I2C Target IP

IP Version: v2.6.0

User Guide

FPGA-IPUG-02072-2.3

June 2026

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
AHB	Advanced High-Performance Bus
APB	Advanced Peripheral Bus
AMBA	Advanced Microcontroller Bus Architecture
CSR	Control and Status Register
DUT	Design Under Test
EBR	Embedded Block RAM
FIFO	First In, First Out
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input/Output
I/O	Input/Output
I2C	Inter-Integrated Circuit
IP	Intellectual Property
LMMI	Lattice Memory Mapped Interface
LSE	Lattice Synthesis Engine
LUT	Look-Up Table
PDC	Post-Synthesis Design Constraints
PIC	Programmable Interrupt Controller
PLL	Phase-Locked Loop
RAM	Random Access Memory
RISC-V	Reduced Instruction Set Computer Five
RTL	Register Transfer Level
RX	Receiver
SCL	Serial Clock
SDA	Serial Data
SoC	System on Chip
SRAM	Static Random Access Memory
TX	Transmitter
UART	Universal Asynchronous Receiver/Transmitter

1. Introduction

1.1. Overview of the IP

I2C (Inter-Integrated Circuit) bus is a simple, low-bandwidth, short-distance protocol. It is often seen in systems with peripheral devices that are accessed intermittently. It is commonly used in short-distance systems, where the number of traces on the board should be minimized. The device that initiates the transmission on the I2C bus is commonly known as the Controller, while the device being addressed is called the Target.

Lattice Semiconductor general-purpose I2C Target IP provides device addressing, read/write operation and an acknowledgement mechanism. The programmable nature of FPGA provides you with the flexibility of configuring the I2C Target device to any legal Target address, thus, avoiding a potential Target address collision on an I2C bus with multiple Target devices.

1.2. Quick Facts

Table 1.1. Summary of the I2C Target IP

IP Requirements	Supported Devices	CrossLink™-NX, Certus™-NX, Certus-NX-RT, CertusPro™-NX, CertusPro-NX-RT, MachXO5™-NX, Lattice Avant™, Certus-N2, and MachXO4™.
	IP Changes[1]	For a list of changes to the IP, refer to the I2C Target IP Release Notes (FPGA-RN-02028).
Resource Utilization	Supported User Interfaces	Lattice Memory Mapped Interface (LMMI) and Advanced Peripheral Bus (APB)
	Resources	Refer to Appendix A. Resource Utilization
Design Tool Support	Lattice Implementation	IP Core v2.6.0 – Lattice Radiant™ Software 2026.1 and Lattice Propel™ Builder 2026.1
	Synthesis	Lattice Synthesis Engine
		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. I2C Target IP Support Readiness

Device Family	System Clock Frequency (MHz)[1]	Supported Feature	Radiant Timing Model
CrossLink-NX Certus-NX Certus-NX-RT CertusPro-NX CertusPro-NX-RT MachXO5-NX MachXO4	40–100	SCL at 100 kHz – 1MHz: I2C Read-Write	Final
Avant Certus-N2	40–100	SCL at 100 kHz – 1MHz: I2C Read-Write	Preliminary

Note:

1. This is the system clock frequency used during hardware validation. For the actual frequency supported by the IP, refer to Appendix A. Resource Utilization.

The example design for the I2C Target IP allows for simulation and deployment to development boards for testing. Refer to the Example Design section for more details on how to run the example design on hardware and simulation.

1.4. Features

The I2C Target IP supports the following key features:

- Supports 7-bit and 10-bit addressing modes
- Supports the following bus speeds:
  - Standard-mode (Sm) – up to 100 kbit/s
  - Fast-mode (Fm) – up to 400 kbit/s
  - Fast-mode Plus (Fm+) – up to 1 Mbit/s
- Supports clock stretching
- Programmable ACK/NACK response on address and data phases
- Integrated pull-up through FPGA I/O settings
- Integrated glitch filter
- Polling and out-of-band interrupt modes

1.5. Licensing and Ordering Information

The I2C Target IP is provided at no additional cost with the Lattice Radiant software.

1.6. Minimum Device Requirements

There is no limitation in device speed grade for the I2C Target IP. See Appendix A. Resource Utilization for minimum required resources to instantiate this IP and maximum clock frequency supported.

1.7. Naming Conventions

1.7.1. Nomenclature

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

1.7.2. 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

1.7.3. Attribute Names

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

2. Functional Description

This section provides a detailed functional description of the I2C Target IP which includes information regarding clock and reset handling, and user interfaces.

2.1. IP Architecture Overview

Functional diagram of the I2C Target is presented in Figure 2.1. The APB2LMMI bridge and APB interface is only available when *APB Mode Enable* attribute is Checked.

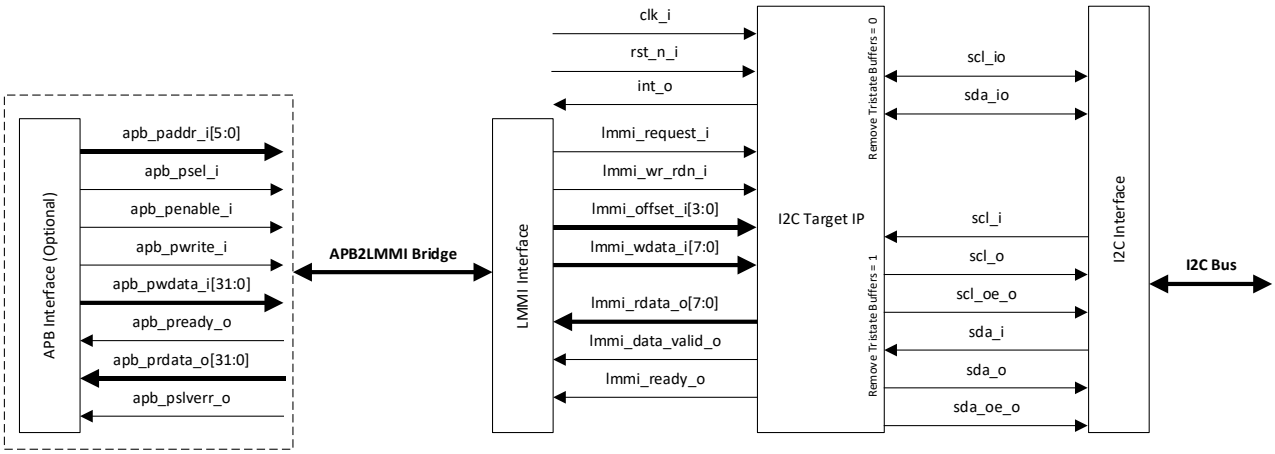

Figure 2.1. I2C Target IP Functional Diagram

2.2. Clocking

The I2C Target IP requires a system clock, *clk_i*. You must provide this clock from an external source and route it to an appropriate pin on the FPGA. The externally sourced system clock provides the timing reference and is used to operate the internal logic of the IP.

2.3. Reset

The I2C Target IP contains an asynchronous active low reset, *rst_n_i*. When asserted, the output ports and registers are forced to their default values. Reset assertion may be asynchronous; however, reset negation must be synchronous. The IP includes internal logic to synchronously de-assert the internal reset once *rst_n_i* is de-asserted, so you do not need to implement your own de-assertion logic.

2.4. User Interfaces

Table 2.1. User Interfaces and Supported Protocols

User Interface	Supported Protocols	Description
CSR Interface	LMMI, APB	The CSR interface is used to configure the IP core and access the FIFOs. For the LMMI interface, refer to the Lattice Memory Mapped Interface and Lattice Interrupt Interface (FPGA-UG-02039). Take note of the following information when checking LMMI timing diagram in the said document: • lmmi_ready_o is always asserted; thus, write and read transactions have no wait state. • Read latency is one clock cycle. For the APB interface, refer to the AMBA 3 APB Protocol v1.0 Specification for interface information and timing diagrams.

User Interface	Supported Protocols	Description
		Take note of the following information when checking APB timing diagram in the said document: • Write transaction has one wait state. • Read transaction has one wait state (IP Core v1.0.2 or earlier has two wait states).

2.5. Operations Details

2.5.1. General I2C Operation

In I2C bus, the transaction is always initiated by the controller. A target may not transmit data unless it has been addressed by the controller. Each device on the I2C bus has a specific device address to differentiate between other devices that are on the same I2C bus. Data transfer is initiated only when the bus is idle. A bus is considered idle if both SDA and SCL lines are high after a STOP condition.

The general procedure for an I2C transaction is as follows:

1. Controller wants to send data to a target:
   - Controller-transmitter sends a START condition and addresses the target-receiver
   - Controller-transmitter sends data to target-receiver
   - Controller-transmitter terminates the transfer with a STOP condition
2. Controller wants to receive/read data from a target:
   - Controller-receiver sends a START condition and addresses the target-transmitter
   - Controller-receiver receives data from the target-transmitter
   - Controller-receiver terminates the transfer with a STOP condition

I2C communication is initiated by the controller sending a START condition and terminated by the controller sending a STOP condition. Normal data on the SDA line must be stable during the high level of the SCL line. The High or Low state of the data line can only change when SCL is Low. The Start condition is a unique case and is defined by a High-to-Low transition on the SDA line while SCL is High. The Stop condition is a unique case and is defined by a Low-to-High transition on the SDA line while SCL is High. These are shown in Figure 2.2.

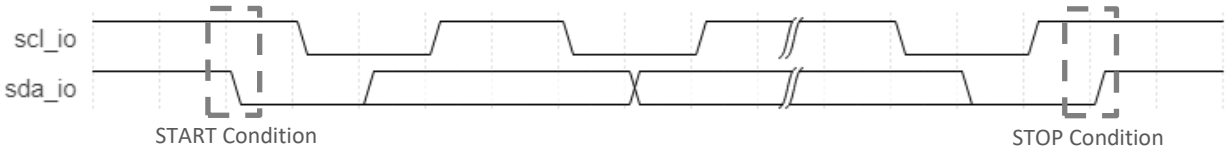

Figure 2.2. START and STOP Conditions

Each data packet on the I2C bus consists of eight bits of data followed by an acknowledge bit (ACK) so one complete data byte transfer requires nine clock pulses. Data is transferred with the most significant bit (MSB) first. The transmitter releases the SDA line during the ACK bit and the receiver of the data transfer must drive the SDA line low during the ACK bit to acknowledge the data receipt. If a Target-receiver does not drive the SDA line low during the ACK bit, this indicates that the Target-receiver was unable to accept the data and the Controller can then generate a Stop condition to abort the transfer. If the Controller-receiver does not generate an ACK, this indicates to the Target-transmitter that this byte was the last byte of the transfer.

For more information on I2C bus, refer to I2C Bus Specification and User Manual.

2.5.2. Glitch Filter

I2C Target IP has integrated glitch filter to remove 50 ns noise/spike as recommended by the I2C Bus Spec for Standard mode, Fast mode and Fast mode Plus. The glitch filter is applied to both the SCL and SDA signals before they are fed to internal logic. Thus, the I2C signals seen by the IP is delayed by a number of clock cycles (~50 ns +1 clock cycle). The filter depth is automatically adjusted based on the *System Clock Frequency* attribute.

2.5.3. Clock Stretching

Clock stretching allows an I2C target to pause a transaction by holding the SCL line low. The controller cannot proceed until the target releases SCL high again. Although a target may receive data bytes quickly, it may require additional time to store a received byte or prepare the next byte for transmission. To accommodate this, the target can pull SCL low to delay the controller until it is ready for the next byte.

The I2C Target IP performs clock stretching at the byte level (during the ACK/NACK bit) when CONTROL_REG.clk_stretch_en is set to 1 and any of the following interrupt status bits are asserted:

- tx_fifo_empty_int (bit 3 of INT_STATUS1)
- rx_fifo_full_int (bit 2 of INT_STATUS1)
- rx_addr_int (bit 3 of INT_STATUS2)

The I2C Target IP releases the SCL line when all of the above interrupt status bits are cleared or when CONTROL_REG.clk_stretch_en is set to 0. This behavior provides firmware with a larger time window to prepare for a new message, fill an empty Transmit FIFO, or empty a full Receive FIFO.

2.5.4. ACK/NACK Response

The I2C Target IP can be configured to send an ACK or a NACK based on settings of CONTROL_REG.nack_data and CONTROL_REG.nack_addr. Refer to the Control Register (CONTROL_REG) section for details. If the host needs to temporarily disable access to the I2C Target IP, it should set CONTROL_REG.nack_addr to 1'b1. In this case, the I2C Target IP sends a NACK when it is addressed by the external I2C controller.

If the host needs to terminate an ongoing I2C write transaction to the I2C Target IP, it should set CONTROL_REG.nack_data to 1'b1. In this case, the I2C Target IP sends a NACK on the next ACK bit for a data byte. Note that the ACK bit is always sent by the receiver, the CONTROL_REG.nack_data has no effect on I2C read transactions.

When updating the ACK/NACK response, the host must ensure that the change meets I2C timing requirements. Register updates made too close to the ACK/NACK sampling edge on SCL may result in I2C timing violations.

2.6. Programming Flow

2.6.1. Initialization

To perform initialization, load the appropriate registers of the I2C Target namely:

- TARGET_ADDRL_REG, TARGET_ADDRH_REG – This step is optional. In most cases, initial value set in I2C Target Addresses attribute of the user interface does not need to be changed.
- CONTROL_REG – It is recommended to set this if you want I2C Target to NACK during the Data or Address phase, or if you want to enable/disable clock stretching and 10-bit addressing.
- TGT_BYTE_CNT_REG – It is recommended to set this if the size of the data is known. Set this to 8'h00 if the number of bytes to transfer is not known, that is receiving unknown amount of data.
- INT_ENABLE1_REG – It is recommended to enable only the following interrupts when receiving commands from controller.
  - Transfer Complete Interrupt – If the size of data is known
  - Receive FIFO Data Interrupt – If the size of data is unknown
- INT_ENABLE2_REG – It is recommended to enable both error interrupts.

2.6.2. Data Transfer in response to I2C Controller Read

Below are the recommended steps to perform data transfer in response to read request of I2C Controller. This assumes that the amount of data to send is known.

To perform data transfer in response to read request of I2C Controller:

1. Write data to WR_DATA_REG, amounting to ≤ FIFO Depth.
2. Enable only Transfer Complete Interrupt. If transmit data is > FIFO Depth, enable also Transmit FIFO Almost Empty interrupt. If no more data to transfer, otherwise, proceed to step 7.

3. Wait for Transmit FIFO Almost Empty Interrupt.
   If polling mode is desired (interrupts are disabled), read INT_STATUS1_REG until tx_fifo_aempty_int asserts.
   If interrupt mode is desired, simply wait for interrupt signal to assert, then read INT_STATUS1_REG and check that tx_fifo_aempty_int is asserted.
   Read INT_STATUS2_REG also to check that no error occurred.
4. Clear Transmit FIFO Almost Empty Interrupt, it also okay to clear all interrupts.
5. Write data byte to WR_DATA_REG, amounting to less than or equal to (FIFO Depth - TX FIFO Almost Empty Setting).
6. If there are remaining data to transfer, go back to Step 3, otherwise, disable Transmit FIFO Almost Empty Interrupt.
7. Wait for Transfer Complete Interrupt
   If polling mode is desired (interrupts are disabled), read INT_STATUS1_REG until tr_cmp_int asserts.
   If interrupt mode is desired, simply wait for interrupt signal to assert, then read INT_STATUS1_REG and check that tr_cmp_int is asserted.
   Read INT_STATUS2_REG also to check that no error occurred.
8. Clear all interrupts.

2.6.3. Data Transfer in response to I2C Controller Write

Below are the recommended steps to perform data transfer in response to write request of I2C Controller. This assumes that the amount of data to receive is known.

To perform data transfer in response to write request of I2C Controller:

1. Enable only Transfer Complete Interrupt. If data to receive is > FIFO Depth, enable also Receive FIFO Almost Full interrupt. If data to receive is <= FIFO Depth, proceed to Step 6.
2. Wait for Receive FIFO Almost Full Interrupt.
   If polling mode is desired (interrupts are disabled), read INT_STATUS2_REG until rx_fifo_afull_int asserts.
   If interrupt mode is desired, simply wait for interrupt signal to assert, then read INT_STATUS2_REG and check that rx_fifo_afull_int is asserted.
   Read INT_STATUS2_REG also to check that no error occurred.
3. Clear Receive FIFO Almost Full Interrupt, it also okay to clear all interrupts.
4. Read data byte from RD_DATA_REG, amounting to less than or equal to (RX FIFO Almost Full Setting).
5. If there are remaining data to receive, go back to Step 2, otherwise, disable Receive FIFO Almost Full Interrupt.
6. Wait for Transfer Complete Interrupt
   If polling mode is desired (interrupts are disabled), read INT_STATUS1_REG until tr_cmp_int asserts.
   If interrupt mode is desired, simply wait for interrupt signal to assert, then read INT_STATUS1_REG and check that tr_cmp_int is asserted.
   Read INT_STATUS2_REG also to check that no error occurred.
7. Clear all interrupts.
8. Read all data from RD_DATA_REG.

2.6.4. Data Transfer in response to I2C Combined Format Read

Below are the recommended steps to perform data transfer in response to a read request of I2C Controller using a combined format.

To perform data transfer in response to read request of I2C Controller:

1. Enable Transmit FIFO Empty, Receive FIFO Full, Receive Address Interrupt, and Stop Detect Interrupt.
2. Enable Clock Stretch in the CONTROL_REG.
3. Wait for the interrupt to assert.
4. Check the cause of the interrupt.
   a. If the interrupt is due to Receive Address Interrupt, check RX_ADDR_1_REG.rx_rnw to check if the request is a write or a read. For a combined format, first transaction is a write.
5. Clear all interrupts and wait for the interrupt to assert again.

6. Check the cause of the interrupt.
    a. If the interrupt is due to Receive Address Interrupt, check RX_ADDR_1_REG.rx_rnw to confirm the next transaction. In a combined format, the second transaction is a read. Read all data from Receive FIFO and provide appropriate response in the Transmit FIFO. Go back to step 5.
    b. If the interrupt is due to Receive FIFO Full, the Receive FIFO should be emptied to allow more data to be received. Go back to step 5.
    c. If the interrupt is due to Transmit FIFO Empty, provide necessary data by writing to the Transmit FIFO. Go back to step 5.
    d. If the interrupt is due to Stop Detect Interrupt, proceed to the remaining steps.
7. Check if there are remaining data in the Receive FIFO and Transmit FIFO.
8. Clear the FIFOs after consuming the necessary data by writing to CONTROL_REG.rx_fifo_reset and CONTROL_REG.tx_fifo_reset.
9. Clear all interrupts.

3. IP Parameter Description

The configurable attributes of the I2C Target IP are shown in Table 3.1. The attributes can be configured through the IP Catalog's Module/IP wizard of the Lattice Radiant software.

Wherever applicable, default values are in **bold**.

Table 3.1. Attributes Table

Attribute	Selectable Values	Description
General		
APB Mode Enable	**Checked**, Unchecked	Selects memory-mapped interface for register access by the host. The unselected interfaces are not available in the generated IP. Checked: APB I/F is enabled. Unchecked: LMMI I/F is enabled.
Remove Tristate Buffers	Checked, **Unchecked**	Removes the tristate buffer to allow you to include additional logic between the IP and the buffers. Checked: Tristate buffers will not be instantiated inside the IP. Unchecked: Tristate buffers will be instantiated inside the IP.
Addressing Mode	**7-bit**, 10-bit	Specifies the reset value of CONTROL_REG.addr_10bit_en. 7-bit: CONTROL_REG.addr_10bit_en = 1'b0 10-bit: CONTROL_REG.addr_10bit_en = 1'b1
I2C Target Address:0x	7-bit: 0–7F, **51** 10 bit: 0–3FF, **51**	Specifies the reset value of Target Address Register. Your input is converted into 10-bit address value. The lower 7 bits are assigned to TARGET_ADDRL_REG.target_addr_l_reg while the upper 3 bits are assigned to TARGET_ADDRH_REG.target_addr_h_reg.
STOP Condition Detected Interrupt Assertion	All STOP condition detected on bus, **STOP Condition addressed to IP**	Specifies whether STOP Detect Interrupt will assert for all STOP condition detected on bus or only for STOP condition after I2C transaction addressed to IP. Refer to the Interrupt Status Registers (INT_STATUS1_REG, INT_STATUS2_REG) section for more details.
System Clock Frequency (MHz)	40–100, **50**	Specifies the frequency of system clock. This is used to configure the internal glitch filter.
Timing Adjustment		
I2C SDA Register Depth	**0**, 1, 2	Allows you to delay the SDA timing by 0 to 2 clock cycles. This helps to compensate any delays on the SCL line as it is sampled by the IP.
FIFO		
FIFO Width	—	Specifies the bit width of each data word of the internal FIFO. This is not editable and is fixed to 8 because each I2C data is 8 bits.
FIFO Depth	**16**, 32, 64, 128, 256	Specifies the number of FIFO levels. Accepts only power of two values.
Implementation of FIFO	EBR, **LUT**	Selects the FPGA resource that are used to implement the FIFO: EBR or LUT.
TX FIFO Almost Empty Flag	1–256, **2**	Specifies the trigger level for asserting INT_STATUS1_REG.tx_fifo_aempty_int. Refer to the Interrupt Status Registers (INT_STATUS1_REG, INT_STATUS2_REG) section for more details.
RX FIFO Almost Full Flag	1–256, **14**	Specifies the trigger level for asserting INT_STATUS1_REG.rx_fifo_afull_int. Refer to the section Interrupt Status Registers (INT_STATUS1_REG, INT_STATUS2_REG) for more details.

4. Signal Description

Table 4.1. I2C Target IP Signal Description

Port Name	I/O	Width	Description
Clock and Reset			
clk_i	In	1	System clock
rst_n_i	In	1	Asynchronous active low reset The reset assertion can be asynchronous but reset negation should be synchronous. When asserted, output ports and registers are forced to their reset values.
Interrupt Port			
int_o	Out	1	Interrupt signal Reset value is 1'b0.
I2C Interface (Remove Tristate Buffers = 0)			
scl_io	In/Out	1	I2C Serial Clock Generated by the external I2C Controller to synchronize data transfers. Reset value is weak high (pull-up).
sda_io	In/Out	1	I2C data signal Reset value is weak high (pull-up).
I2C Interface (Remove Tristate Buffers = 1)			
scl_i	In	1	I2C serial clock input Input from tristate buffer.
scl_o	Out	1	I2C serial clock output Output to tristate buffer. This signal is fixed to 1'b0.
scl_oe_o	Out	1	I2C serial clock output enable Active low signal to enable output of a tristate buffer. Reset value is 1'b1.
sda_i	In	1	I2C data signal input Input from tristate buffer.
sda_o	Out	1	I2C data signal output Output to tristate buffer. This signal is fixed to 1'b0.
sda_oe_o	Out	1	I2C data signal output enable Active low signal to enable output of a tristate buffer. Reset value is 1'b1.
LMMI Interface[1]			
lmmi_request_i	In	1	Start transaction
lmmi_wr_rdn_i	In	1	Write = 1'b1, Read = 1'b0
lmmi_offset_i	In	4	Register offset, starting at offset 0
lmmi_wdata_i	In	8	Input data bus.
lmmi_rdata_o	Out	8	Output data bus Reset value is 0.
lmmi_rdata_valid_o	Out	1	Read transaction is complete and lmmi_rdata_o contains valid data. Reset value is 1'b0.
lmmi_ready_o	Out	1	IP is ready to receive a new transaction. This is always asserted (tied to 1'b1).
APB Interface[1]			
apb_psel_i	In	1	APB Select signal Indicates that the device is selected and a data transfer is required.
apb_paddr_i	In	6	APB Address signal.
apb_pwdata_i	In	32	APB Write data signal Bits [31:8] are not used.
apb_pwrite_i	In	1	APB Direction signal Write = 1, Read = 0

Port Name	I/O	Width	Description
apb_penable_i	In	1	APB Enable signal Indicates the second and subsequent cycles of an APB transfer.
apb_pready_o	Out	1	APB Ready signal Indicates transfer completion. IP uses this signal to extend an APB transfer. Reset value is 1'b0.
apb_pslverr_o	Out	1	APB Error signal Indicates a transfer failure. This signal is tied to 1'b0.
apb_prdata_o	Out	32	APB Read data signal Bits [31:8] are not used and are fixed to 0. Reset value is 0.

Note:

1. Only one of the two interfaces are available as selected by *APB Mode Enable* attribute.

5. Register Description

5.1. Overview

The I2C Target IP configuration registers are located at the addresses shown in Table 5.1. The offset of each register is dependent on *APB Mode Enable* attribute setting as follows:

- *APB Mode Enable* is Unchecked – the offset increments by 1.
- *APB Mode Enable* is Checked – the offset increments by 4 to allow easy interfacing with the Processor and System Buses. In this mode, each register is 32-bit wide wherein the upper bits [31:8] are reserved and the lower 8 bits [7:0] are described in the next section.

Table 5.1. Registers Address Map

Offset LMMI	Offset APB	Register Name	Access Type	Description
0x0	0x00	WR_DATA_REG	WO	Write Data Register
0x0	0x00	RD_DATA_REG	RO	Read Data Register
0x1	0x04	TARGET_ADDRL_REG	RW	Target Address Lower Register
0x2	0x08	TARGET_ADDRH_REG	RW	Target Address Higher Register
0x3	0x0C	CONTROL_REG	RW	Control Register
0x4	0x10	TGT_BYTE_CNT_REG	RW	Target Byte Count Register
0x5	0x14	INT_STATUS1_REG	RW1C	Interrupt Status First Register
0x6	0x18	INT_ENABLE1_REG	RW	Interrupt Enable First Register
0x7	0x1C	INT_SET1_REG	WO	Interrupt Set First Register
0x8	0x20	INT_STATUS2_REG	RW1C	Interrupt Status Second Register
0x9	0x24	INT_ENABLE2_REG	RW	Interrupt Enable Second Register
0xA	0x28	INT_SET2_REG	WO	Interrupt Set Second Register
0xB	0x2C	FIFO_STATUS_REG	RO	FIFO Status Register
0xC	0x30	RX_ADDR_1_REG	RO	Received Address First Register
0xD	0x34	RX_ADDR_2_REG	RO	Received Address Second Register
0xE – 0xF	0x38 – 0x3C	Reserved	RSVD	Reserved Write access is ignored and 0 is returned on read access.

The RD_DATA_REG and WR_DATA_REG share the same offset. Write access to this offset goes to WR_DATA_REG while read access goes to RD_DATA_REG.

The behavior of registers to write and read access is defined by its access type, which is defined in Table 5.2.

Table 5.2. Access Type Definition

Access Type	Behavior on Read Access	Behavior on Write Access
RO	Returns register value	Ignores write access
WO	Returns 0	Updates register value
RW	Returns register value	Updates register value
RW1C	Returns register value	Writing 1’b1 on register bit clears the bit to 1’b0. Writing 1’b0 on register bit is ignored.
RSVD	Returns 0	Ignores write access

5.2. Write Data Register (WR_DATA_REG)

Table 5.3 presents the Write Data Register. This is the interface to Transmit FIFO. Writing to WR_DATA_REG pushes a word to Transmit FIFO. When writing to WR_DATA_REG, the host should ensure that Transmit FIFO is not full. This can be done by reading FIFO_STATUS_REG. Data is popped from WR_DATA_REG during I2C read transaction. When reset is performed, the contents of Transmit FIFO are not reset but the FIFO control logic is reset. Thus, content is not guaranteed after reset.

Table 5.3. Write Data Register

Field	Name	Access	Width	Reset
[7:0]	tx_fifo	WO	8	not guaranteed

5.3. Read Data Register (RD_DATA_REG)

Table 5.4 presents the Read Data register. This is the interface to Receive FIFO. After a data is received from I2C bus during I2C write transaction, the received data is pushed to Receive FIFO. Reading from RD_DATA_REG pops a word from Receive FIFO. The host should ensure that Receive FIFO has data before reading RD_DATA_REG, data is not guaranteed when this register is read during Receive FIFO empty condition. On the other hand, if Receive FIFO is full but I2C Target continues to receive data, new data is lost. Read FIFO_STATUS_REG to determine the status of Receive FIFO. Similar to Transmit FIFO, the reset value of Receive FIFO is also not guaranteed after reset.

Table 5.4. Read Data Register

Field	Name	Access	Width	Reset
[7:0]	rx_fifo	RO	8	not guaranteed

5.4. Target Address Registers (TARGET_ADDRL_REG, TARGET_ADDRH_REG)

The Target Address Lower Register (TARGET_ADDRL_REG) shown in Table 5.5 is a 7-bit Target address. This is used for 7-bit and 10-bit addressing mode as follows:

- For 7-bit Addressing Mode, it is the Target address
- For 10-bit Addressing Mode, it is the lower 7 bits of the Target address

Table 5.5. Target Address Lower Register

Field	Name	Access	Width	Reset
[7]	reserved	RSVD	1	–
[6:0]	target_addr_l_reg	RW	7	I2C Target Address[6:0]

The Target Address Higher Register (TARGET_ADDRH_REG) shown in Table 5.6 is the upper 3 bits of 10-bit Target address. This is not used in 7-bit addressing mode. The reset values of TARGET_ADDRL_REG and TARGET_ADDRH_REG is set by the I2C Target Address attribute as shown in Table 5.5 and Table 5.6.

Table 5.6. Target Address Higher Register

Field	Name	Access	Width	Reset
[7:3]	reserved	RSVD	5	–
[2:0]	target_addr_h_reg	RW	3	I2C Target Address[9:7]

5.5. Control Register (CONTROL_REG)

Table 5.7 presents a summary of Control Register. Each bit of this register controls the behavior of I2C Target IP.

Table 5.7. Control Register

Field	Name	Access	Width	Reset
[7]	reserved	RSVD	1	–
[6]	rx_fifo_reset	WO	1	1'b0

Field	Name	Access	Width	Reset
[5]	tx_fifo_reset	WO	1	1'b0
[4]	nack_data	RW	1	1'b0
[3]	nack_addr	RW	1	1'b0
[2]	reset	RW	1	1'b0
[1]	clk_stretch_en	RW	1	1'b0
[0]	addr_10bit_en	RW	1	See Addressing Mode in Table 3.1

- rx_fifo_reset

  Resets the Receive FIFO logic, effectively flushing the contents. This is write-only bit because it has an auto clear feature; it is cleared to 1'b0 after 1 clock cycle.
  1'b0 – No action.
  1'b1 – Resets the Receive FIFO.
- tx_fifo_reset

  Resets the Transmit FIFO logic, effectively flushing the contents. This is write-only bit because it has an auto clear feature; it is cleared to 1'b0 after 1 clock cycle.
  1'b0 – No action.
  1'b1 – Resets the Transmit FIFO.
- nack_data

  NACK on Data Phase. Specifies ACK/NACK response on I2C data phase.
  1'b0 – Sends ACK to received data
  1'b1 – Sends NACK to received data
- nack_addr

  NACK on Address Phase. Specifies ACK/NACK response on I2C address phase.
  1'b0 – Sends ACK to received address if it matches the programmed target address
  1'b1 – Sends NACK to received address
- reset

  Reset. Resets I2C Target IP. The registers and LMMI interface are not affected by this reset.
  1'b0 – No action.
  1'b1 – Resets I2C Target IP.
- clk_stretch_en

  Clock Stretch Enable. Enables clock stretching on ACK bit.
  1'b0 – I2C Target IP releases SCL signal
  1'b1 – I2C Target IP pulls down SCL signal on the next ACK bit and keeps pulling-down until the host writes 1'b0 on this bit.
- addr_10bit_en

  10-bit Address Mode Enable. Enables the reception of 10-bit I2C address.
  1'b0 – I2C Target IP rejects the 10-bit I2C address, it sends NACK.
  1'b1 – I2C Target IP responds to 10-bit I2C address. If TARGET_ADDRH_REG.target_addr_h_reg is 3'h0, it also responds to 7-bit address.

5.6. Target Byte Count Register (TGT_BYTE_CNT_REG)

Table 5.8 presents the summary of Target Byte Count Register. The desired number of bytes to transfer (read/write) in I2C bus should be written to this register. This is used for Transfer Complete interrupt generation – asserts when the target byte count is achieved.

Table 5.8. Target Byte Count Register

Field	Name	Access	Width	Reset
[7:0]	byte_cnt	RW	8	8'h00

5.7. Interrupt Status Registers (INT_STATUS1_REG, INT_STATUS2_REG)

Table 5.9 and Table 5.10 presents Interrupt Status Register (INT_STATUS1_REG and INT_STATUS2_REG), they contain all the interrupts currently pending in the I2C Target IP. When an interrupt bit asserts, it remains asserted until it is cleared by the host by writing 1'b1 to the corresponding bit.

The interrupt status bits are independent of the interrupt enable bits; in other words, status bits may indicate pending interrupts, even though those interrupts are disabled in the Interrupt Enable Register, see the Interrupt Enable Registers (INT_ENABLE1_REG, INT_ENABLE2_REG) section for details. The logic which handles interrupts should mask (bitwise AND logic) the contents of INT_STATUS1_REG and INT_ENABLE1_REG registers as well as INT_STATUS2_REG and INT_ENABLE2_REG to determine the interrupts to service. The int_o interrupt signal is asserted whenever both an interrupt status bit and the corresponding interrupt enable bits are set.

Table 5.9. Interrupt Status First Register

Field	Name	Access	Width	Reset
[7]	tr_cmp_int	RW1C	1	1'b0
[6]	stop_det_int	RW1C	1	1'b0
[5]	tx_fifo_full_int	RW1C	1	1'b0
[4]	tx_fifo_aempty_int	RW1C	1	1'b0
[3]	tx_fifo_empty_int	RW1C	1	1'b0
[2]	rx_fifo_full_int	RW1C	1	1'b0
[1]	rx_fifo_afull_int	RW1C	1	1'b0
[0]	rx_fifo_ready_int	RW1C	1	1'b0

- tr_cmp_int

  Transfer Complete Interrupt Status. This interrupt status bit asserts when the number of bytes transferred in I2C interface is equal to TGT_BYTE_CNT_REG.byte_cnt.
  - 1'b0 – No interrupt
  - 1'b1 – Interrupt pending
- stop_det_int

  STOP Condition Detected Interrupt Status. This interrupt status bit asserts when STOP condition is detected after an ACK/NACK following any I2C address or I2C IP Target Address depending on the *STOP Condition Detected Interrupt Assertion* attribute.
  - 1'b0 – No interrupt
  - 1'b1 – Interrupt pending
- tx_fifo_full_int

  Transmit FIFO Full Interrupt Status. This interrupt status bit asserts when Transmit FIFO changes from not full state to full state.
  - 1'b0 – No interrupt
  - 1'b1 – Interrupt pending
- tx_fifo_aempty_int

  Transmit FIFO Almost Empty Interrupt Status. This interrupt status bit asserts when the amount of data words in Transmit FIFO changes from 'TX FIFO Almost Empty Flag' + 1 to 'TX FIFO Almost Empty Flag'.
  - 1'b0 – No interrupt
  - 1'b1 – Interrupt pending

- tx_fifo_empty_int

  Transmit FIFO Empty Interrupt Status. This interrupt status bit asserts when the last data in Transmit FIFO is popped-out, causing the FIFO to become empty.
  1'b0 – No interrupt
  1'b1 – Interrupt pending
- rx_fifo_full_int

  Receive FIFO Full Interrupt Status. This interrupt status bit asserts when Receive FIFO full status changes from not full to full state.
  1'b0 – No interrupt
  1'b1 – Interrupt pending
- rx_fifo_afull_int

  Receive FIFO Almost Full Interrupt Status. This interrupt status bit asserts when the amount of data words in Receive FIFO changes from 'RX FIFO Almost Full Flag' – 1 to 'RX FIFO Almost Full Flag'.
  1'b0 – No interrupt
  1'b1 – Interrupt pending
- rx_fifo_ready_int

  Receive FIFO Ready Interrupt Status. This interrupt status bit asserts when Receive FIFO is empty and receives a data word from I2C interface.
  1'b0 – No interrupt
  1'b1 – Interrupt pending

Table 5.10. Interrupt Status Second Register

Field	Name	Access	Width	Reset
[7:4]	reserved	RSVD	4	–
[3]	rx_addr_det_int	RW1C	1	1'b0
[2]	start_det_int	RW1C	1	1'b0
[1]	stop_err_int	RW1C	1	1'b0
[0]	start_err_int	RW1C	1	1'b0

- rx_addr_int

  Received I2C Address Interrupt Status. Indicates that a new transaction has started and that the correctly configured I2C address has been received by the IP.
  1'b0 – No interrupt
  1'b1 – Interrupt pending
- start_det_int

  START Condition Detect Interrupt Status. This interrupt status bit asserts after detecting a START condition or a repeated START condition.
  1'b0 – No interrupt
  1'b1 – Interrupt pending
- stop_err_int

  STOP Condition Error Interrupt Status. This interrupt status bit asserts after detecting a STOP condition when it is not expected. STOP condition is expected to occur only after the ACK/NACK bit. The stop_err_int and stop_det_int do not assert at the same time.
  1'b0 – No interrupt
  1'b1 – Interrupt pending
- start_err_int

  START Condition Error Interrupt Status. This interrupt status bit asserts after detecting a START condition when it is not expected. START condition is expected to occur only when I2C bus is idle and after receiving an ACK or a NACK (repeated START condition).
  1'b0 – No interrupt
  1'b1 – Interrupt pending

5.8. Interrupt Enable Registers (INT_ENABLE1_REG, INT_ENABLE2_REG)

Table 5.11 and Table 5.12 present the summary of Interrupt Enable Registers, they corresponds to interrupts status bits in INT_STATUS1_REG and INT_STATUS2_REG. They do not affect the contents of the INT_STATUS1_REG and INT_STATUS2_REG. If one of the INT_STATUS1_REG/ INT_STATUS2_REG bits asserts, and the corresponding bit of INT_ENABLE1_REG/ INT_ENABLE2_REG is 1'b1, the interrupt signal int_o asserts.

Table 5.11. Interrupt Enable First Register

Field	Name	Access	Width	Reset
[7]	tr_cmp_en	RW	1	1'b0
[6]	stop_det_en	RW	1	1'b0
[5]	tx_fifo_full_en	RW	1	1'b0
[4]	tx_fifo_aempty_en	RW	1	1'b0
[3]	tx_fifo_empty_en	RW	1	1'b0
[2]	rx_fifo_full_en	RW	1	1'b0
[1]	rx_fifo_afull_en	RW	1	1'b0
[0]	rx_fifo_ready_en	RW	1	1'b0

- tr_cmp_en

  Transfer Complete Interrupt Enable. Interrupt enable bit corresponding to Transfer Complete Interrupt Status.
  1'b0 – Interrupt disabled
  1'b1 – Interrupt enabled
- stop_det_en

  STOP Condition Detected Interrupt Enable. Interrupt enable bit corresponding to STOP Condition Detected Interrupt Status.
  1'b0 – Interrupt disabled
  1'b1 – Interrupt enabled
- tx_fifo_full_en

  Transmit FIFO Full Interrupt Enable. Interrupt enable bit corresponding to Transmit FIFO Full Interrupt Status.
  1'b0 – Interrupt disabled
  1'b1 – Interrupt enabled
- tx_fifo_aempty_en

  Transmit FIFO Almost Empty Interrupt Enable. Interrupt enable bit corresponding to Transmit FIFO Almost Empty Interrupt Status.
  1'b0 – Interrupt disabled
  1'b1 – Interrupt enabled
- tx_fifo_empty_en

  Transmit FIFO Empty Interrupt Enable. Interrupt enable bit corresponding to Transmit FIFO Empty Interrupt Status.
  1'b0 – Interrupt disabled
  1'b1 – Interrupt enabled
- rx_fifo_full_en

  Receive FIFO Full Interrupt Enable. Interrupt enable bit corresponding to Receive FIFO Full Interrupt Status.
  1'b0 – Interrupt disabled
  1'b1 – Interrupt enabled
- rx_fifo_afull_en

  Receive FIFO Almost Full Interrupt Enable. Interrupt enable bit corresponding to Receive FIFO Almost Full Interrupt Status.
  1'b0 – Interrupt disabled
  1'b1 – Interrupt enabled

- rx_fifo_ready_en

  Receive FIFO Ready Interrupt Enable. Interrupt enable bit corresponding to Receive FIFO Ready Interrupt Status.
  1'b0 – Interrupt disabled
  1'b1 – Interrupt enabled

Table 5.12. Interrupt Enable Second Register

Field	Name	Access	Width	Reset
[7:4]	reserved	RSVD	4	–
[3]	rx_addr_en	RW	1	1'b0
[2]	start_det_en	RW	1	1'b0
[1]	stop_err_en	RW	1	1'b0
[0]	start_err_en	RW	1	1'b0

- rx_addr_en

  Received I2C Address Interrupt Set. Interrupt enable bit corresponding to the Received I2C Address Interrupt Status.
  1'b0 – Interrupt disabled
  1'b1 – Interrupt enabled
- start_det_en

  START Condition Detected Interrupt Enable. Interrupt enable bit corresponding to START Condition Detected Interrupt Status.
  1'b0 – Interrupt disabled
  1'b1 – Interrupt enabled
- stop_err_en

  STOP Condition Error Interrupt Enable. Interrupt enable bit corresponding to STOP Condition Error Interrupt Status.
  1'b0 – Interrupt disabled
  1'b1 – Interrupt enabled
- start_err_en

  START Condition Error Interrupt Enable. Interrupt enable bit corresponding to START Condition Error Interrupt Status.
  1'b0 – Interrupt disabled
  1'b1 – Interrupt enabled

5.9. Interrupt Set Registers (INT_SET1_REG, INT_SET2_REG)

Table 5.13 and Table 5.14 presents the summary of Interrupt Set Registers. Writing 1'b1 to a register bit in INT_SET1_REG or INT_SET2_REG asserts the corresponding interrupts status bit in INT_STATUS1_REG or INT_STATUS2_REG while writing 1'b0 is ignored. This is intended for testing purposes only.

Table 5.13. Interrupt Set First Register

Field	Name	Access	Width	Reset
[7]	tr_cmp_set	WO	1	1'b0
[6]	stop_det_set	WO	1	1'b0
[5]	tx_fifo_full_set	WO	1	1'b0
[4]	tx_fifo_aempty_set	WO	1	1'b0
[3]	tx_fifo_empty_set	WO	1	1'b0
[2]	rx_fifo_full_set	WO	1	1'b0
[1]	rx_fifo_afull_set	WO	1	1'b0
[0]	rx_fifo_ready_set	WO	1	1'b0

- tr_cmp_set

  Transfer Complete Interrupt Set. Interrupt set bit corresponding to Transfer Complete Interrupt Status.
  1'b0 – No action
  1'b1 – Asserts INT_STATUS1_REG.tr_cmp_int
- stop_det_set

  STOP Condition Detected Interrupt Set. Interrupt set bit corresponding to STOP Condition Detected Interrupt Status.
  1'b0 – No action
  1'b1 – Asserts INT_STATUS1_REG.stop_det_int
- tx_fifo_full_set

  Transmit FIFO Full Interrupt Set. Interrupt set bit corresponding to Transmit FIFO Full Interrupt Status.
  1'b0 – No action
  1'b1 – Asserts INT_STATUS1_REG.tx_fifo_full_int
- tx_fifo_aempty_set

  Transmit FIFO Almost Empty Interrupt Set. Interrupt set bit corresponding to Transmit FIFO Almost Empty Interrupt Status.
  1'b0 – No action
  1'b1 – Asserts INT_STATUS1_REG.tx_fifo_aempty_int
- tx_fifo_empty_set

  Transmit FIFO Empty Interrupt Set. Interrupt set bit corresponding to Transmit FIFO Empty Interrupt Status.
  1'b0 – No action
  1'b1 – Asserts INT_STATUS1_REG.tx_fifo_empty_int
- rx_fifo_full_set

  Receive FIFO Full Interrupt Set. Interrupt set bit corresponding to Receive FIFO Full Interrupt Status.
  1'b0 – No action
  1'b1 – Asserts INT_STATUS1_REG.rx_fifo_full_int
- rx_fifo_afull_set

  Receive FIFO Almost Full Interrupt Set. Interrupt set bit corresponding to Receive FIFO Almost Full Interrupt Status.
  1'b0 – No action
  1'b1 – Asserts INT_STATUS1_REG.rx_fifo_afull_int
- rx_fifo_ready_set

  Receive FIFO Ready Interrupt Set. Interrupt set bit corresponding to Receive FIFO Ready Interrupt Status.
  1'b0 – No action
  1'b1 – Asserts INT_STATUS1_REG.rx_fifo_ready_int

Table 5.14. Interrupt Set Second Register

Field	Name	Access	Width	Reset
[7:4]	reserved	RSVD	4	–
[3]	rx_addr_set	WO	1	1'b0
[2]	start_det_set	WO	1	1'b0
[1]	stop_err_set	WO	1	1'b0
[0]	start_err_set	WO	1	1'b0

- rx_addr_set

  Received I2C Address Interrupt Set. Interrupt set bit corresponding to the Received I2C Address Interrupt Status.
  0 – No action.
  1 – Asserts INT_STATUS2_REG.rx_addr_int.
- start_det_set

  START Condition Detected Interrupt Set. Interrupt set bit corresponding to START Condition Detected Interrupt Status.

0 – No action.
1 – Asserts INT_STATUS2_REG.start_det_int.

- stop_err_set

  STOP Condition Error Interrupt Set. Interrupt set bit corresponding to STOP Condition Error Interrupt Status.
  0 – No action.
  1 – Asserts INT_STATUS2_REG.stop_err_int.

- start_err_set

  START Condition Error Interrupt Set. Interrupt set bit corresponding to START Condition Error Interrupt Status.
  0 – No action.
  1 – Asserts INT_STATUS2_REG.start_err_int.

5.10. FIFO Status Register (FIFO_STATUS_REG)

FIFO Status Register reflects the status of Transmit FIFO and Receive FIFO as shown in Table 5.15.

Table 5.15. FIFO Status Register

Field	Name	Access	Width	Reset
[7:6]	reserved	RSVD	2	–
[5]	tx_fifo_full	RO	1	1'b0
[4]	tx_fifo_aempty	RO	1	1'b1
[3]	tx_fifo_empty	RO	1	1'b1
[2]	rx_fifo_full	RO	1	1'b0
[1]	rx_fifo_afull	RO	1	1'b0
[0]	rx_fifo_empty	RO	1	1'b1

- tx_fifo_full

  Transmit FIFO Full. This bit reflects the full condition of Transmit FIFO.
  1'b0 – Transmit FIFO is not full
  1'b1 – Transmit FIFO is full

- tx_fifo_aempty

  Transmit FIFO Almost Empty. This bit reflects the almost empty condition of Transmit FIFO.
  1'b0 – Data words in Transmit FIFO is greater than 'TX FIFO Almost Empty Flag' attribute
  1'b1 – Data words in Transmit FIFO is less than or equal to 'TX FIFO Almost Empty Flag' attribute

- tx_fifo_empty

  Transmit FIFO Empty. This bit reflects the empty condition of Transmit FIFO.
  1'b0 – Transmit FIFO is not empty – has at least 1 data word
  1'b1 – Transmit FIFO is empty

- rx_fifo_full

  Receive FIFO Full. This bit reflects the full condition of Receive FIFO.
  1'b0 – Receive FIFO is not full
  1'b1 – Receive FIFO is full

- rx_fifo_afull

  Receive FIFO Almost Full. This bit reflects the almost full condition of Receive FIFO.
  1'b0 – Data words in Receive FIFO is less than 'RX FIFO Almost Full Flag' attribute
  1'b1 – Data words in Receive FIFO is greater than or equal to 'RX FIFO Almost Full Flag' attribute

- rx_fifo_empty

  Receive FIFO Empty. This bit reflects the empty condition of Receive FIFO.
  1'b0 – Receive FIFO is not empty – has at least 1 data word
  1'b1 – Receive FIFO is empty

5.11. Received Address Registers (RX_ADDR_1_REG, RX_ADDR_2_REG)

The first (RX_ADDR_1_REG) and second (RX_ADDR_2_REG) most recently received I2C address bytes. RX_ADDR_2_REG is only applicable in 10-bit addressing mode. The LSB of RX_ADDR_1_REG always contains the most recently received RNW bit.

Table 5.16. Received Address 1

Field	Name	Access	Width	Reset
[7:1]	rx_addr_1	RO	7	7'b0
[0]	rx_rnw	RO	1	1'b0

- rx_addr_1

  First address of the most recently received transaction. When using the 7-bit Address Mode, it contains the most recently received address. When using the 10-bit Address Mode, the 5 most significant bits should contain the 10-bit Address Mode reserved address. The next 2 bits contain the two most significant bits of the 10-bit address.
- rx_rnw

  Received I2C Read or Write Bit. Read or write bit of the most recently received I2C transaction.
  1'b0 – Most recently received transaction is a write.
  1'b1 – Most recently received transaction is a read.

Table 5.17. Received Address 2

Field	Name	Access	Width	Reset
[7:0]	rx_addr_2	RO	8	8'b0

- rx_addr_2

Second address of the most recently received transaction. When using the 7-bit Address Mode, data in this field should not be used. When using the 10-bit Address Mode, this field contains the 8 least significant bits of the received address.

6. Example Design

The I2C Target example design allows you to compile, simulate, and test the I2C Target IP on the following Lattice evaluation boards:

- CertusPro-NX Evaluation Board
- MachXO5-NX Development Board

6.1. Example Design Supported Configuration

Note: In the table below, ✓ refers to a checked option while — refers to an unchecked option in the I2C Target IP example design.

Table 6.1. I2C Target IP Configuration Supported by the Example Design

Attribute	I2C Target
General	
APB Mode Enable	✓
Remove Tristate Buffers	—
Addressing Mode	7-bit
I2C Target Address:0x	51
STOP Condition Detected Interrupt Assertion	STOP Condition addressed to IP
System Clock Frequency (MHz)	50
Timing Adjustment	
I2C SDA Register Depth	0
FIFO	
FIFO Width	*8 (Display only)*
FIFO Depth	16
Implementation of FIFO	EBR
TX FIFO Almost Empty Flag	2
RX FIFO Almost Full Flag	14

6.2. Overview of the Example Design and Features

The example design discussed in this section is created using the *RISC-V MC SoC Project* template in the Lattice Propel Design Environment. The generated project includes the following components:

- Processor – RISC-V (MC w/ PIC/Timer)
- GPIO
- Asynchronous SRAM
- UART – Serial port
- PLL
- Glue Logic

I2C Controller and I2C Target IPs are instantiated and connected in the project as shown in Figure 6.1. In this example, I2C Controller and I2C Target are instantiated in the same system. In actual hardware or use case, you can connect the I2C Target IP to external I2C Controller and Target devices.

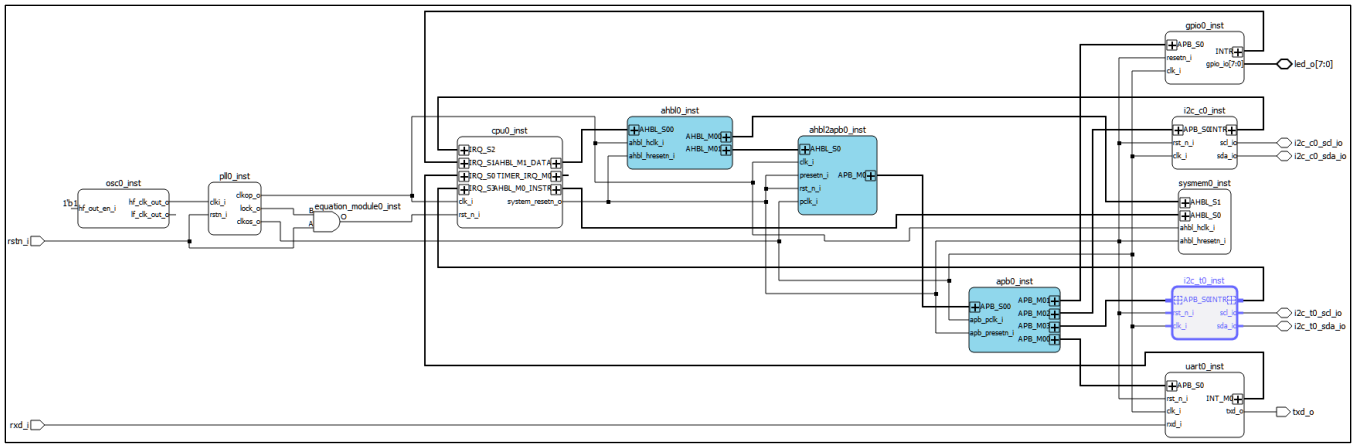

Figure 6.1. I2C Target IP in Propel SoC Project

An Embedded C/C++ Project is also created in the Propel software to enable developing and debugging application code for different IP features. I2C Target features can be tested by sending I2C Commands from an I2C Controller to the I2C Target. Runtime configuration of IP and feature testing can be done through C-Code Test Routine. Figure 6.2 shows an example routine to write to *WR_DATA_REG* and read from *RD_DATA_REG*. This sample test routine can be found in the driver file included in the generated IP.

```
48  uint8_t i2c_slave_data_write(struct i2c_slave_instance *this_i2cs,uint8_t *tx_buf,uint8_t length)
49  {
50      uint8_t count=0;
51      if (this_i2cs == NULL)
52                  return I2C_SLV_PTR_ERROR;
53
54      while (count < length) {
55          reg_8b_write(this_i2cs->base_addr | I2C_SLAVE_DATA_BUFFER,
56                                  tx_buf[count++]);
57      }
58      return I2C_SLV_SUCCESS;
59  }
60
61  uint8_t i2c_slave_data_read(struct i2c_slave_instance *this_i2cs,uint8_t *rx_buf,uint8_t *length)
62  {
63      uint8_t count=0;
64      uint8_t fifo_status = 0;
65      uint8_t rd_data;
66
67      if (this_i2cs == NULL)
68                  return I2C_SLV_PTR_ERROR;
69
70      reg_8b_read(this_i2cs->base_addr | I2C_SLAVE_FIFO_STATUS, &fifo_status);
71
72      while ((fifo_status & I2C_SLAVE_RX_FIFO_EMPTY) == 0) {
73          reg_8b_read(this_i2cs->base_addr | I2C_SLAVE_DATA_BUFFER, &rd_data);
74          *rx_buf=rd_data;
75          rx_buf++;
76          count++;
77          reg_8b_read(this_i2cs->base_addr | I2C_SLAVE_FIFO_STATUS, &fifo_status);
78      }
79      *length = count;
80
81      return I2C_SLV_SUCCESS;
82  }
```

Figure 6.2. Sample C-Code Test Routine

6.3. Design Components Example

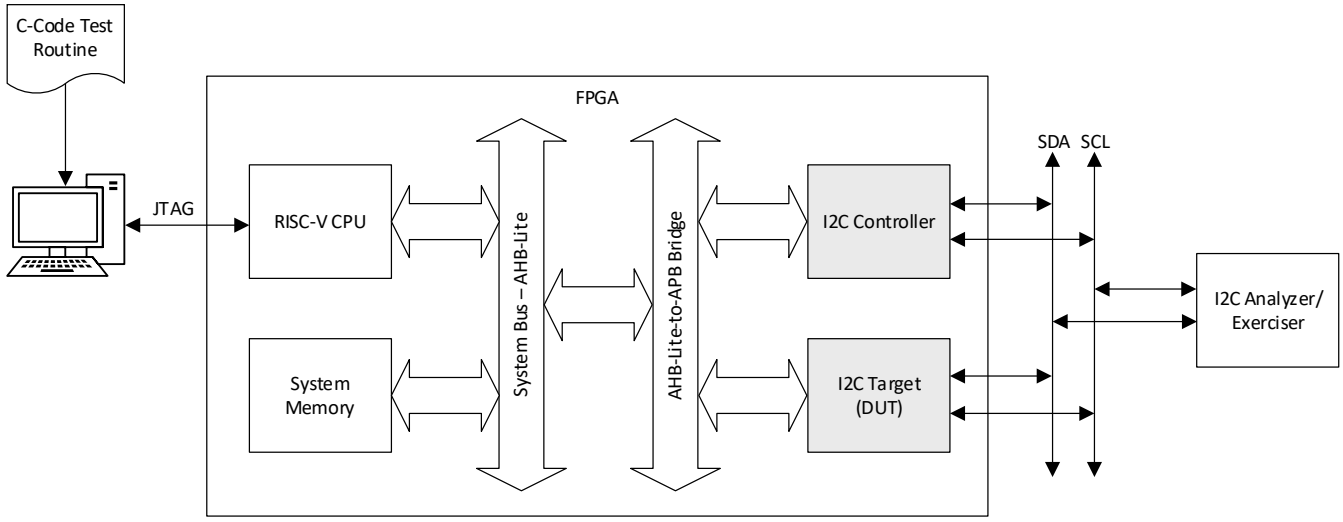

Figure 6.3. I2C Target Example Design Block Diagram

The I2C Target example design includes the following blocks:

- RISC-V CPU – Passes the C-Code Test Routine from system memory to system bus. Handles interrupts.
- Memory – Contains commands for testing.
- System Bus – AHB-Lite systems bus for transfers between memory and IP
- I2C Controller – I2C Controller IP instance and/or I2C Analyzer/Exerciser that acts as the Controller
- I2C Target Device(s) – I2C Target IP instance and/or I2C Analyzer/Exerciser that acts as the Target

6.4. Generating the Example Design

Refer to the Lattice Propel SDK User Guide for more details on the Lattice Propel software.

1. Launch Lattice Propel software and set your workspace directory.
2. In the Propel software, create a new Lattice SoC Design Project by navigating to **File** > **New** > **Lattice SoC Design Project.**
3. The **Create SoC Project** window will open.

   In **Device Select** section, indicate the correct details of the device or board that you are using. In Figure 6.4, device is set to LFMXO5-25-9BBG400C since MachXO5-NX Development Board is used in the hardware testing. In **Template Design** section, choose **RISC-V MC SoC Project**. Click **Finish**.

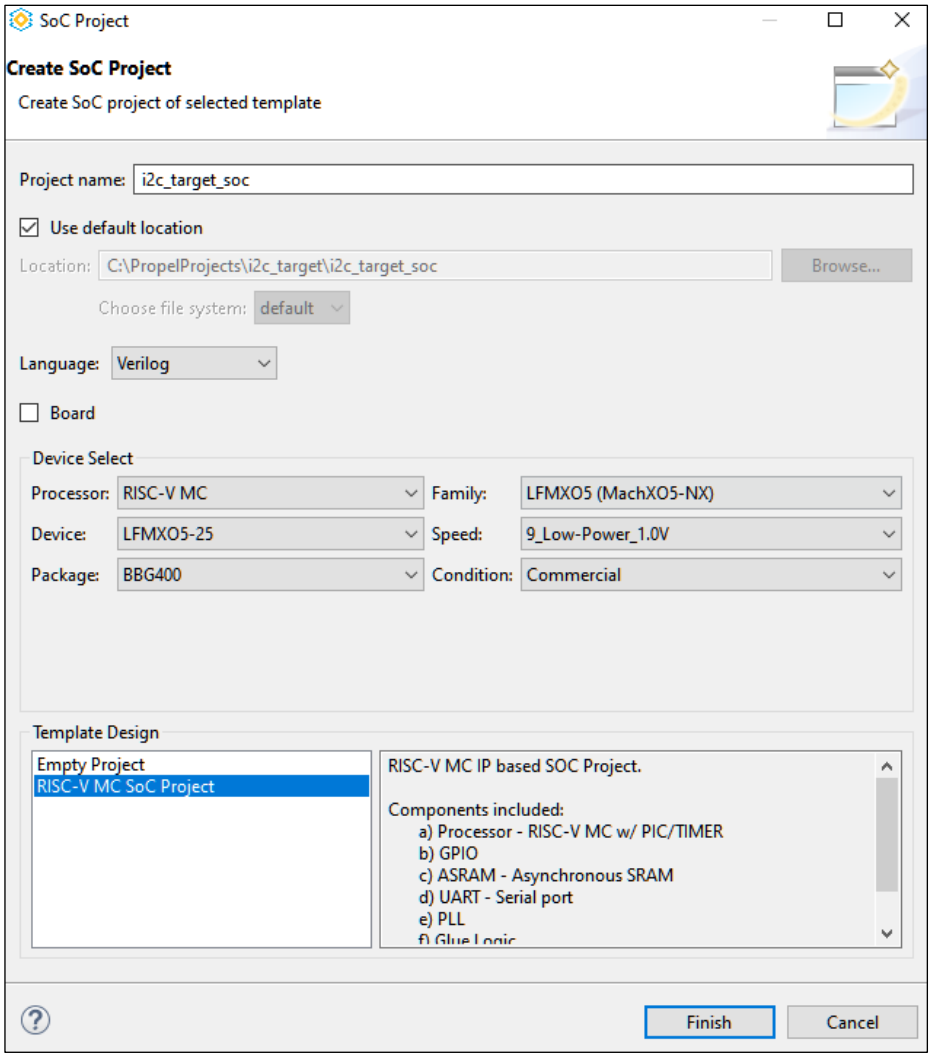

Figure 6.4. Create SoC Project

4. Run Propel Builder by clicking the icon or navigate to **LatticeTools** > **Open Design** in Propel Builder**.** The Propel Builder will open and load the design template.
5. In the **IP Catalog** tab, instantiate the I2C Target IP. Refer to the Generating and Instantiating the IP section for more details. In this example, there is one instance of the I2C Target IP.

   See the Example Design Supported Configuration section for the corresponding parameter settings.
6. After generating the IP, the **Define Instance** window will open. Modify the instance name if needed, then click **OK.**

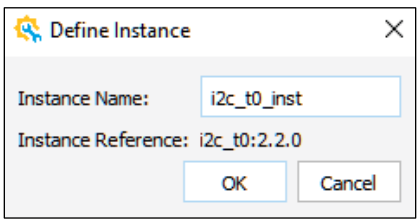

Figure 6.5. Define Instance

7. Connect the instantiated IPs to the system. Refer to Figure 6.1 for the connections used in this IP. You will need to update other components of the system for clock and reset sources, interrupt, and bus interface.

8. Click the icon or navigate to **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 software, build your SoC project to generate the system environment needed for the embedded C/C++ project. Select your SoC project then navigate to **Project** > **Build Project**.
11. Check the build result from the **Console** view.

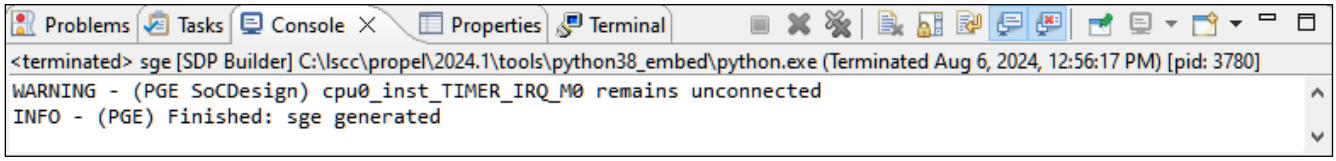

Figure 6.6. Build SoC Project Result

12. Generate a new Lattice C/C++ project by navigating to **File** > **New** > **Lattice C/C++ Project**. Update your **Project name**, click **Next**, and then click **Finish**.

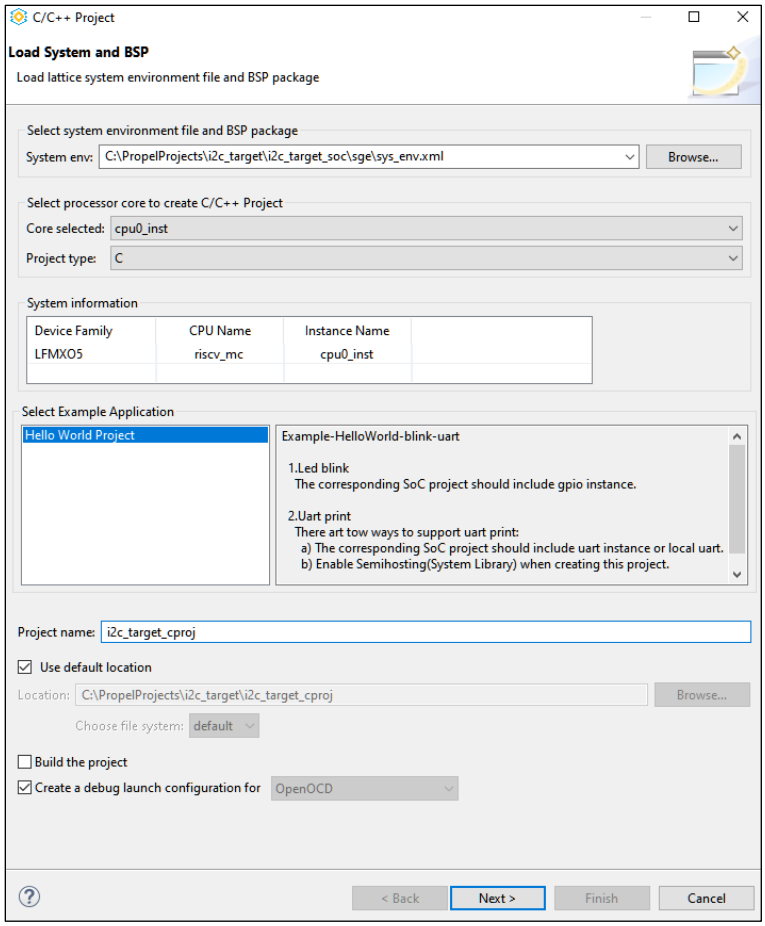

Figure 6.7. Lattice C/C++ Design Project

13. Select your C/C++ project then select **Project** > **Build**.
14. Check the build result from the **Console** view.

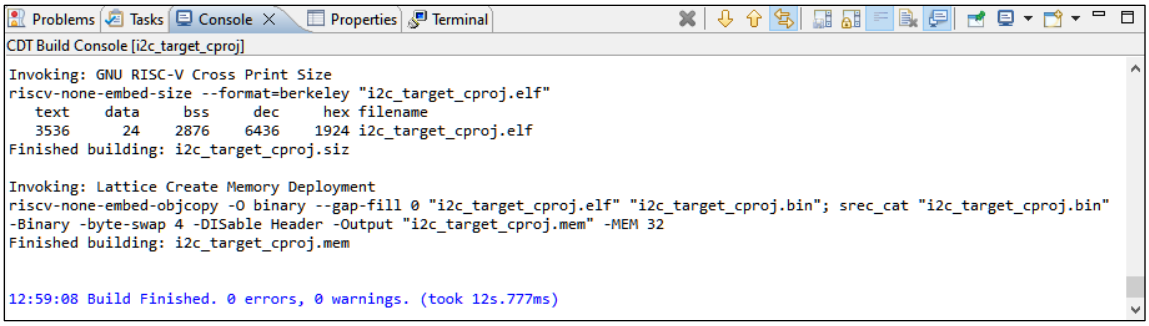

Figure 6.8. Build C/C++ Project Result

15. This environment is now ready for running your tests on the device. Refer to the *Lattice Propel Tutorial* section of the Lattice Propel SDK User Guide for step-by-step guide.

6.5. Hardware Testing

6.5.1. Hardware Testing Setup

Download the generated bitstream file from the Generating the Example Design section to the MachXO5-NX Development Board via the Lattice Radiant Programmer.

6.5.2. Expected Output

Below is a sample waveform captured via I2C Analyzer/Exerciser tool.

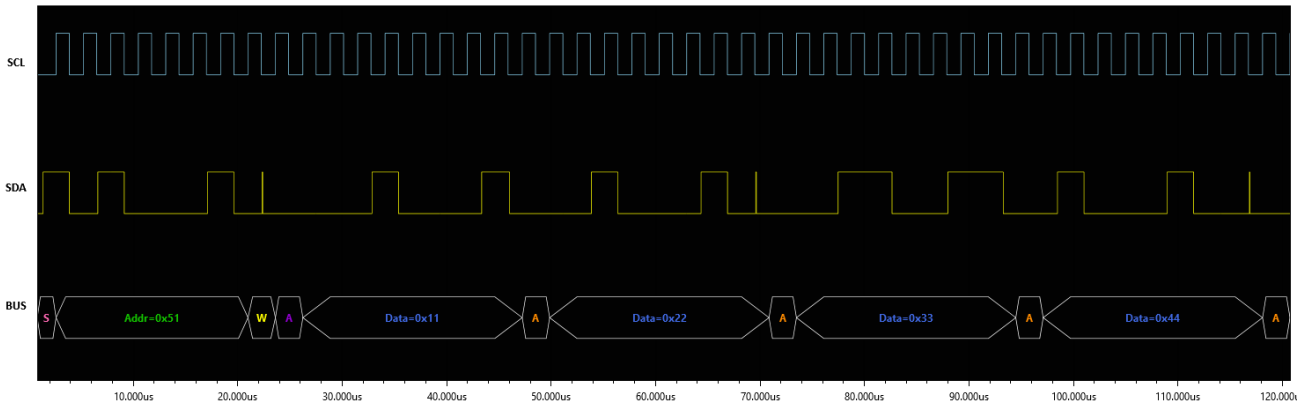

Figure 6.9. Sample I2C Write Command Sent to I2C Target

7. Designing with the IP

This section provides information on how to generate the IP Core using the Lattice Radiant software and how to run simulation and synthesis. 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.

7.1. Generating and Instantiating the IP

The Lattice Radiant software allows you to customize and generate modules and IPs and integrate them into the device's architecture. The procedure for generating the I2C Target IP in Lattice Radiant software is described below.

To generate the I2C Target IP:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click on **I2C Target** 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 the **Create in** fields and click **Next**.

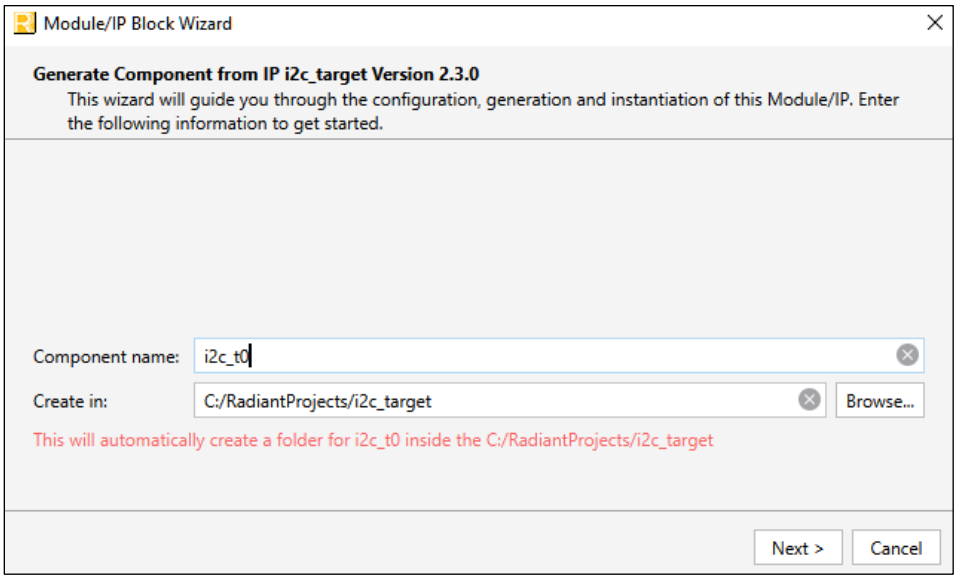

Figure 7.1. Module/IP Block Wizard

3. In the module's dialog box of the **Module/IP Block Wizard** window, customize the selected I2C Target IP using drop-down menus and check boxes. As a sample configuration, see Figure 7.2. For configuration options, see the IP Parameter Description section.

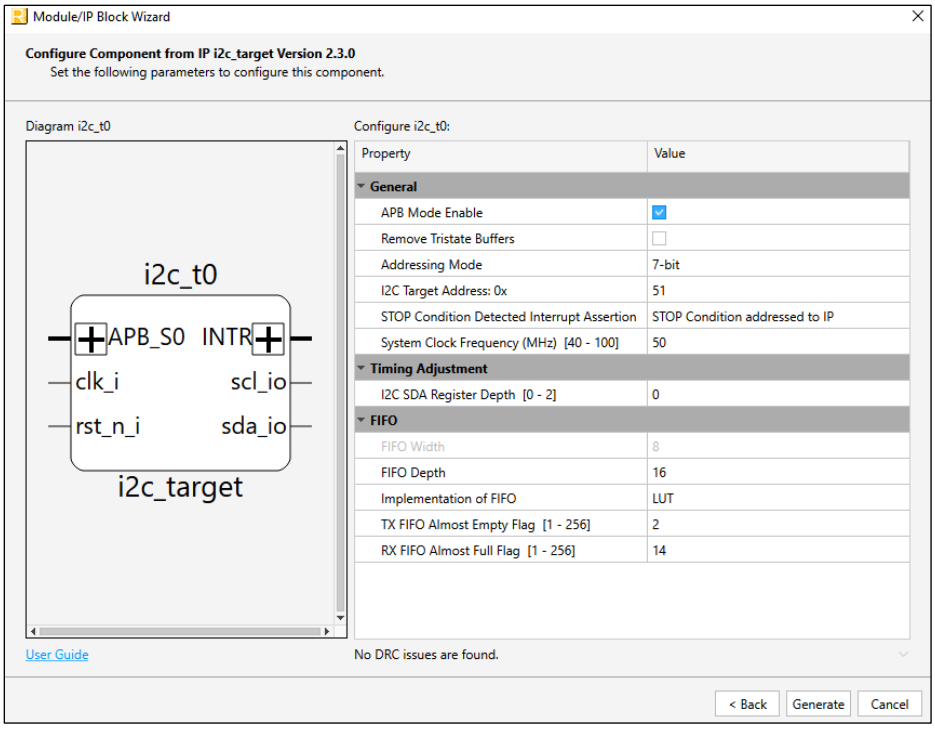

Figure 7.2. Configure User Interface of I2C Target IP

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

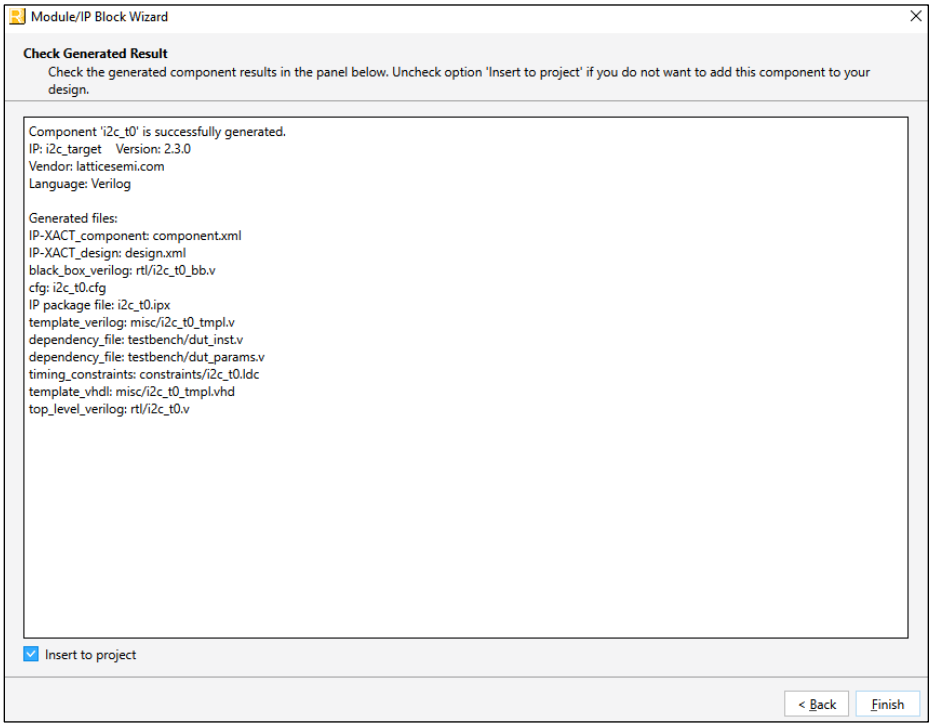

Figure 7.3. Check Generating Result

5. Click the **Finish** button. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields shown in Figure 7.1.

7.1.1. Generated Files and File Structure

The generated I2C Target IP package includes the closed-box (<Component name>_bb.v) and instance templates (<Component name>_tmpl.v/vhd) that can be used to instantiate the IP in a top-level design. An example RTL top-level reference source file (<Component name>.v) that can be used as an instantiation template for the IP is also provided. You may also use this top-level reference as the starting template for the top-level for your complete design. The generated files are listed in Table 7.1.

Table 7.1. Generated File List

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

7.2. Design Implementation

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

A post-synthesis constraint file (.pdc) contains both timing and non-timing constraints. Constraints that are added using the Device Constraint Editor are saved to the active .pdc file. The active post-synthesis design constraint file is then used as input for post-synthesis processes.

7.2.1. Device Constraint Editor

Refer to the Lattice Radiant Software User Guide for more information on how to use the device constraint editor.

7.2.2. Manual PDC File Creation

Add the post-synthesis constraint file (.pdc) in the Lattice Radiant software and define the I/O pins according to the schematic design for ports defined in your design. You can define different types of constraints such as pins, clocks, and other timing paths.

7.3. Timing Constraints

The timing constraints are based on the clock frequency used. The timing constraints for the IP are defined in relevant constraint files. The example below shows the IP timing constraints generated for I2C Target IP.
You need to provide proper timing and physical design constraints to ensure that your design meets the desired performance goals on the FPGA. Add the content of the following IP constraint file to your design constraints:

<IP_Instance_Path>/<IP_Instance_Name>/constraints/<IP_Instance_Name>.ldc

The constraint file has been verified during IP evaluation with the IP instantiated directly in the top-level module. You can modify the constraints in this file with thorough understanding of the effect of each constraint.
To use this constraint file:

1. Copy the contents of <IP_Instance_Name>.ldc to the top-level design constraint for post-synthesis
2. Remove create_clock constraints if the I2C Target is instantiated in another module.
3. Remove ldc_set_port for scl_io and sda_io if REMOVE_TRISTATE buffer is enabled.

Refer to Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059) for details on how to constrain your design.

7.4. Running Functional Simulation

1. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard** shown in Figure 7.4.

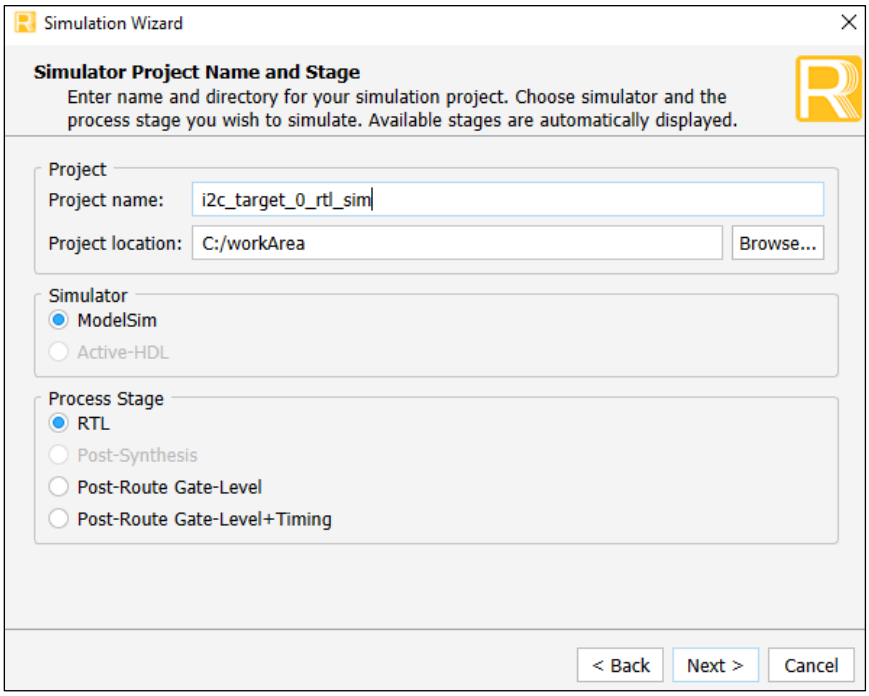

Figure 7.4. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window as shown in Figure 7.5.

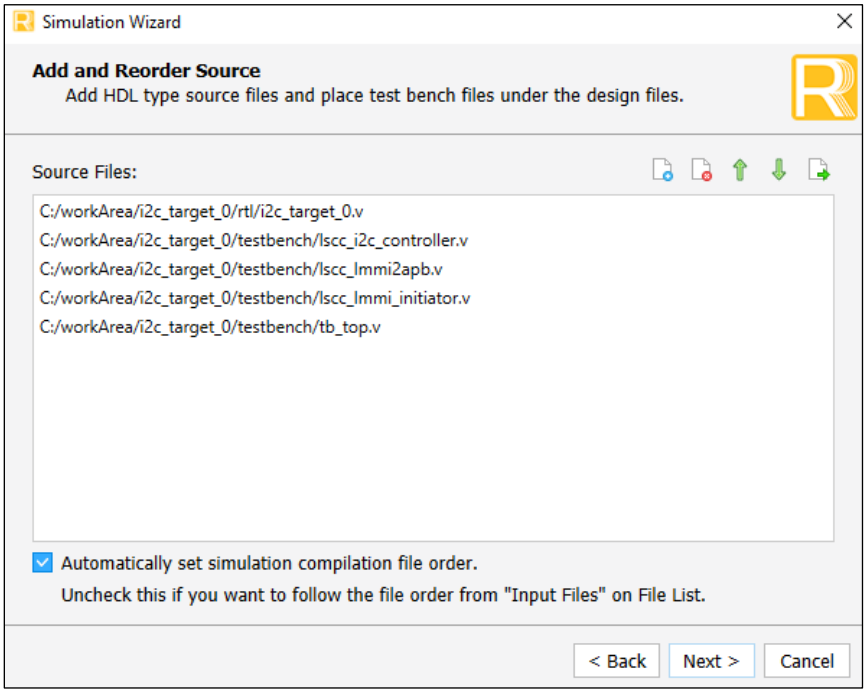

Figure 7.5. Adding and Reordering Source

3. Click **Next**. The **Summary** window is shown. Click **Finish** to run the simulation.

Note: It is necessary to follow the procedure above until it is fully automated in the Lattice Radiant software suite.

The results of the simulation in our example are provided in Figure 7.6.

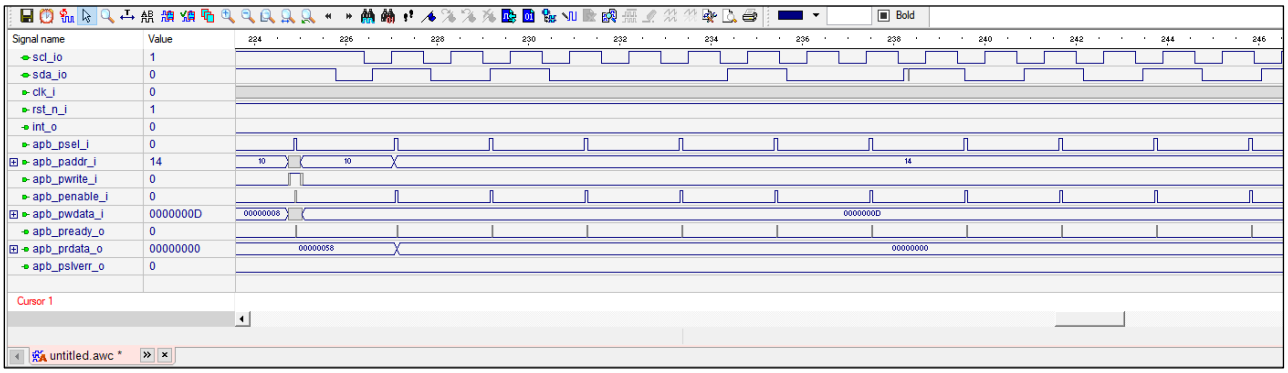

Figure 7.6. Simulation Waveform

Appendix A. Resource Utilization

Table A.1 shows the resource utilization of the I2C Target IP for the LAV-AT-E70-1LFG1156I device using Synplify Pro of the Lattice Radiant Software 2023.1. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.1. LAV-AT-E70-1LFG1156I Device Resource Utilization

Configuration	Clk Fmax (MHz)[1]	Registers	LUTs	EBRs
Default	107	320	394	0
APB Mode Enable: false, Others = Default	251	318	394	0
Implementation of FIFO: EBR, Others = Default	127	304	372	2
TX FIFO Almost Empty Flag: 1, *RX FIFO Almost Full Flag*: 1, Others = Default	112	320	394	0
FIFO Depth: 256, *TX FIFO Almost Empty Flag*: 256, *RX FIFO Almost Full Flag*: 256, Others = Default	105	350	910	0

Note:

1. Fmax is generated when the FPGA design contains only I2C Target IP, and the target frequency is 50 MHz. These values may be reduced when user logic is added to the FPGA design.

Table A.2 shows the resource utilization of the I2C Target IP for the LFMXO5-25-9BBG400I device using Synplify Pro of the Lattice Radiant Software 3.1. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.2. LFMXO5-25-9BBG400I Device Resource Utilization

Configuration	Clk Fmax (MHz)[1]	Registers	LUTs	EBRs
Default	168	319	397	0
APB Mode Enable: false, Others = Default	200	304	370	0
Implementation of FIFO: EBR, Others = Default	157	303	364	2
TX FIFO Almost Empty Flag: 1, *RX FIFO Almost Full Flag*: 1, Others = Default	168	319	385	0
FIFO Depth: 256, *TX FIFO Almost Empty Flag*: 256, *RX FIFO Almost Full Flag*: 256, Others = Default	167	349	1088	0

Note:

1. Fmax is generated when the FPGA design contains only I2C Target IP, and the target frequency is 50 MHz. These values may be reduced when user logic is added to the FPGA design.

Table A.3 shows the resource utilization of the I2C Target IP for the LFMXO5-25-7BBG400I device using Synplify Pro of the Lattice Radiant Software 3.1. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.3. LFMXO5-25-7BBG400I Device Resource Utilization

Configuration	Clk Fmax (MHz)[1]	Registers	LUTs	EBRs
Default	101	319	385	0
APB Mode Enable: false, Others = Default	120	304	370	0
Implementation of FIFO: EBR, Others = Default	103	303	365	2
TX FIFO Almost Empty Flag: 1, *RX FIFO Almost Full Flag*: 1, Others = Default	94	319	385	0
FIFO Depth: 256, *TX FIFO Almost Empty Flag*: 256, *RX FIFO Almost Full Flag*: 256, Others = Default	101	349	1089	0

Note:

1. Fmax is generated when the FPGA design contains only I2C Target IP, and the target frequency is 50 MHz. These values may be reduced when user logic is added to the FPGA design.

Table A.4 shows the resource utilization of the I2C Target IP for the LN2-CT-20-1CBG484C device using Synplify Pro of the Lattice Radiant Software 2024.2. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.4. LN2-CT-20-1CBG484C Device Resource Utilization

Configuration	Clk Fmax (MHz)[1]	Registers	LUTs	EBRs
Default	103.146	343	393	0
APB Mode Enable: false, Others = Default	250	328	396	0
Implementation of FIFO: EBR, Others = Default	115.128	327	378	2
TX FIFO Almost Empty Flag: 1, *RX FIFO Almost Full Flag*: 1, Others = Default	97.314	343	393	0
FIFO Depth: 256, *TX FIFO Almost Empty Flag*: 1, *RX FIFO Almost Full Flag*: 256, Others = Default	120.120	414	936	0

Note:

1. Fmax is generated when the FPGA design contains only I2C Target IP, and the target frequency is 50 MHz. These values may be reduced when user logic is added to the FPGA design.

Table A.5 shows the resource utilization of the I2C Target IP for the LFMXO4-110HC-5BBG484I device using Synplify Pro of the Lattice Radiant Software 2025.2. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.5. LFMXO4-110HC-5BBG484I Device Resource Utilization

Configuration	Clk Fmax (MHz)[1]	Registers	LUTs	EBRs
Default	77.35	572	555	0
APB Mode Enable: false, Others = Default	97.95	541	538	0
Implementation of FIFO: EBR, Others = Default	69.47	397	283	2
TX FIFO Almost Empty Flag: 1, *RX FIFO Almost Full Flag*: 1, Others = Default	76.29	563	555	0
FIFO Depth: 256, *TX FIFO Almost Empty Flag*: 1, *RX FIFO Almost Full Flag*: 256, Others = Default	61.93	3197	4515	0

Note:

1. Fmax is generated when the FPGA design contains only I2C Target IP, and the target frequency is 50 MHz. These values may be reduced when user logic is added to the FPGA design.

References

For more information, refer to:

- Lattice Memory Mapped Interface and Lattice Interrupt Interface (FPGA-UG-02039)
- AMBA 3 APB Protocol v1.0 Specification
- I2C Bus Specification and User Manual
- Lattice Radiant Software 2023.1 User Guide
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- Reveal User Guide for Radiant Software
- I2C Target IP Release Notes (FPGA-RN-02028)
- I2C Target IP Core web page
- Lattice Propel Design Environment web page
- Lattice Radiant Software web page
- Lattice Solutions IP Cores web page
- Lattice Solutions Reference Designs web page
- Avant-E web page
- Avant-G web page
- Avant-X web page
- Certus-N2 web page
- Certus-NX web page
- CertusPro-NX web page
- CrossLink-NX web page
- MachXO4 web page
- MachXO5-NX web page
- CertusPro-NX Evaluation Board web page
- MachXO5-NX Development Board 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/en/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 2.3, IP v2.6.0, June 2026

Section	Change Summary
All	• Updated the IP version information on the cover page. • Made editorial fixes.
Abbreviations in This Document	• Replaced *acronym* with *abbreviation*. • Added *AHB*, *CSR*, and *PDC*.

Introduction	• Updated *Lattice Implementation* in Table 1.1. Summary of the I2C Target IP.
	• In the IP Support Summary section:
	  • Updated Table 1.2. I2C Target IP Support Readiness.
	  • Added a reference to the *Example Design* section.
	• Updated the Features section.
	• Removed the *Hardware Support* section.
	• Added *and Ordering* to the Licensing and Ordering Information header.

Functional Description	• Updated Figure 2.1. I2C Target IP Functional Diagram. • Removed the *Clocking and Reset* section. • Added the Clocking and Reset sections. • Reordered the User Interfaces section and updated its content. • Updated the Clock Stretching and ACK/NACK Response sections. • Added *y* to *tx_fifo_aempty_int* in the Data Transfer in response to I2C Controller Read section.

IP Parameter Description	• In Table 3.1. Attributes Table:
	  • Updated the selectable values for the *I2C Target Address:0x* attribute.
	  • Added *1* to *INT_STATUS1_REG.tx_fifo_aempty_int* for the *TX FIFO Almost Empty Flag* and *RX FIFO Almost Full Flag* attributes.

Register Description	• Updated the *rx_addr_int* and *tr_cmp_int* descriptions in the Interrupt Status Registers (INT_STATUS1_REG, INT_STATUS2_REG) section. • Added *det_* to *rx_addr_det_int* in Table 5.10. Interrupt Status Second Register. • Updated the *rx_addr_en* description in the Interrupt Enable Registers (INT_ENABLE1_REG, INT_ENABLE2_REG) section.
Example Design	• Updated the *I2C Target Address:0x* attribute value in Table 6.1. I2C Target IP Configuration Supported by the Example Design. • Updated the tutorial section name in the Generating the Example Design section.

Revision 2.2, IP v2.5.0, December 2025

Section	Change Summary
All	• Updated the IP version information on the cover page. • Added a note on the IP version in the *Quick Facts* and *Revision History* sections. • Made editorial fixes.

Introduction	• In Table 1.1. Summary of the I2C Target IP:
	  • Added MachXO4 to Supported Devices.
	  • Updated Lattice Implementation.
	• In Table 1.2. I2C Target IP Support Readiness:
	  • Added a table note.
	  • Removed the *IP* column.
	• Added the Attribute Names section.

Signal Description	In Table 4.1. I2C Target IP Signal Description:

Section	Change Summary
	• Added *(Remove Tristate Buffers = 0)* to the *I2C Interface* header. • Added ports for *I2C Interface (Remove Tristate Buffers = 1).*
Designing with the IP	Added a note on screenshots in this section.
Resource Utilization	Added the MachXO4 device resource utilization.
References	Added *I2C Target IP Core*, *MachXO4*, *MachXO5*-NX, *Lattice Propel Design Environment*, and *Lattice Solutions Reference Designs* web pages.

Revision 2.1, IP v2.4.0, June 2025

Section	Change Summary
All	Updated the IP version information on the cover page.

Introduction	• In Table 1.1. Summary of the I2C Target IP:
	  • Updated *Supported FPGA Family* to *Supported Devices*.
	  • Removed *Targeted Devices*.

Revision 2.0, IP v2.3.0, December 2024

Section	Change Summary
All	• Added the IP version information on the cover page. • Updated *I²C* to *I2C*. • Removed the *Debugging* section. • Made editorial fixes.
Acronyms in This Document	Added *Design Under Test* (*DUT*), *Embedded Block RAM* (*EBR*), *First In, First Out* (*FIFO*), *General Purpose Input/Output* (*GPIO*), *Input/Output* (*I/O*), *Intellectual Property* (*IP*), *Lattice Synthesis Engine* (*LSE*), *Look-Up Table* (*LUT*), *Programmable Interrupt Controller* (*PIC*), *Phase-Locked Loop* (*PLL*), *Random Access Memory* (*RAM*), *Receiver* (*RX*), *Reduced Instruction Set Computer Five* (*RISC-V*), *Serial Clock* (*SCL*), *Static Random Access Memory* (*SRAM*), *System on Chip* (*SoC*), *Serial Data* (*SDA*), *Transmitter* (*TX*), and *Universal Asynchronous Receiver/Transmitter* (*UART*).

Introduction	• Updated Table 1.1. Summary of the I2C Target IP:
	  • Added the *Certus-N2* device family to *Supported FPGA Family*.
	  • Removed *IP Version* and added *IP Changes*.
	  • Added the *LFD2NX-9*, *LFD2NX-28*, *LAV-AT-G70*, *LAV-AT-X70*, and *LN2-CT-20* devices to *Targeted Devices*.
	  • Updated *Resources* and *Lattice Implementation*.
	• Updated the Licensing Information and Minimum Device Requirements sections.
	• Replaced the *IP Validation Summary* section with the IP Support Summary section.
	• Removed the *1.7.3. Host* and *1.7.4. Attribute* sections.

Functional Description	• Changed *I2C Controller* to *I2C Target* and added description to *CONTROL_REG* in the Initialization section. • Added *(interrupts are disabled)* to steps *3* and *7* in the Data Transfer in response to I2C Controller Read section and steps 2 and 6 in the Data Transfer in response to I2C Controller Write section.
IP Parameter Description	Updated this section.
Register Description	Updated the description for *stop_det_int* in the Interrupt Status Registers (INT_STATUS1_REG, INT_STATUS2_REG) section.
Example Design	Added this section.
Designing with the IP	• Updated the paragraph in the Designing with the IP section. • Updated Figure 7.1. Module/IP Block Wizard, Figure 7.2. Configure User Interface of I2C Target IP, and Figure 7.3. Check Generating Result.
Resource Utilization	Added resource utilizations for the Lattice Radiant software version 2024.2.
References	Added the *Lattice Solutions IP Cores* web page, *Avant-G* web page, *Avant-X* web page, *Certus-N2* web page, *CertusPro-NX Evaluation Board* web page, *MachXO5-NX Development*

Section	Change Summary
	Board web page, and *I2C Target IP Release Notes (FPGA-RN-02028)*.

Revision 1.9, June 2024

Section	Change Summary
IP Parameter Description	Added *Remove Tristate Buffers* attribute to Table 3.1. Attributes Table and Table 3.2. Attributes Descriptions.

Revision 1.8, March 2024

Section	Change Summary
Functional Description	Updated Step 1 to point to Step 6 if the data to receive is <= FIFO Depth in 2.5.3 Data Transfer in response to I2C Controller Write section.
Register Description	Updated stop_det_int description in 5.7 Interrupt Status Registers (INT_STATUS1_REG, INT_STATUS2_REG) section.

Revision 1.7, January 2024

Section	Change Summary
All	• Renamed the document from *I²C Target IP Core – Lattice Radiant Software* to *I²C Target IP*. • Updated the instances of *I²C Target IP Core* to *I²C Target IP*. • Made editorial fixes. • Minor adjustments to ensure that the document is consistent with Lattice Semiconductor's inclusive language policy. • Updated the device name from *LAV-AT-500E* to *LAV-AT-E70*.
Disclaimers	Updated boilerplate.
Inclusive Language	Added boilerplate.
Introduction	• Moved the first two paragraphs of the 1. Introduction section to 1.1. Overview of the IP section. • Updated the heading number for 1.2. Quick Facts and 1.3. Features sections. • Updated Table 1.1 caption from *Quick Facts* to *Summary of the I²C Target IP*. • Updated the IP Version from *2.0.0* to *2.1.0* in Table 1.1. Summary of the I2C Target IP and Table 1.2. IP Validation Level. • Moved the previous *3.1. Licensing the IP*, and *3.4. IP Evaluation* sections to 1.4. Licensing Information section. • Moved the previous *3.5. Hardware Validation* section to 1.5. IP Validation Summary section. • Added the 1.6. Minimum Device Requirements section. • Updated the previous header from *1.3. Conventions* to 1.7. Naming Conventions.
Functional Description	• Updated the previous header from *2.1. Overview* to 2.1. IP Architecture Overview. • Added the 2.2. Clocking and Reset section. • Moved the previous *2.5. Operations Details (except 2.5.5. Selectable Memory-Mapped Interface)* section to 2.3. Operations Details section. • Moved the previous *2.5.5. Selectable Memory-Mapped Interface* to 2.4.1. Selectable Memory-Mapped Interface section. • Moved the previous *2.6. Programming Flow* section to 2.5. Programming Flow section.
IP Parameter Description	• Moved the previous *2.3. Attributes Summary* section to 3. IP Parameter Description section. • Removed *Remove Tristate Buffers* attribute from Table 3.1. Attributes Table and Table 3.2. Attributes Descriptions.
Signal Description	Moved the previous *2.2. Signal Description* section to 4. Signal Description section.
Register Description	Moved the previous *2.4. Register Description* section to 5. Register Description section.

Section	Change Summary
Designing with the IP	• Moved the previous section *3.2. Generation and Synthesis* to 6.1. Generating and Instantiating the IP section. • Updated Figure 6.1. Module/IP Block Wizard, Figure 6.2. Configure User Interface of I2C Target IP, and Figure 6.3. Check Generating Result. • Added 6.2. Design Implementation and 6.3. Timing Constraints sections. • Moved the previous section *3.3. Running Functional Simulation* to 6.4. Running Functional Simulation section.
Debugging	Added this section.
References	• Updated the names of existing references. • Added references to *Lattice Memory Mapped Interface and Lattice Interrupt Interface (FPGA-UG-02039)*, *AMBA 3 APB Protocol v1.0 Specification*, *I²C Bus Specification and User Manual*, *Lattice Radiant Software 2023.1 User Guide*, *Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)*, *Reveal User Guide for Radiant Software*, *Avant-E web page*, and *Lattice Insights web page*.

Revision 1.6, August 2023

Section	Change Summary
All	• Changed the word 'Master' to 'Controller'. • Changed the word 'Slave' to 'Target'.
Introduction	In Table 1.1. Quick Facts: • Added Certus-NX-RT and CertusPro-NX-RT to Supported FPGA Family. • Added LIFCL-33, LFCPNX-50, LFMXO5-55T, LFMXO5-100T, UT24C40, and UT24CP100 to Targeted Devices. • Updated Lattice Implementation to 'IP Core v2.x.x - Lattice Radiant Software 2022.1 or later and Lattice Propel Builder 2022.1 or later'.
Functional Description	• Updated Figure 2.1. I2C Target IP Core Functional Diagram. • Updated Table 2.1. I2C Target IP Core Signal Description. • Updated Table 2.2. Attributes Table. • Updated Table 2.3. Attributes Descriptions. • Updated Table 2.4. Registers Address Map. • Updated Table 2.8. Target Address Lower Register. • Updated Table 2.11. Target Byte Count Register. • Updated the Register Description section. • Updated the General I2C Operation section. • Updated the Programming Flow section.
Core Generation, Simulation, and Validation	• Updated the Generation and Synthesis section. • Updated Figure 3.1. Module/IP Block Wizard. • Updated Figure 3.2. Configure User Interface of I2C Target IP Core. • Updated Figure 3.3. Check Generating Result. • Updated Figure 3.4. Simulation Wizard. • Updated Figure 3.5. Adding and Reordering Source.
References	Added reference links to the following: • Lattice Radiant Software website • CrossLink-NX FPGA website • Certus-NX FPGA website • CertusPro-NX FPGA website • MachXO5-NX FPGA website • Lattice Avant-E FPGA website

Revision 1.5, April 2023

Section	Change Summary
Introduction	In Table 1.1. Summary of the I2C Target IP: • Added Lattice Avant to Supported FPGA Family. • Added LAV-AT-500E to Targeted Devices.
Functional Description	• Revised Figure 2.1. I2C Target IP Functional Diagram to update the int_o connection. • Revised Table 5.1. Registers Address Map to add new register fields. • Revised Table 5.7. Control Register to add new register fields and their descriptions. • Revised Table 5.10. Interrupt Status Second Register, Table 5.12. Interrupt Enable Second Register, and Table 5.14. Interrupt Set Second Register to add new register fields and their descriptions. • Added Received Address Registers (RX_ADDR_1_REG, RX_ADDR_2_REG) section. • Consolidated the content of the Clock Stretching sections, and removed the duplicate Clock Stretching text.
Core Generation, Simulation, and Validation	• Updated Figure 6.1. Module/IP Block Wizard, • Figure 6.2. Configure User Interface of I2C Target IP, and Figure 6.3. Check Generating Result to show the latest version 1.4.0. • Changed the section title Hardware Evaluation to IP Evaluation. • Changed default value for the hardware evaluation capability from 'enabled' to 'disabled' in IP Evaluation section. • Added Hardware Validation section. • Moved the Data Transfer in response to I2C Combined Format Read section from 2.4.13 to 2.6.4.
Appendix A. Resource Utilization	Added Resource Utilization for LAV-AT.
Technical Support Assistance	Added link to the Lattice Answer Database.

Revision 1.4, May 2022

Section	Change Summary
Introduction	Added MachXO5-NX to the Supported FPGA Family, and LFMXO5-25 to the Supported User Interfaces in Table 1.1.
Functional Description	Corrected range for TX FIFO Almost Empty Flag and RX FIFO Almost Full Flag in Table 3.1.
IP Core Generation, Simulation, and Validation	Updated Figure 6.1, Figure 6.2, and Figure 6.3 to show the latest version 1.3.0.
Appendix A. Resource Utilization	• Updated Table A.2 for resource utilization of the I2C Target IP Core for the LFMXO5-25-9BBG400I. • Added Table A.3 for resource utilization of the I2C Target IP Core for the LFMXO5-25-7BBG400I.

Revision 1.3, June 2021

Section	Change Summary

Introduction	• Remove second paragraph.
	• Updated Table 1.1. Summary of the I2C Target IP.
	  • Revised Supported FPGA Families
	  • Revised Targeted Devices
	  • Revised Lattice Implementation.
	  • Revised reference to Lattice Radiant Software User Guide

References	Updated this section.

Revision 1.2, June 2020

Section	Change Summary
Introduction	Updated Table 1.1. Quick Facts. • Added Certus-NX as supported FPGA family. • Added LFD2NX-40 as targeted device. • Changed Synplify version to Pro for Lattice.
Functional Description	• Updated Implementation of FIFO attribute default value to LUT in Table 2.2. • Updated reset values in Table 2.8, Table 2.9, and Table 2.10. • The wait state of APB read transaction has been reduced to one in Selectable Memory-Mapped Interface section. • Updated content of Initialization section.
Appendix A. Resource Utilization	Updated Table A1.
References	Updated this section.

Revision 1.1, February 2020

Section	Change Summary
Introduction	Updated Table 1.1. Quick Facts. • Changed LIFCL to CrossLink-NX as supported FPGA family. • Added LIFCL-17 as targeted device.
Core Generation, Simulation, and Validation	Updated user interface item to *IP, Processors_Controllers_and_Peripherals* category.

Revision 1.0, December 2019

Section	Change Summary
All	Changed document status from Preliminary to final.
Introduction	Updated Table 1.1. Quick Facts. • Added Resource information. • Updated Lattice Implementation information.
References	Removed reference to the FPGA device web page.
Appendix A. Resource Utilization	Added resource utilization information.
All	Minor editorial changes

Revision 0.80, October 2019

Section	Change Summary
All	Preliminary release.

www.latticesemi.com