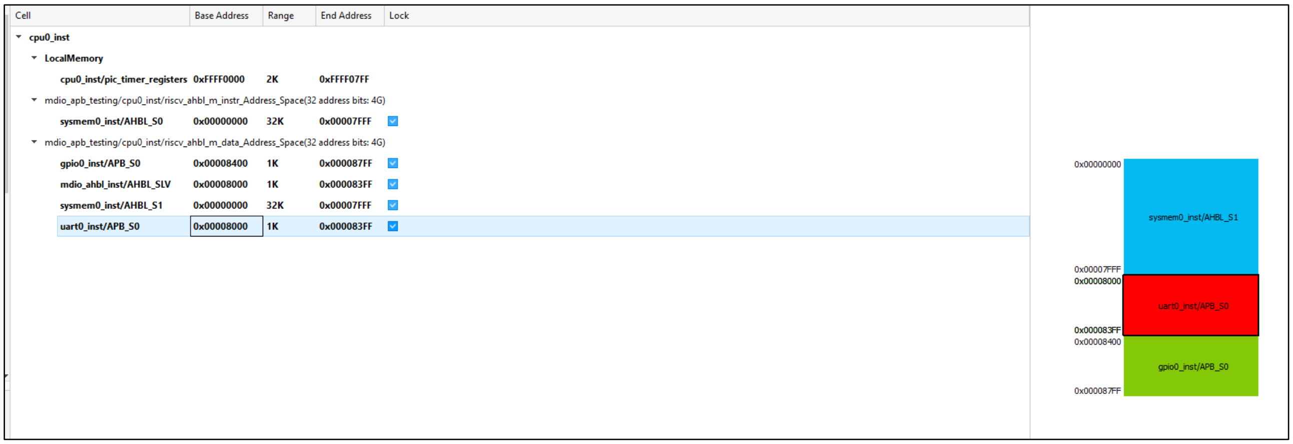


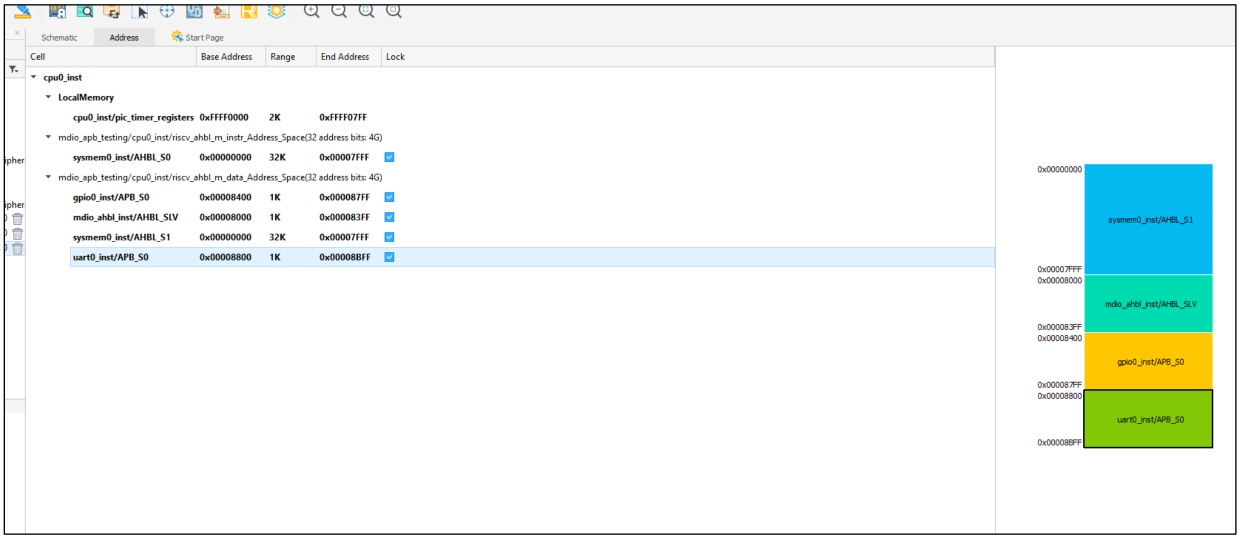
Figure 2.4. Basic APB Read Transfer

Figure 2.5 and Figure 2.6 show the timing diagrams for AHBL Write and Read.



Cell	Base Address	Range	End Address	Lock
cpu0_inst				
LocalMemory				
cpu0_inst/pic_timer_registers	0xFFFF0000	2K	0xFFFF07FF	
mdio_apb_testing/cpu0_inst/riscv_ahbl_m_instr_Address_Space(32 address bits: 4G)				
sysmem0_inst/AHBL_S0	0x00000000	32K	0x00007FFF	☑
mdio_apb_testing/cpu0_inst/riscv_ahbl_m_data_Address_Space(32 address bits: 4G)				
gpio0_inst/APB_S0	0x00008400	1K	0x000087FF	☑
mdio_ahbl_inst/AHBL_SLV	0x00008000	1K	0x000083FF	☑
sysmem0_inst/AHBL_S1	0x00000000	32K	0x00007FFF	☑
uart0_inst/APB_S0	0x00008000	1K	0x000083FF	☑

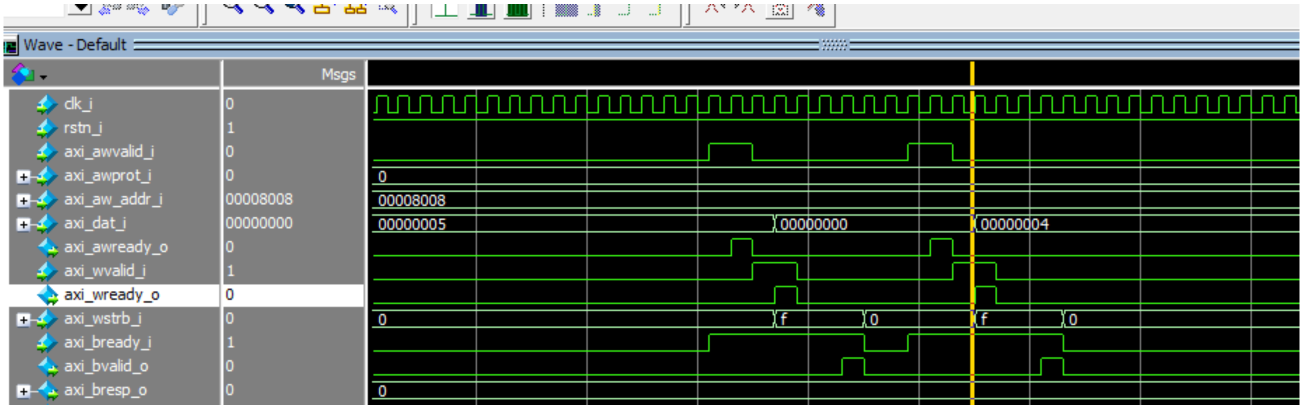
Figure 2.5. Basic AHBL Write



Cell	Base Address	Range	End Address	Lock
cpu0_inst				
LocalMemory				
cpu0_inst/pic_timer_registers	0xFFFF0000	2K	0xFFFF07FF	
mdio_apb_testing/cpu0_inst/riscv_ahbl_m_instr_Address_Space(32 address bits: 4G)				
sysmem0_inst/AHBL_S0	0x00000000	32K	0x00007FFF	☑
mdio_apb_testing/cpu0_inst/riscv_ahbl_m_data_Address_Space(32 address bits: 4G)				
gpio0_inst/APB_S0	0x00008400	1K	0x000087FF	☑
mdio_ahbl_inst/AHBL_SLV	0x00008000	1K	0x000083FF	☑
sysmem0_inst/AHBL_S1	0x00000000	32K	0x00007FFF	☑
uart0_inst/APB_S0	0x00008800	1K	0x00008BFF	☑

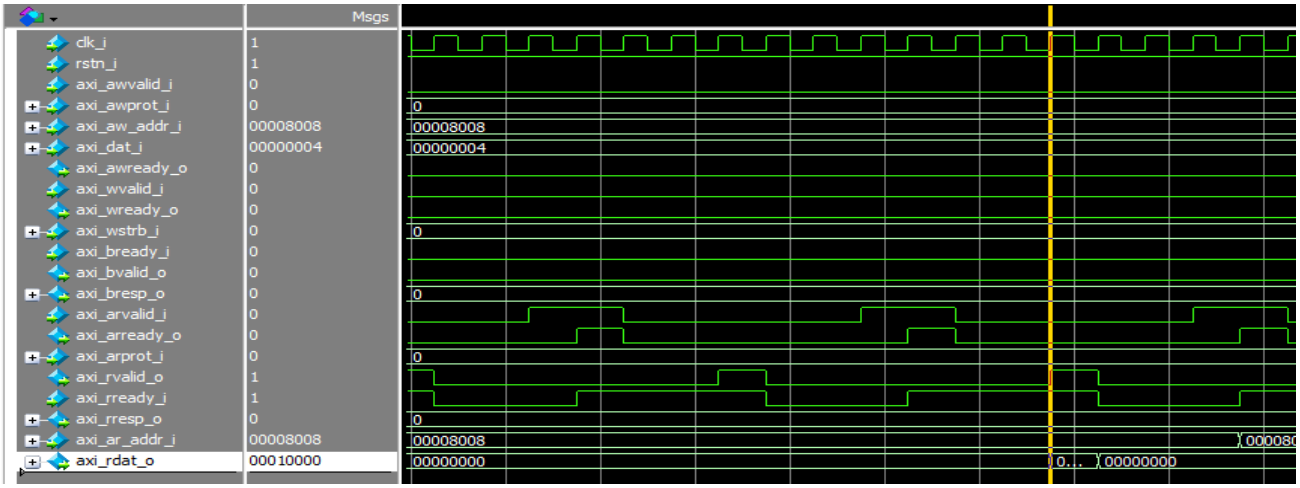
Figure 2.6. Basic AHBL Read

Figure 2.7 and Figure 2.8 show the timing diagrams for AXI Write and Read.



	Msgs
clk_i	0
rstn_i	1
axi_awvalid_i	0
axi_awprot_i	0
axi_aw_addr_i	00008008
axi_dat_i	00000000
axi_awready_o	0
axi_wvalid_i	1
axi_wready_o	0
axi_wstrb_i	0
axi_bready_i	1
axi_bvalid_o	0
axi_bresp_o	0

Figure 2.7. Basic AXI Write



	Msgs
clk_i	1
rstn_i	1
axi_awvalid_i	0
axi_awprot_i	0
axi_aw_addr_i	00008008
axi_dat_i	00000004
axi_awready_o	0
axi_wvalid_i	0
axi_wready_o	0
axi_wstrb_i	0
axi_bready_i	0
axi_bvalid_o	0
axi_bresp_o	0
axi_arvalid_i	0
axi_arready_o	0
axi_arprot_i	0
axi_rvalid_o	1
axi_rready_i	1
axi_rresp_o	0
axi_ar_addr_i	00008008
axi_rdat_o	00010000

Figure 2.8. Basic AXI Read

Figure 2.9 shows the timing diagram for MDIO transactions.

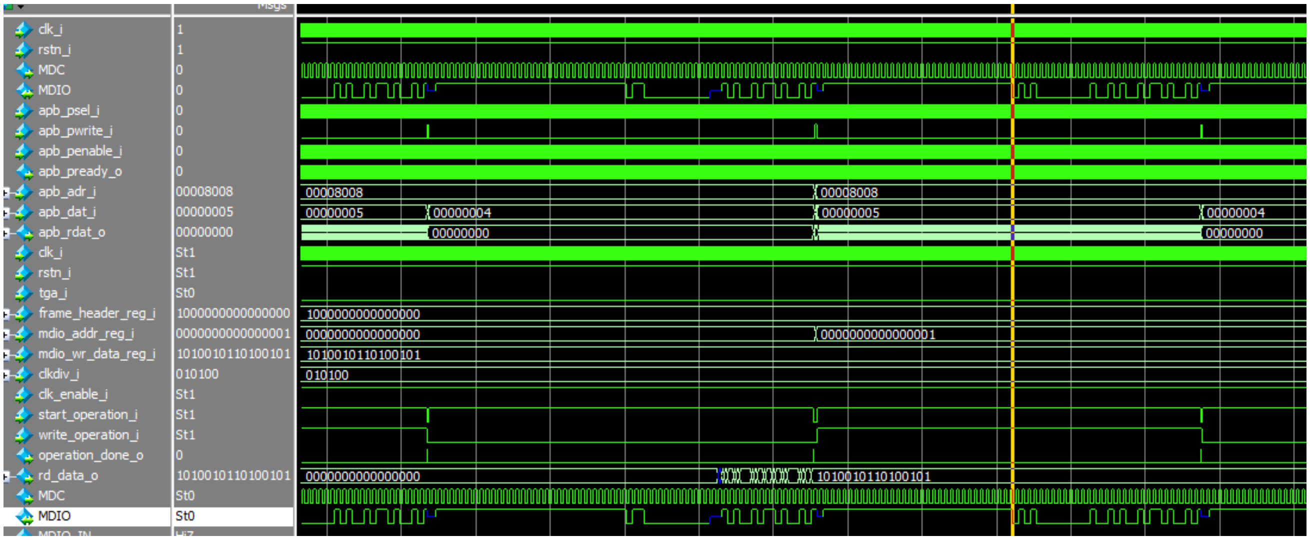


Figure 2.9. MDIO Transaction After Writing 1 to Start Operation Bit

2.9. IP Generation, Simulation, and Validation

This section provides information on how to generate the MDIO Leader IP core using the Lattice Radiant software and how to run synthesis and simulation. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide.

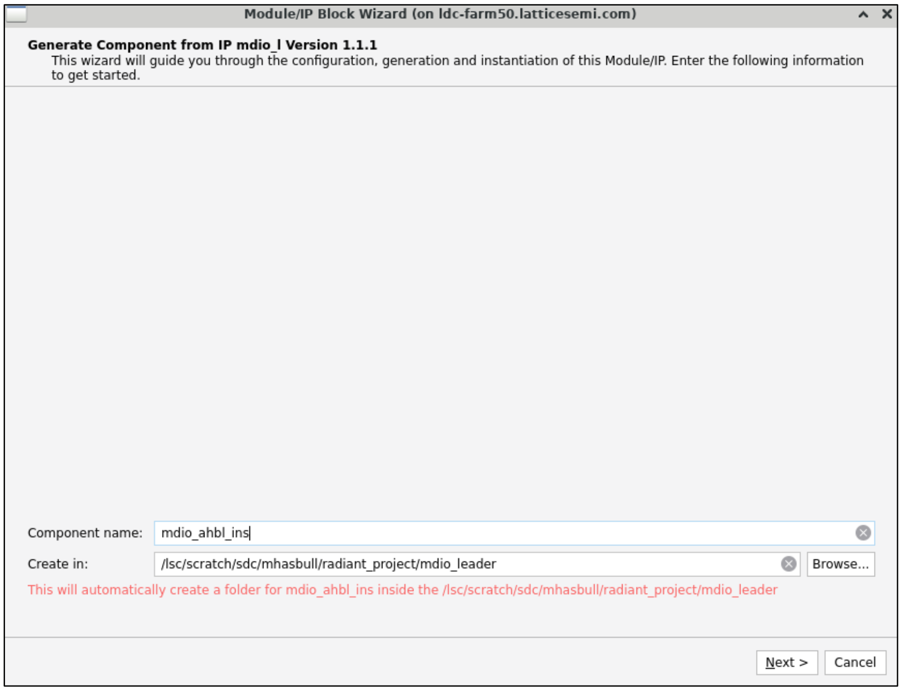
Note: The screenshots provided are for reference only. Details may vary depending on the version of the IP or software being used. If there have been no significant changes to the GUI, a screenshot may reflect an earlier version of the IP.

2.9.1. Generating the IP

The Lattice Radiant software allows you to customize and generate modules and IP cores and integrate them into the device architecture. The procedure for generating the MDIO Leader IP core in the Lattice Radiant software is described below.

To generate the MDIO Leader IP core, follow these steps:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click on **MDIO Leader** under **IP, Processor_Controllers_and_Peripherals** category. The Module/IP Block Wizard opens as shown in Figure 2.10. Enter values in the **Instance name** and the **Create in** fields and click **Next**.



Module/IP Block Wizard (on ldc-farm50.latticesemi.com)

Generate Component from IP mdio_l Version 1.1.1

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

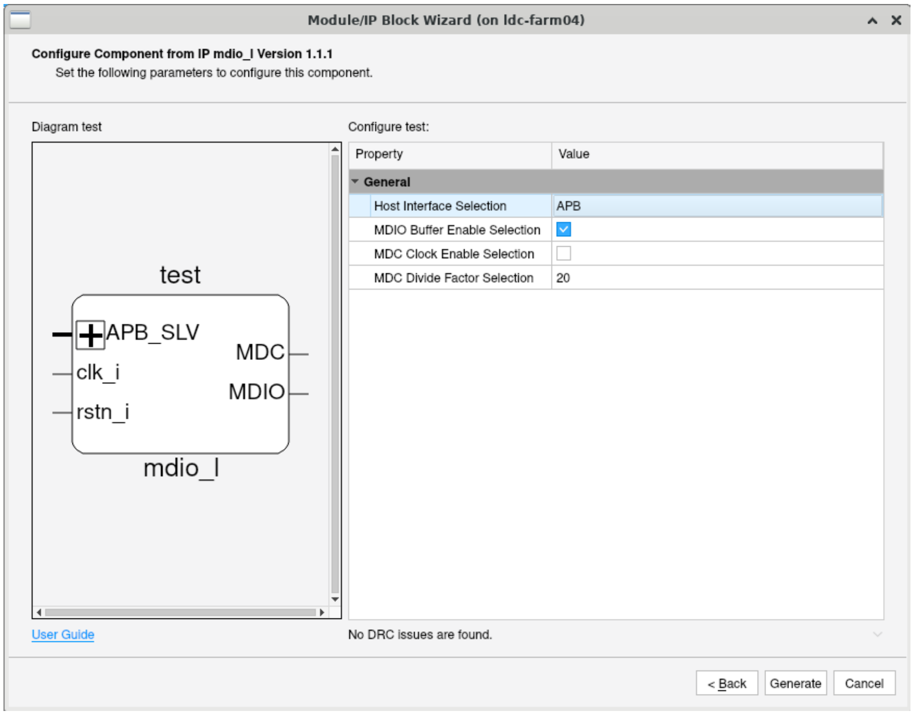
Component name: mdio_ahbl_ins

Create in: /lsc/scratch/sdc/mhasbull/radiant_project/mdio_leader

This will automatically create a folder for mdio_ahbl_ins inside the /lsc/scratch/sdc/mhasbull/radiant_project/mdio_leader

Figure 2.10. Module/IP Block Wizard

3. In the module dialog box of the **Module/IP Block Wizard** window, customize the selected MDIO Leader IP core. As a sample configuration, see Figure 2.11. For configuration options, see the Attribute Summary section.



Module/IP Block Wizard (on ldc-farm04)

Configure Component from IP mdio_l Version 1.1.1

Set the following parameters to configure this component.

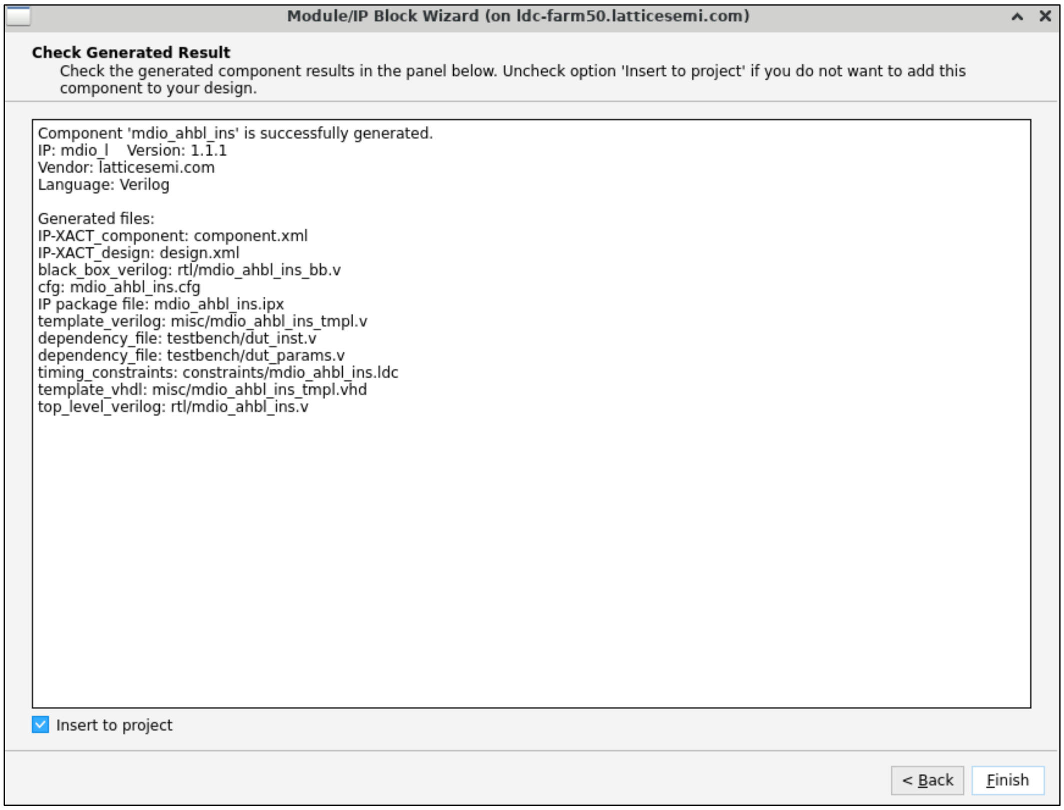
Configure test:

Property	Value
General	
Host Interface Selection	APB
MDIO Buffer Enable Selection	☑
MDC Clock Enable Selection	□
MDC Divide Factor Selection	20

No DRC issues are found.

Figure 2.11. Configure User Interface of MDIO Leader IP

4. Click **Generate**. The **Check Generating Result** dialog box opens, showing design block messages and results as shown in Figure 2.12.



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 'mdio_ahbl_ins' is successfully generated.
IP: mdio_l    Version: 1.1.1
Vendor: latticesemi.com
Language: Verilog

Generated files:
IP-XACT_component: component.xml
IP-XACT_design: design.xml
black_box_verilog: rtl/mdio_ahbl_ins_bb.v
cfg: mdio_ahbl_ins.cfg
IP package file: mdio_ahbl_ins.ipx
template_verilog: misc/mdio_ahbl_ins_tmpl.v
dependency_file: testbench/dut_inst.v
dependency_file: testbench/dut_params.v
timing_constraints: constraints/mdio_ahbl_ins.ldc
template_vhdl: misc/mdio_ahbl_ins_tmpl.vhd
top_level_verilog: rtl/mdio_ahbl_ins.v
```

☑ Insert to project

Figure 2.12. Check Generating Results

5. Click the **Finish** button. All the generated files are placed under the directory paths in the **Create in** and the **Instance name** fields shown in Figure 2.10.

The generated MDIO Leader IP core package includes the closed-box (*_bb.v*) and instance templates (*_tmpl.v/vhd*) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file (*.v*) that can be used as an instantiation template for the IP core is also provided. You can also use this top-level reference as the starting template for your top-level design. The generated files are listed in Table 2.8.

Table 2.8. Generated File List

Attribute	Description
<Instance Name>.ipx	This file contains information on the files associated to the generated IP.
<Instance 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/<Instance Name>.v	This file provides an example RTL top file that instantiates the IP core.
rtl/<Instance Name>_bb. v	This file provides the synthesis closed-box.
misc/<Instance Name>_tmpl.v *misc /<Instance Name>_tmpl.vhd*	These files provide instance templates for the IP core.
constraints/<Instance Name>.sdc	Constraint file.

2.9.2. Running Functional Simulation

Running functional simulation can be performed after the IP is generated.

To run Verilog simulation, follow these steps:

1. Click the button located on the **Toolbar** to initiate the **Simulation Wizard** shown in Figure 2.13.

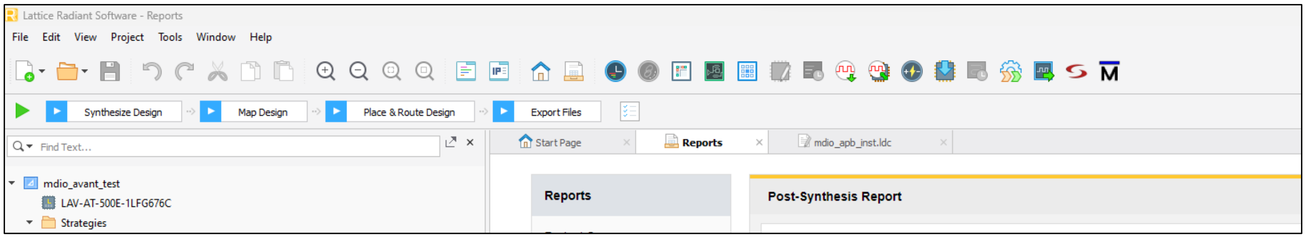
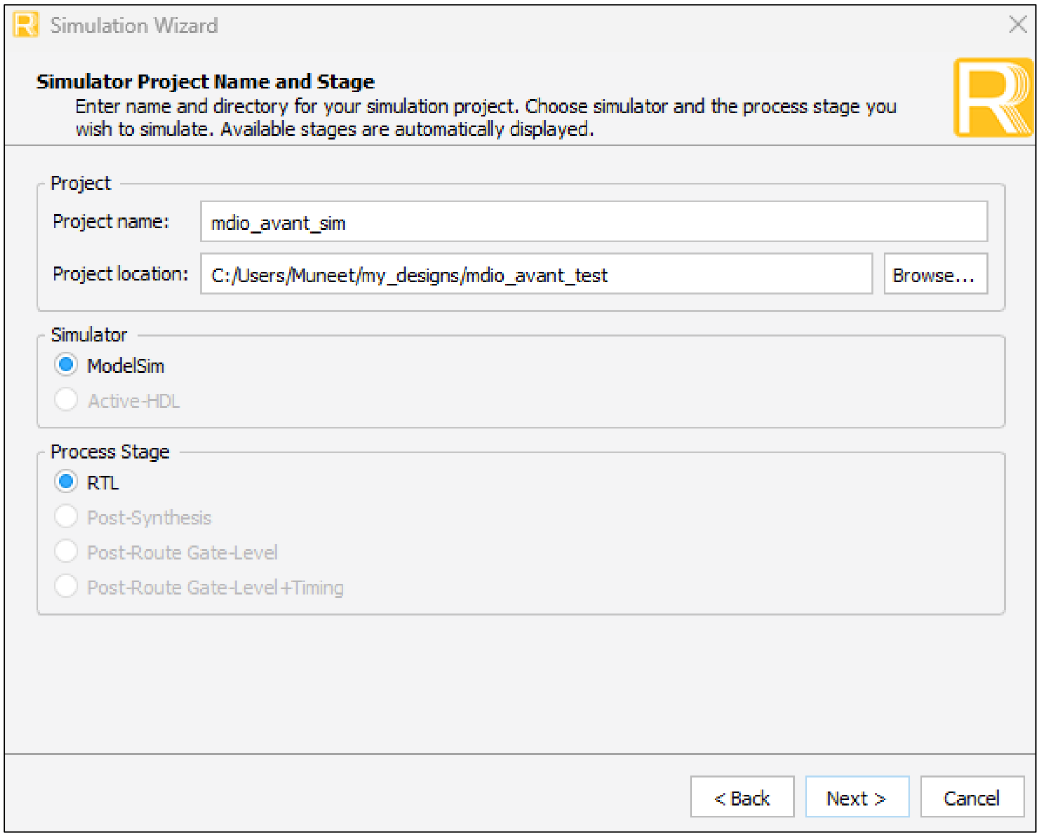


Figure 2.13. Toolbar Tab

2. The Simulation Wizard opens. Click **Next** to open the main wizard for **Simulation Project and Simulator** settings as shown in Figure 2.14. Enter the project name and click **Next**.



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: mdio_avant_sim

Project location: C:/Users/Muneet/my_designs/mdio_avant_test

Simulator

● ModelSim

○ Active-HDL

Process Stage

● RTL

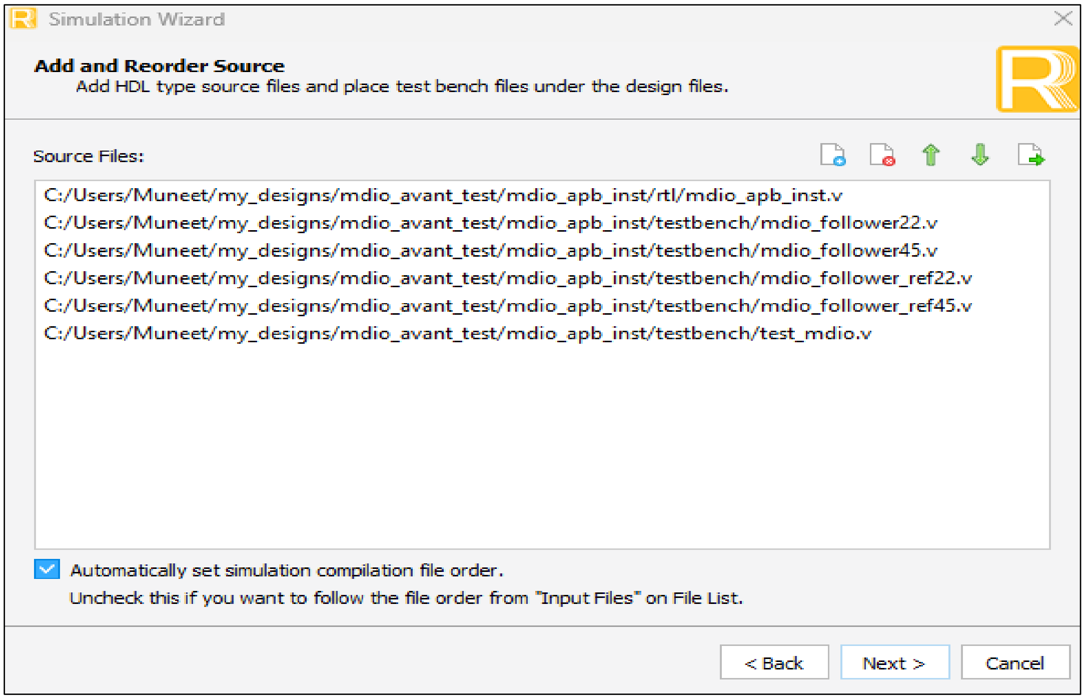
○ Post-Synthesis

○ Post-Route Gate-Level

○ Post-Route Gate-Level+Timing

Figure 2.14. Simulation Wizard Project Settings

3. Click **Next** to open the **Add and Reorder Source** window as shown in Figure 2.15.



Add and Reorder Source

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

Source Files:

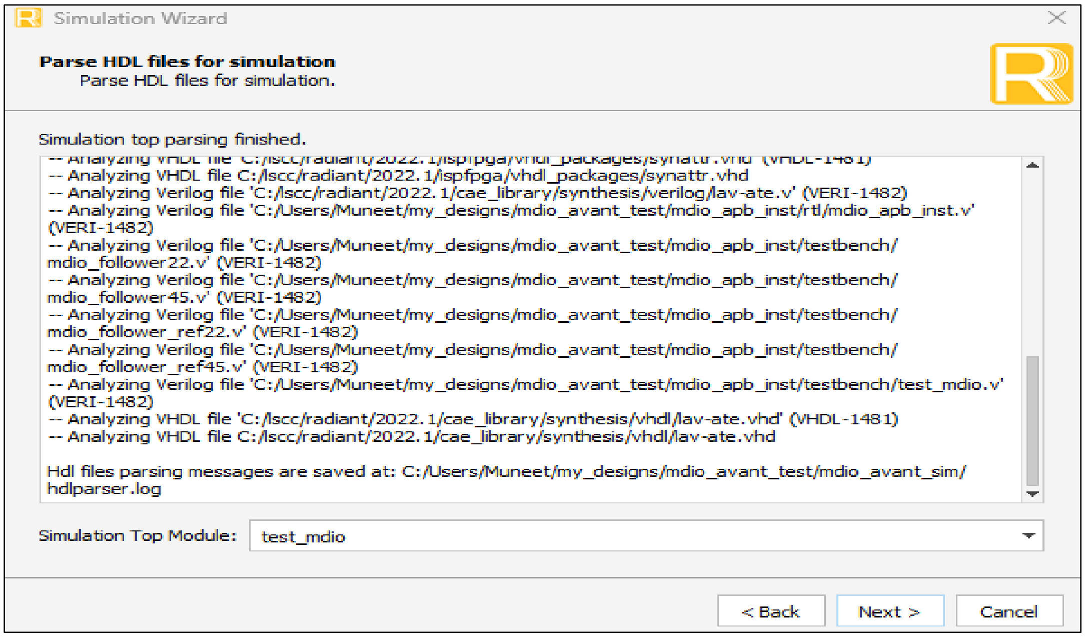
```
C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/rtl/mdio_apb_inst.v
C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/mdio_follower22.v
C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/mdio_follower45.v
C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/mdio_follower_ref22.v
C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/mdio_follower_ref45.v
C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/test_mdio.v
```

☑ Automatically set simulation compilation file order.

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

Figure 2.15. Adding Re-Ordering Source

4. Click **Next**. The Parse HDL files for simulation window are shown in Figure 2.16.



Parse HDL files for simulation

Parse HDL files for simulation.

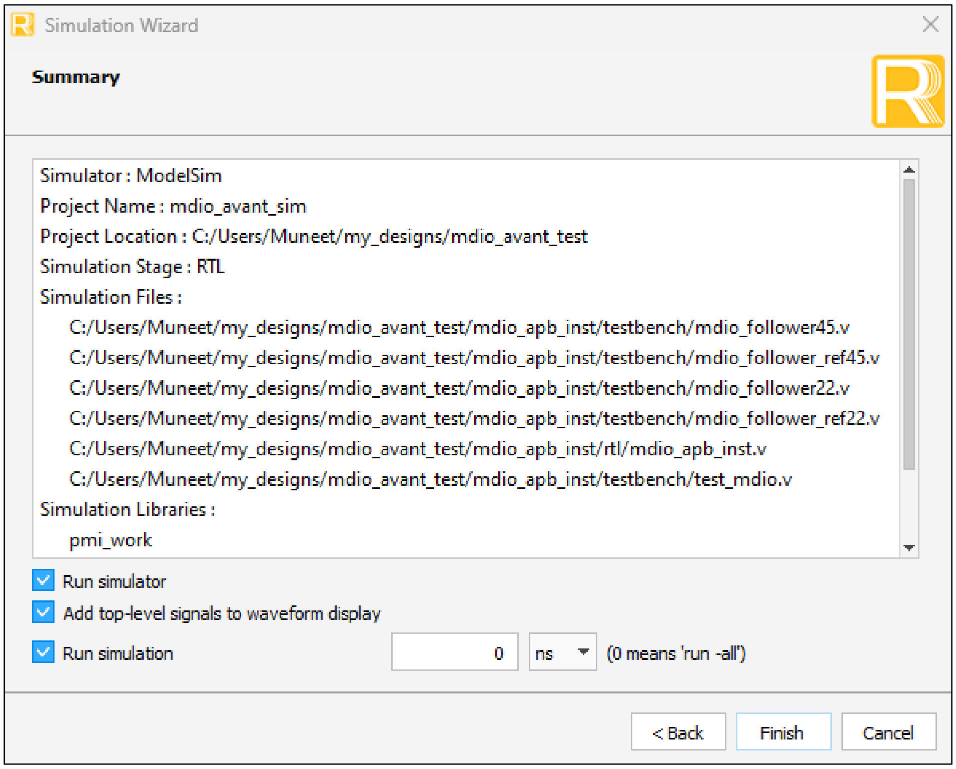
Simulation top parsing finished.

```
-- Analyzing VHDL file 'C:/lscc/radiant/2022.1/ispfpga/vhdl_packages/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/lav-ate.v' (VERI-1482)
-- Analyzing Verilog file 'C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/rtl/mdio_apb_inst.v'
(VERI-1482)
-- Analyzing Verilog file 'C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/
mdio_follower22.v' (VERI-1482)
-- Analyzing Verilog file 'C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/
mdio_follower45.v' (VERI-1482)
-- Analyzing Verilog file 'C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/
mdio_follower_ref22.v' (VERI-1482)
-- Analyzing Verilog file 'C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/
mdio_follower_ref45.v' (VERI-1482)
-- Analyzing Verilog file 'C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/test_mdio.v'
(VERI-1482)
-- Analyzing VHDL file 'C:/lscc/radiant/2022.1/cae_library/synthesis/vhdl/lav-ate.vhd' (VHDL-1481)
-- Analyzing VHDL file C:/lscc/radiant/2022.1/cae_library/synthesis/vhdl/lav-ate.vhd

Hdl files parsing messages are saved at: C:/Users/Muneet/my_designs/mdio_avant_test/mdio_avant_sim/
hdlparser.log
```

Simulation Top Module: test_mdio

Figure 2.16. Parse HDL Files for Simulation

5. Click **Next**. The Summary window is shown. Set the time to **0** to select **run -all**, and click **Finish** to run the simulation.



Simulator : ModelSim
Project Name : mdio_avant_sim
Project Location : C:/Users/Muneet/my_designs/mdio_avant_test
Simulation Stage : RTL
Simulation Files :
    C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/mdio_follower45.v
    C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/mdio_follower_ref45.v
    C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/mdio_follower22.v
    C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/mdio_follower_ref22.v
    C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/rtl/mdio_apb_inst.v
    C:/Users/Muneet/my_designs/mdio_avant_test/mdio_apb_inst/testbench/test_mdio.v
Simulation Libraries :
    pmi_work

☑ Run simulator

☑ Add top-level signals to waveform display

☑ Run simulation 0 ns (0 means 'run -all')

Figure 2.17. Summary

6. Check the ModelSim® software for the simulation results. The transcript file can directly print the Pass/Fail Status of the test case.

   Note: If you want to change the test case, change the TESTCASENUM parameter in the *test_mdio.v* file. Valid test cases are from TESTCASENUM value 1 to 7.

2.9.3. Constraining the IP

The */constraints/<Instance Name>.sdc* file described in Table 2.8 is generated based on the IP configuration you selected. The content of this file will be automatically included in the top-level design constraint file.

2.9.4. IP Evaluation

The IP core supports Lattice's IP evaluation capability when used in the supported FPGA family and targeted device. The IP evaluation capability may be enabled or disabled in the Strategy dialog box. It is disabled by default. To change this setting, go to **Strategies > Strategy1 (active strategy)> Bitstream**.

2.9.5. IP Validation

This IP has been validated using the CertusPro-NX device (LFCPNX-100 BFG484).

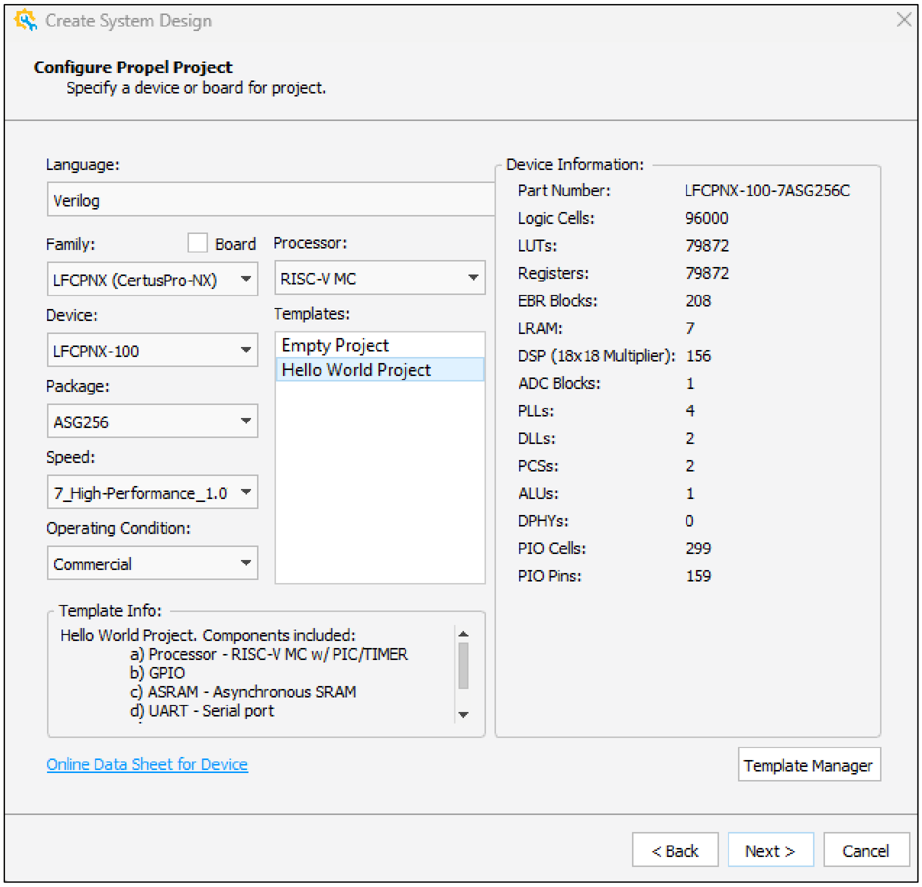
This IP has been validated using the Lattice Avant device advance timing models.

2.10. Propel Builder Example Design Steps

2.10.1. Generating the Project and Basic Instantiation

To generate the project, follow these steps:

1. Generate a new SoC project by clicking **File > New > Lattice SoC Design Project**.
2. Enter the project name and click **Next**. The configuration window opens as shown in Figure 2.18. Configure the Propel software project for the **LPCPNX** device in **Hello World Project** template**.**



Create System Design

Configure Propel Project
Specify a device or board for project.

Language: Verilog

Family: LFCPNX (CertusPro-NX)   □ Board

Processor: RISC-V MC

Device: LFCPNX-100

Templates: Empty Project, Hello World Project

Package: ASG256

Speed: 7_High-Performance_1.0

Operating Condition: Commercial

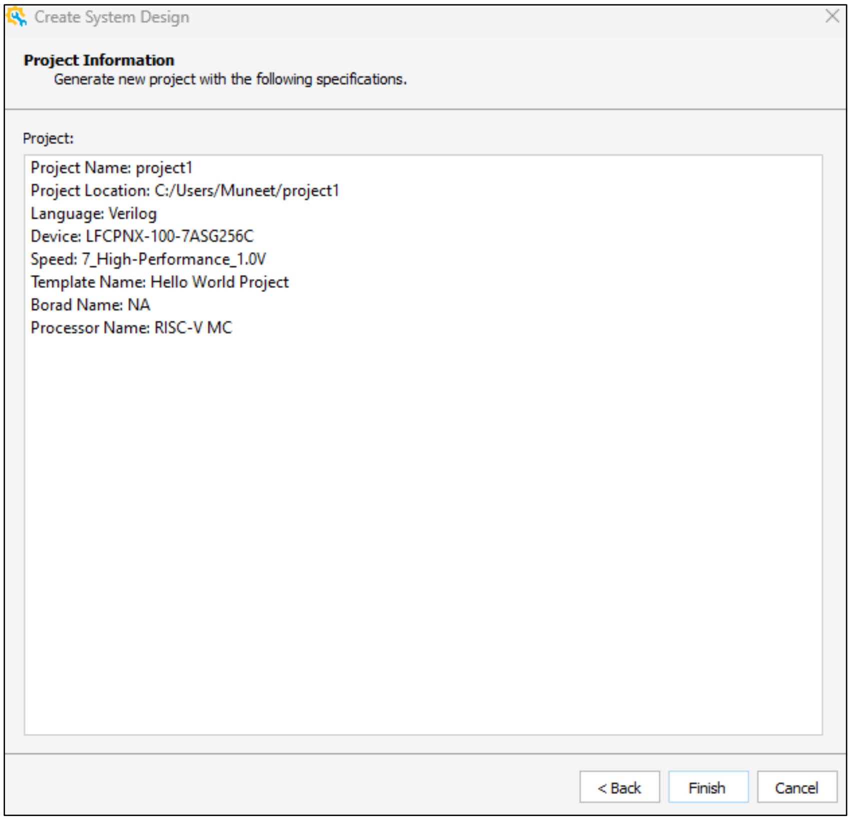
Device Information:

Part Number:	LFCPNX-100-7ASG256C
Logic Cells:	96000
LUTs:	79872
Registers:	79872
EBR Blocks:	208
LRAM:	7
DSP (18x18 Multiplier):	156
ADC Blocks:	1
PLLs:	4
DLLs:	2
PCSs:	2
ALUs:	1
DPHYs:	0
PIO Cells:	299
PIO Pins:	159

Template Info:
Hello World Project. Components included:
a) Processor - RISC-V MC w/ PIC/TIMER
b) GPIO
c) ASRAM - Asynchronous SRAM
d) UART - Serial port

Figure 2.18. Propel Project Configuration

3. Click **Next** to open the Project Information window as shown in Figure 2.19.



Create System Design

Project Information
Generate new project with the following specifications.

Project:

Project Name: project1
Project Location: C:/Users/Muneet/project1
Language: Verilog
Device: LFCPNX-100-7ASG256C
Speed: 7_High-Performance_1.0V
Template Name: Hello World Project
Borad Name: NA
Processor Name: RISC-V MC

Figure 2.19. Project Information

4. Click **Finish**. The MDIO Leader IP appears in the IP Catalog as shown in Figure 2.20.

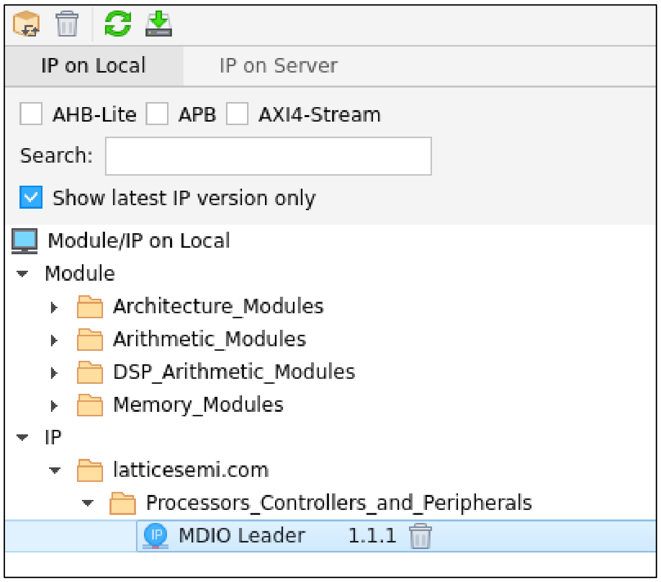
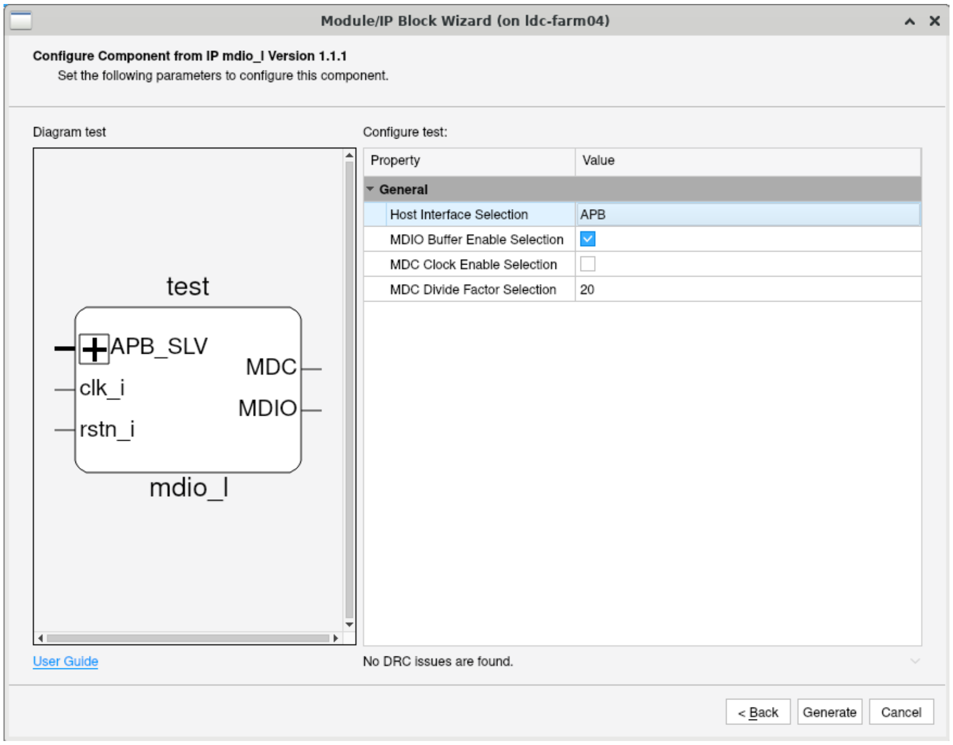


Figure 2.20. MDIO Leader in IP Catalog

5. Drag the MDIO Leader IP and instantiate it in the schematic view. Provide the settings as shown in Figure 2.21.



Module/IP Block Wizard (on ldc-farm04)

Configure Component from IP mdio_l Version 1.1.1

Set the following parameters to configure this component.

Diagram test

Configure test:

Property	Value
General	
Host Interface Selection	APB
MDIO Buffer Enable Selection	☑
MDC Clock Enable Selection	☐
MDC Divide Factor Selection	20

User Guide

No DRC issues are found.

Figure 2.21. MDIO Leader Configurability

In this example, the design details are mentioned for the AHBL interface. Upgrade the AHB-L Interconnect (Leader instance) instantiated with name ahbl0_inst to include three subordinates.

6. Add custom ports such as MDC or MDIO for MDIO interface, and then connect to the **MDIO Leader IP** instance.

   Figure 2.22 shows the schematic view of the sample design.

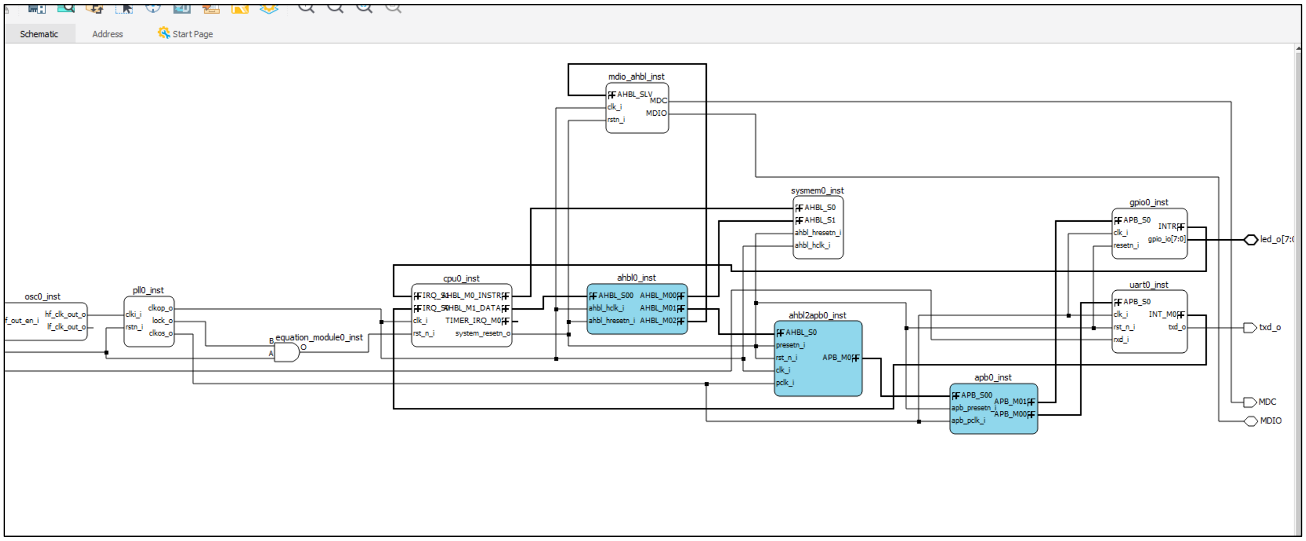
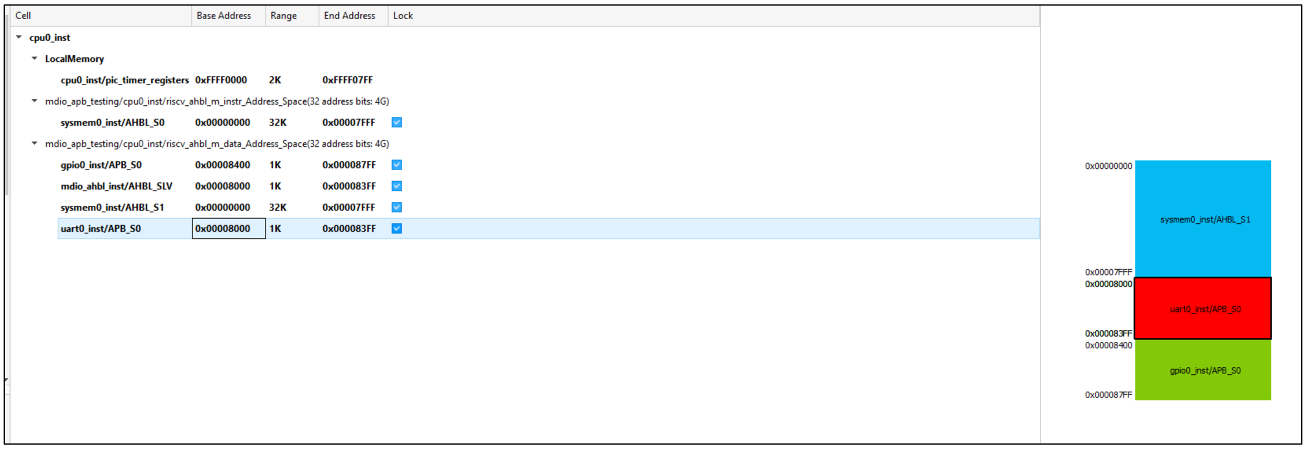


Figure 2.22. Schematic View for Example Design

2.10.2. Accessing the Address Tab and Register Memory Space View

To access the Address tab and Register Memory Space View, follow these steps:

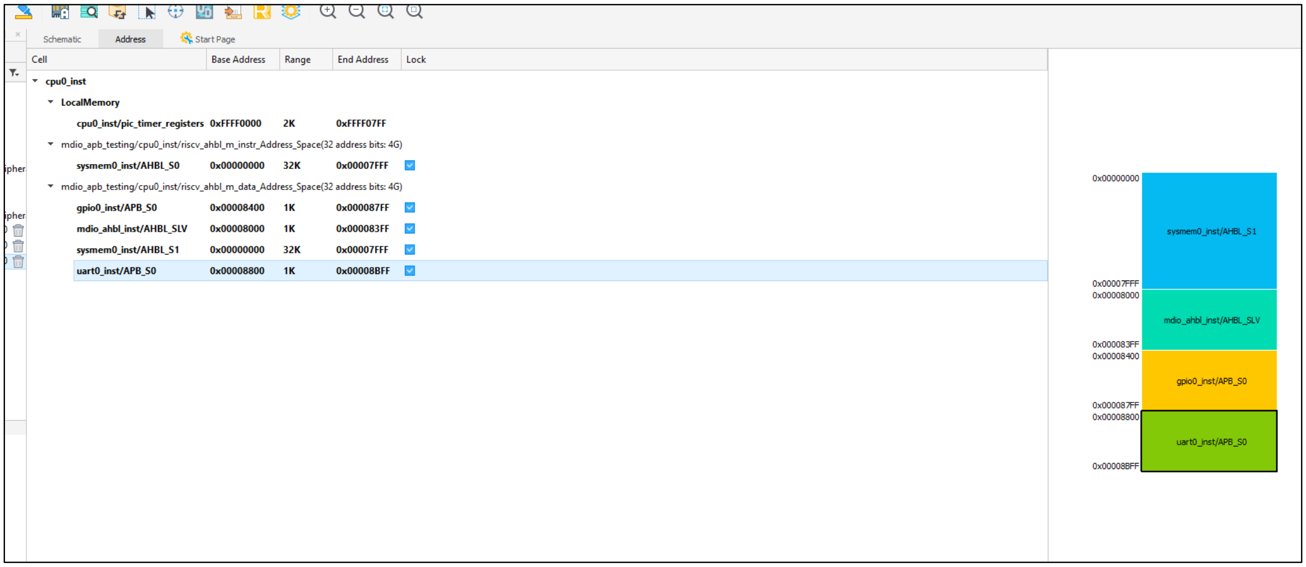
1. Make sure there are no conflicts in the addresses. Conflicts are shown in red color as shown in Figure 2.23.



Cell	Base Address	Range	End Address	Lock
cpu0_inst				
LocalMemory				
cpu0_inst/pic_timer_registers	0xFFFF0000	2K	0xFFFF07FF	
mdio_apb_testing/cpu0_inst/riscv_ahbl_m_instr_Address_Space(32 address bits: 4G)				
sysmem0_inst/AHBL_S0	0x00000000	32K	0x00007FFF	☑
mdio_apb_testing/cpu0_inst/riscv_ahbl_m_data_Address_Space(32 address bits: 4G)				
gpio0_inst/APB_S0	0x00008400	1K	0x000087FF	☑
mdio_ahbl_inst/AHBL_SLV	0x00008000	1K	0x000083FF	☑
sysmem0_inst/AHBL_S1	0x00000000	32K	0x00007FFF	☑
uart0_inst/APB_S0	0x00008000	1K	0x000083FF	☑

Figure 2.23. Conflicting Addresses in the Memory Range

Figure 2.24 shows the Register Space view after the conflict is resolved.



Cell	Base Address	Range	End Address	Lock
cpu0_inst				
LocalMemory				
cpu0_inst/pic_timer_registers	0xFFFF0000	2K	0xFFFF07FF	
mdio_apb_testing/cpu0_inst/riscv_ahbl_m_instr_Address_Space(32 address bits: 4G)				
sysmem0_inst/AHBL_S0	0x00000000	32K	0x00007FFF	☑
mdio_apb_testing/cpu0_inst/riscv_ahbl_m_data_Address_Space(32 address bits: 4G)				
gpio0_inst/APB_S0	0x00008400	1K	0x000087FF	☑
mdio_ahbl_inst/AHBL_SLV	0x00008000	1K	0x000083FF	☑
sysmem0_inst/AHBL_S1	0x00000000	32K	0x00007FFF	☑
uart0_inst/APB_S0	0x00008800	1K	0x00008BFF	☑

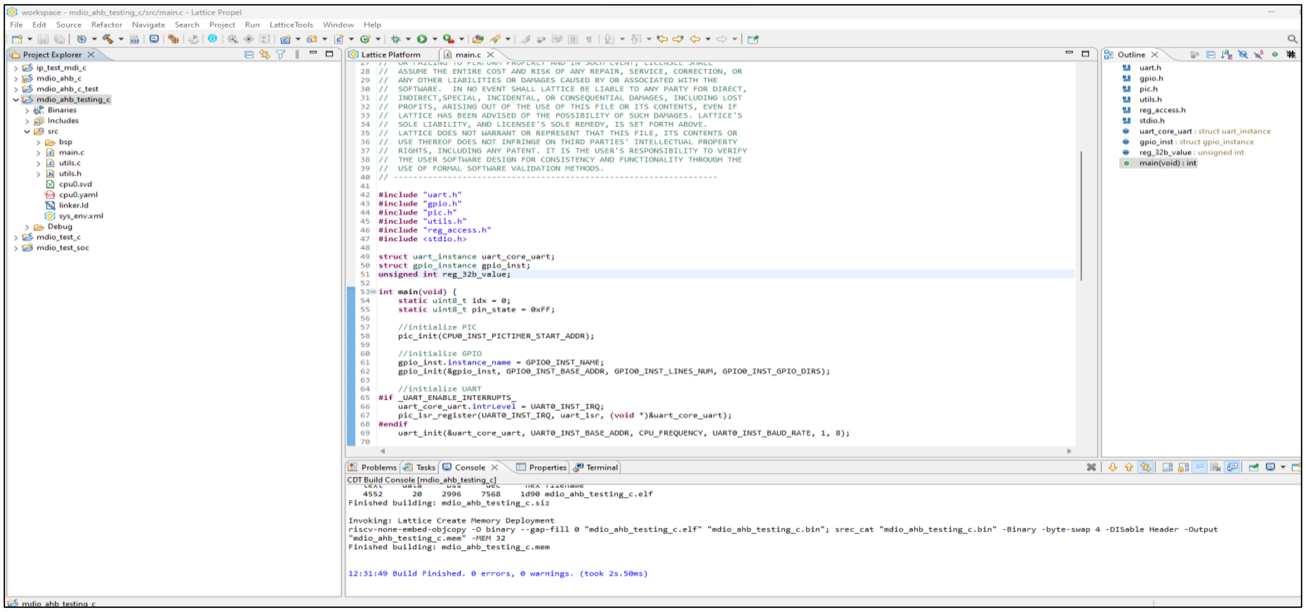
Figure 2.24. Register Space Memory View for the Device

2. After making the necessary changes, build the design.

2.10.3. Building Propel C-Based Project

To build the Propel C-based project, follow these steps:

1. Run the Lattice Propel Builder to open the Lattice Propel IDE for C-based project generation.
2. After the Propel IDE opens, enter the project name and create the project.
3. After the project is created, modify the *main.c* file, as required, to do the Register Read/Write that uses the *reg_access.h* header file.
4. Select **Project > Build Project** to build the C-based project.



```
//  OR FAILING TO PERFORM PROPERLY AND IN SUCH EVENT, LICENSEE SHALL
//  ASSUME THE ENTIRE COST AND RISK OF ANY REPAIR, SERVICE, CORRECTION, OR
//  ANY OTHER LIABILITIES OR DAMAGES CAUSED BY OR ASSOCIATED WITH THE
//  SOFTWARE.  IN NO EVENT SHALL LATTICE BE LIABLE TO ANY PARTY FOR DIRECT,
//  INDIRECT,SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST
//  PROFITS, ARISING OUT OF THE USE OF THIS FILE OR ITS CONTENTS, EVEN IF
//  LATTICE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. LATTICE'S
//  SOLE LIABILITY, AND LICENSEE'S SOLE REMEDY, IS SET FORTH ABOVE.
//  LATTICE DOES NOT WARRANT OR REPRESENT THAT THIS FILE, ITS CONTENTS OR
//  USE THEREOF DOES NOT INFRINGE ON THIRD PARTIES' INTELLECTUAL PROPERTY
//  RIGHTS, INCLUDING ANY PATENT. IT IS THE USER'S RESPONSIBILITY TO VERIFY
//  THE USER SOFTWARE DESIGN FOR CONSISTENCY AND FUNCTIONALITY THROUGH THE
//  USE OF FORMAL SOFTWARE VALIDATION METHODS.
// ------------------------------------------------------------------

#include "uart.h"
#include "gpio.h"
#include "pic.h"
#include "utils.h"
#include "reg_access.h"
#include <stdio.h>

struct uart_instance uart_core_uart;
struct gpio_instance gpio_inst;
unsigned int reg_32b_value;

int main(void) {
    static uint8_t idx = 0;
    static uint8_t pin_state = 0xFF;

    //initialize PIC
    pic_init(CPU0_INST_PICTIMER_START_ADDR);

    //initialize GPIO
    gpio_inst.instance_name = GPIO0_INST_NAME;
    gpio_init(&gpio_inst, GPIO0_INST_BASE_ADDR, GPIO0_INST_LINES_NUM, GPIO0_INST_GPIO_DIRS);

    //initialize UART
#if _UART_ENABLE_INTERRUPTS_
    uart_core_uart.intrLevel = UART0_INST_IRQ;
    pic_isr_register(UART0_INST_IRQ, uart_isr, (void *)&uart_core_uart);
#endif
    uart_init(&uart_core_uart, UART0_INST_BASE_ADDR, CPU_FREQUENCY, UART0_INST_BAUD_RATE, 1, 8);
```

CDT Build Console [mdio_ahb_testing_c]

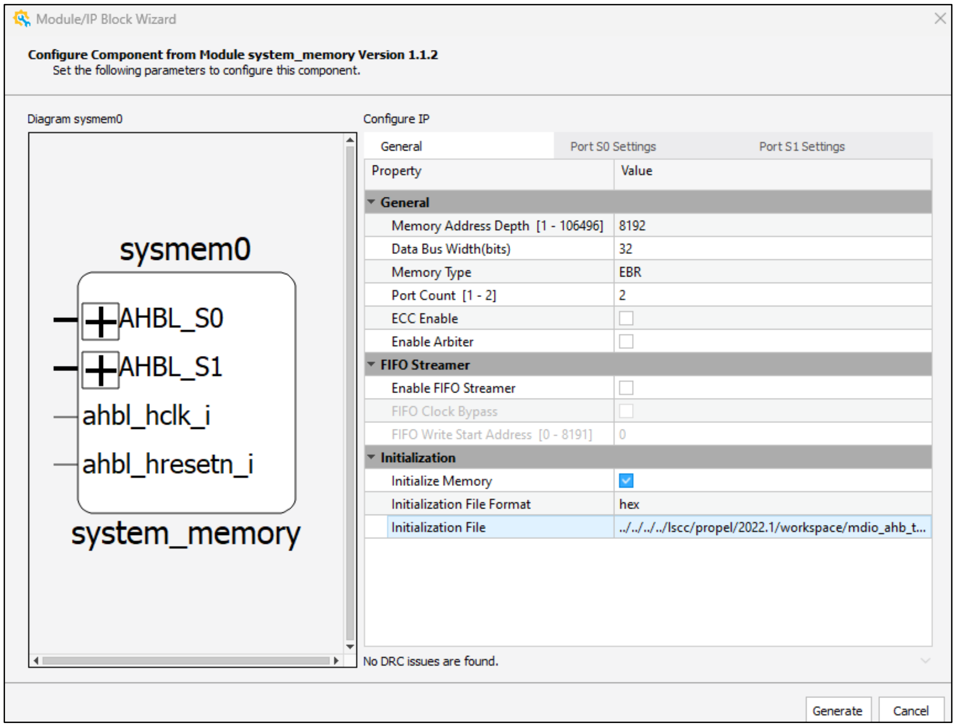
```
   text    data     bss     dec     hex filename
   4552      20    2996    7568    1d90 mdio_ahb_testing_c.elf
Finished building: mdio_ahb_testing_c.siz

Invoking: Lattice Create Memory Deployment
riscv-none-embed-objcopy -O binary --gap-fill 0 "mdio_ahb_testing_c.elf" "mdio_ahb_testing_c.bin"; srec_cat "mdio_ahb_testing_c.bin" -Binary -byte-swap 4 -DISable Header -Output
"mdio_ahb_testing_c.mem" -MEM 32
Finished building: mdio_ahb_testing_c.mem


12:31:49 Build Finished. 0 errors, 0 warnings. (took 2s.50ms)
```

Figure 2.25. C-Based Project Builder View

2.10.4. Integrating C-Based Project Files to Schematic View

To integrate C-based project files to schematic view, follow these steps:

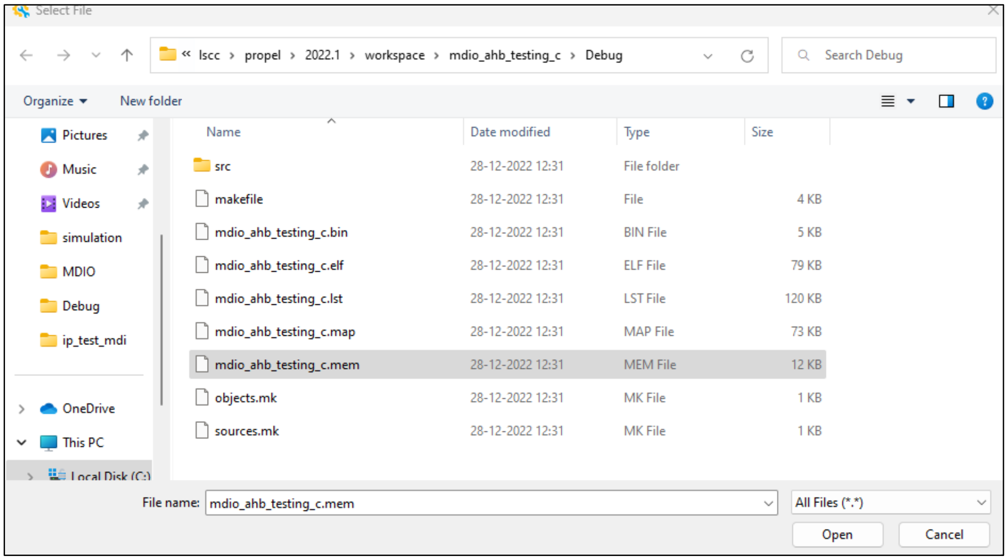
1. Modify the *System Memory* module instantiation in the schematic view by initializing it using the memory file generated in the debug folder of the C-based project as shown below.



Configure Component from Module system_memory Version 1.1.2
Set the following parameters to configure this component.

Property	Value
General	
Memory Address Depth [1 - 106496]	8192
Data Bus Width(bits)	32
Memory Type	EBR
Port Count [1 - 2]	2
ECC Enable	☐
Enable Arbiter	☐
FIFO Streamer	
Enable FIFO Streamer	☐
FIFO Clock Bypass	☐
FIFO Write Start Address [0 - 8191]	0
Initialization	
Initialize Memory	☑
Initialization File Format	hex
Initialization File	../../../../lscc/propel/2022.1/workspace/mdio_ahb_t...

No DRC issues are found.



Name	Date modified	Type	Size
src	28-12-2022 12:31	File folder	
makefile	28-12-2022 12:31	File	4 KB
mdio_ahb_testing_c.bin	28-12-2022 12:31	BIN File	5 KB
mdio_ahb_testing_c.elf	28-12-2022 12:31	ELF File	79 KB
mdio_ahb_testing_c.lst	28-12-2022 12:31	LST File	120 KB
mdio_ahb_testing_c.map	28-12-2022 12:31	MAP File	73 KB
mdio_ahb_testing_c.mem	28-12-2022 12:31	MEM File	12 KB
objects.mk	28-12-2022 12:31	MK File	1 KB
sources.mk	28-12-2022 12:31	MK File	1 KB

File name: mdio_ahb_testing_c.mem

All Files (*.*)

Figure 2.26. System Memory Modification

2. After modifying and regenerating the system memory, select **Project > Build Project** to re-validate and re-build the schematic view.

2.10.5. Verifying SoC and Design View

To verify SoC and Design View, follow these steps:

1. Click the **VD** button to switch to *Verification and SoC Design,* as shown in the figure below.

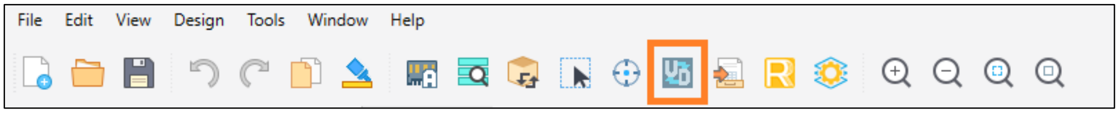


Figure 2.27. Propel Builder Icons List

2. The view is changed as shown in Figure 2.28.

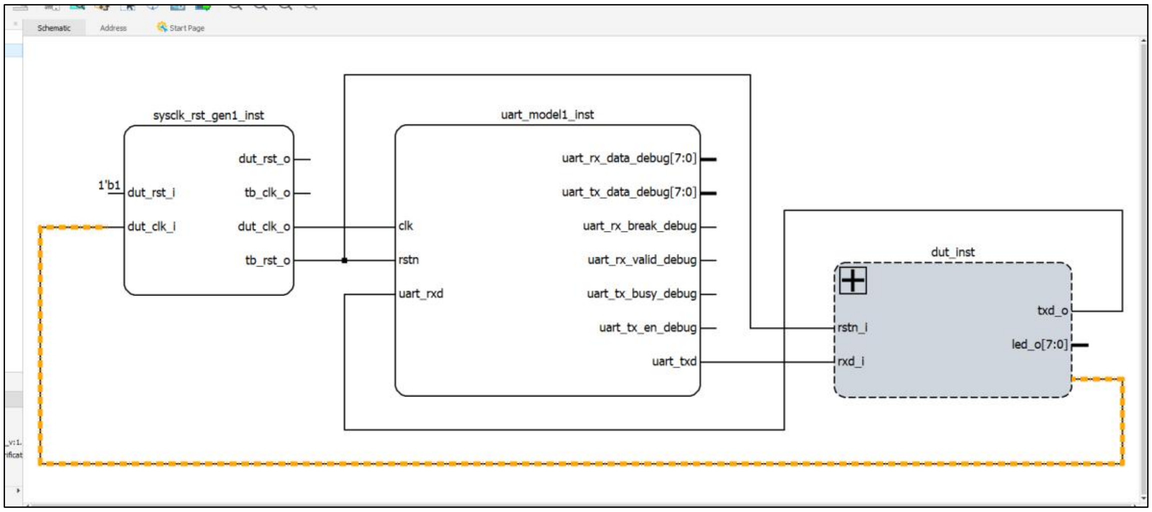
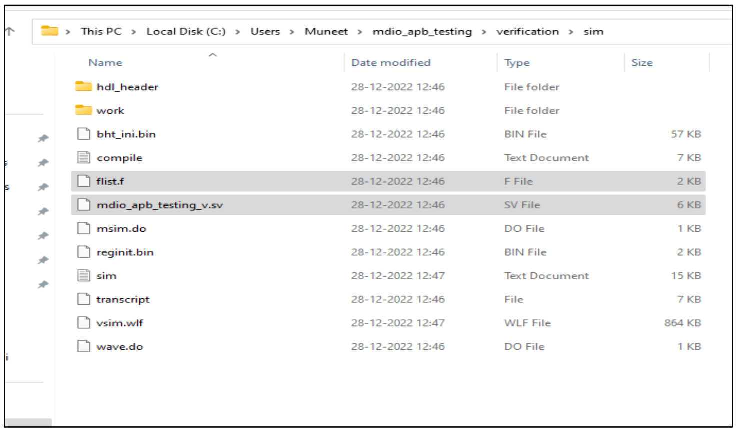


Figure 2.28. SoC Verification and Design View

3. Perform validation and re-build the design again.

4. Click **Launch Simulation** to launch the simulation in default view and to generate all related library and script files. This is an important step for running the simulation.
5. If the modifications in testbench are required, go to the *project_location/verification/sim* and modify the *.f* and *.sv* files as shown in Figure 2.29.



This PC > Local Disk (C:) > Users > Muneet > mdio_apb_testing > verification > sim

Name	Date modified	Type	Size
hdl_header	28-12-2022 12:46	File folder	
work	28-12-2022 12:46	File folder	
bht_ini.bin	28-12-2022 12:46	BIN File	57 KB
compile	28-12-2022 12:46	Text Document	7 KB
flist.f	28-12-2022 12:46	F File	2 KB
mdio_apb_testing_v.sv	28-12-2022 12:46	SV File	6 KB
msim.do	28-12-2022 12:46	DO File	1 KB
reginit.bin	28-12-2022 12:46	BIN File	2 KB
sim	28-12-2022 12:47	Text Document	15 KB
transcript	28-12-2022 12:46	File	7 KB
vsim.wlf	28-12-2022 12:47	WLF File	864 KB
wave.do	28-12-2022 12:46	DO File	1 KB

Figure 2.29. Verification Project Location

6. After modifications, run the simulation directly from SoC Verification and Design View. Do not re-validate or re-build the project because it reverts to default view only.

Appendix A. Resource Utilization

Table A.1 shows the resource utilization of the MDIO Leader IP.

Table A.1. Resource Utilization

Interface	PFU Registers	LUT-4
AHB-Lite	76	149
APB	79	150
AXI-Lite	145	194

References

- MDIO Leader IP Release Notes (FPGA-RN-02029)
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- Lattice Propel SDK 2.1 User Guide.
- AMBA AHB-L Specification.
- AMBA APB Specification.
- AMBA AXI Specification.
- IEEE802.3aeMDC/MDIO.
- Avant-E web page
- Avant-G web page
- Avant-X web page
- CertusPro-NX web page
- Certus-N2 web page
- MachXO5-NX web page
- Lattice Propel Design Environment web page
- Lattice Radiant Software web page
- Lattice Solutions IP Cores web page
- Lattice Insights web page 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 1.3, IP v1.3.0, December 2025

Section	Change Summary
All	Updated the IP version information on the cover page.
Introduction	Updated Table 1.1. Summary of the MDIO Leader IP.
Functional Description	• Updated Table 2.8. Generated File List. • Updated the Constraining the IP section.

Revision 1.2, IP v1.2.0, July 2025

Section	Change Summary
All	• Updated the IP version information on the cover page. • Made editorial and formatting fixes.
Abbreviations in This Document	• Updated the title of this section. • Added the following abbreviations: • MAC • MDC • MIIM • PCS
Introduction	• Updated Table 1.1. Summary of the MDIO Leader IP. • Added the Licensing Information section.
Functional Description	• Updated the Overview section. • Updated Table 2.1. MDIO Leader IP Core Signal Description. • Updated Table 2.2. Attributes Table. • Updated Figure 2.2. Lattice Radiant Software IP Wizard Reference. • Updated Table 2.7. Register Details for MDIO Clocking Control Register (0x000C). • Updated Figure 2.11. Configure User Interface of MDIO Leader IP. • Updated Figure 2.21. MDIO Leader Configurability.
References	Added the following references: • MDIO Leader IP Release Notes (FPGA-RN-02029) • Certus-N2 web page • MachXO5-NX web page

Revision 1.1, IP v1.1.1, December 2024

Section	Change Summary
All	• Renamed the document title from *MDIO Leader IP Core - Lattice Radiant and Lattice Propel* to *MDIO Leader IP*. • Added the IP version information on the cover page. • Made editorial fixes.
Disclaimers	Updated boilerplate.
Acronyms in This Document	Added *Advanced Extensible Interface* (*AXI*).
Introduction	Added the Quick Facts section.
Functional Description	• In Table 2.1. MDIO LEADER IP Core Signal Description: • Updated the instances of *slave* to *completer* for *APB Interface Signals* • Updated the *axi_bresp_o [1:0]* and *axi_rresp_o [1:0]* port descriptions.

Section	Change Summary
	- Updated instances of *black box* to *closed-box* in the Generating the IP section. - Removed *Hardware* from the *IP Validation* section title and updated its content. - Updated an instance of *slave* to *subordinate* for *AHB Lite Interconnect Configuration* in the Generating the Project and Basic Instantiation section. - Updated the following figures: - Figure 2.2. Lattice Radiant IP Wizard Reference - Figure 2.10. Module/IP Block Wizard - Figure 2.11. Configure User Interface of MDIO Leader IP - Figure 2.12. Check Generating Results - Figure 2.20. MDIO Leader in IP Catalog - Figure 2.21. MDIO Leader ConfigurabilityFigure 2.2. Lattice Radiant Software IP Wizard Reference - Removed *Figure 2.22. AHB Lite Interconnect Configuration* and updated the numbers of remaining figures accordingly.
Resource Utilization	Updated this section.
References	- Added *Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)*. - Added the *Avant-E*, *Avant-G*, *Avant-X*, *CertusPro-NX*, *Certus-N2*, *Lattice Propel Design Environment*, *Lattice Radiant Software*, *Lattice Solutions IP Cores*, and *Lattice Insights* web pages.

Revision 1.0, April 2023

Section	Change Summary
All	Initial release.

LATTICE SEMICONDUCTOR®

www.latticesemi.com