I3C Filter IP

IP Version: v1.7.0

User Guide

FPGA-IPUG-02257-1.4

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
ACK	Acknowledgment
AHBL/AHB-Lite	Advanced High-performance Bus – Lite
CCC	Common Command Code
CPU	Central Processing Unit
EBR	Embedded Block RAM
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
HDL	Hardware Description Language
HDR	High Data Rate
I/O	Input/Output
I2C	Inter-Integrated Circuit
I3C	Improved Inter-Integrated Circuit
IBI	In-Band Interrupt
IP	Intellectual Property
LSE	Lattice Synthesis Engine
MC	Microcontroller
MIPI	Mobile Industry Processor Interface
PIC	Programmable Interrupt Controller
PLL	Phase-Locked Loop
RAM	Random Access Memory
RISC-V	Reduced Instruction Set Computer Five
RoT	Root of Trust
RO	Read-Only access
RW	Read-Write access
RW1C	Read and Write 1 to Clear
SCL	Serial Clock
SDA	Serial Data
SDK	Software Development Kit
SDR	Single Data Rate
SMBus	System Management Bus
SoC	System on Chip
WO	Write-Only access

1. Introduction

1.1. Overview of the IP

The I3C Filter IP functions as an invisible relay from the point of view of the Controller and Target devices on the bus. The IP must be connected between the Controller and all Target devices at the SDA and SCL ports. This IP protects all Target devices from malicious traffic generated from the Controller port based on a list of allowable commands set by the host. The I3C Filter IP supports the I3C protocol. The IP can also be configured as an I2C Filter. In the I2C Filter mode, the IP is a subset of the SMBus protocol.

1.2. Quick Facts

Table 1.1. Summary of the I3C Filter IP

IP Requirements	Supported Devices	MachXO5™-NX
	IP Changes[1]	For a list of changes to the IP, refer to the I3C Filter IP Release Notes (FPGA-RN-02004).
Resource Utilization	Supported User Interface	Advanced High-Performance Bus-Lite (AHB-Lite)
	Resources	Refer to Appendix A. Resource Utilization
Design Tool Support	Lattice Implementation	IP Core v1.7.0 – Lattice Radiant™ Software 2025.2 and Lattice Propel™ Builder Software 2025.2
	Synthesis	Lattice Synthesis Engine (LSE) Synopsys Synplify Pro® for Lattice
	Simulation	For a list of supported simulators, see the Lattice Radiant Software User Guide

Note:

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

1.3. IP Support Summary

Table 1.2. I3C Filter IP Support Readiness

Device Family	IP	System Clock Frequency (MHz)[1]	Supported Feature	Radiant Timing Model	Hardware Validated
MachXO5-NX	I3C Filter	100	Dynamic Address Assignment	Final	Yes
			IBI	Final	Yes
			Hot-Join	Final	Yes
			I3C Write/Read	Final	Yes
			I2C Write/Read	Final	Yes

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.

1.4. Features

Key features of the I3C Filter IP include:

- Provides four interfaces, namely, AHB-Lite, I3C/I2C Bus Controller, I3C/I2C Bus Target, and Interrupt.
- Connects between a single Controller and multiple Target devices.
- Compatible with MIPI I3C Specification v1.1.1 (SDR mode only).
- Supports legacy I2C backward compatibility.
- Allows configurable filter selection to either I3C filter or I2C filter.

- Supports configurable glitch filtering and clock stretching when the I2C filter is selected (not required for the I3C protocol).
- Supports configurable bidirectional filtering when the I2C filter is selected, allowing only one controller on each side of the filter.
- Allows all I3C/I2C read access.
- Verifies all I3C/I2C write access against a list of allowable commands stored in memory. Supports 256 bits for each target device. When a command is not allowed, the write transaction is halted, an interrupt is sent to the host if enabled, and the status register is updated with the blocked command information for the corresponding target address.
- Supports up to 60 lists of allowable commands that can be shared by up to 128 target devices on the bus (7-bit addressing only).
- Allowable commands in the memory can either be initialized (all blocked, or all passed through) or be written by the host (such as CPU, FPGA RoT design, and others) through the AHB-Lite bus.
- Samples the commands from controller with a high frequency system clock before passing it to the target devices.
- Allows all CCCs in I3C standard.

The following features are not supported in this IP:

- I3C HDR mode
- Optional SMBus signals SMBSUS# and SMBALERT# (when I2C mode is used)

1.5. Licensing and Ordering Information

The I3C Filter IP is provided at no additional cost with the Lattice Radiant software.

1.6. Hardware Support

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

1.7. Minimum Device Requirements

There is no limitation in device speed grade for the use of I3C Filter IP. See the maximum clock frequency in Appendix A. Resource Utilization.

1.8. Naming Conventions

1.8.1. Nomenclature

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

1.8.2. Signal Names

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

1.8.3. Attribute Names

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

2. Functional Description

2.1. IP Architecture Overview

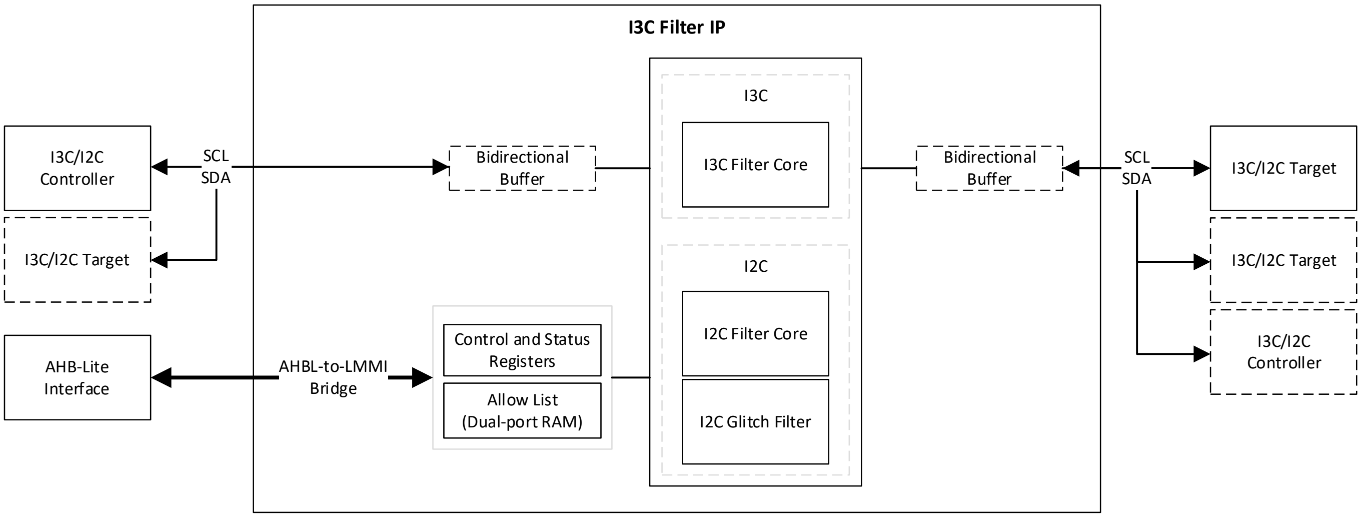


Figure 2.1. I3C Filter Core Block Diagram

The I3C Filter IP includes the following blocks:

- AHB-Lite Interface
- Control and Status Registers
- Memory for allow list
- I3C Filter Core (when I3C filter is enabled)
- I2C Filter Core (when I2C filter is enabled)

2.2. Clocking

There are three clocks for the I3C Filter IP:

- clk_i: System clock, used for drive in the entire IP
- scl_c_io: I3C serial clock connection to I3C Controller, used for sampling SDA
- scl_t_io: I3C serial clock connection to I3C Target

2.2.1. Clocking Overview

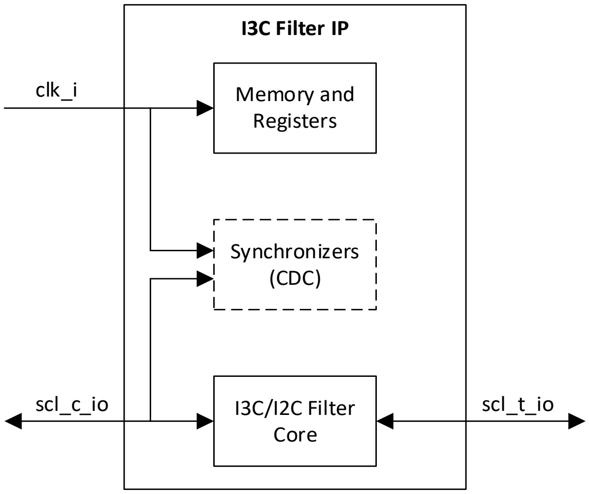


Figure 2.2. I3C Filter IP Clock Domain Block Diagram

Note: The clk_i is selectable from 50 MHz to 125 MHz. There are no restrictions in clk_i frequency to support scl_c_io and scl_t_io frequency up to 12.5 MHz.

2.3. Reset

There is one reset signal, rstn_i, for the I3C Filter IP.

2.3.1. Reset Overview

The rstn_i signal is an asynchronous active low reset. Reset assertion can be asynchronous but reset negation should be synchronous. When asserted, output ports and registers are forced to their reset values.

2.4. User Interfaces

Table 2.1. User Interfaces and Supported Protocols

User Interface	Supported Protocols	Description
Memory-Mapped Interface	AHB-Lite	The AHB-Lite interface is used in this IP for register and memory access. Refer to the Signal Description and Register Description sections for more details. Refer to the AMBA 3 AHB-Lite Protocol Specification for information and timing diagram of the AHB-Lite interface.
Device Receiver/Transmitter Interface	I3C	The I3C Interface can complete the communication between Lattice I3C Filter IP core (when configured as I3C filter) and external I3C Controller and Target devices. Refer to the MIPI I3C Specification for more information on the I3C protocol.
Device Receiver/Transmitter Interface	I2C	The I2C Interface can complete the communication between Lattice I3C Filter IP core (when configured as I2C filter) and external I2C Controller and Target devices. Refer to the I2C Bus Specification and User Manual for more information on the I2C protocol.

2.5. Performing Various I2C/I3C Transactions

The following figures show various I3C and I2C access in non-blocking and blocking mode of the I3C Filter IP.

2.5.1. I2C/I3C Read Transaction

I3C Filter IP allows all read access. All read commands are passed from the Controller to Target through the I3C Filter IP.

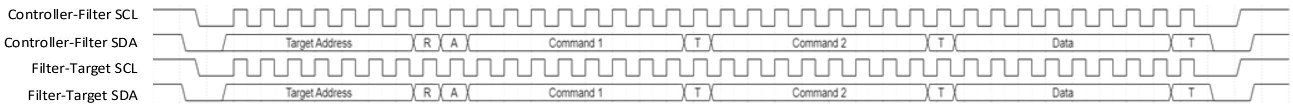


Figure 2.3. I2C/I3C Read Transaction Format 1

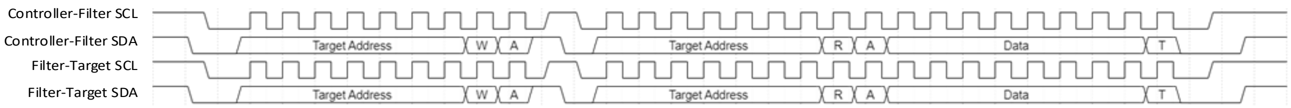


Figure 2.4. I2C/I3C Read Transaction Format 2

2.5.2. I2C/I3C Write Transaction in Non-Blocking Mode

In non-blocking mode, all write access is allowed. All write commands are passed from Controller to Target through the I3C Filter IP.

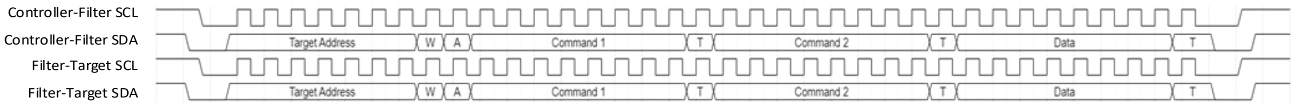


Figure 2.5. I2C/I3C Write Transaction in Non-Blocking Mode

2.5.3. I2C/I3C Write in 1-Byte Blocking Mode

In 1-byte blocking mode, if *Command 1* in below figure is blocked (set Target to 0 in assigned allow list), the I3C Filter IP will generate a STOP condition and block the remaining I3C traffic.

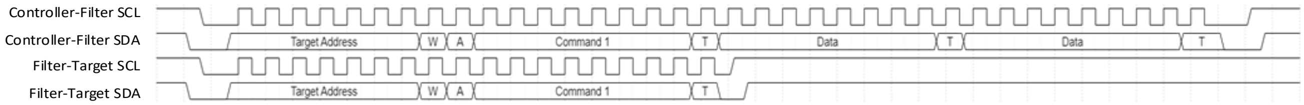


Figure 2.6. I2C/I3C Write in 1-Byte Blocking Mode

This IP also supports the format that starts with Broadcast Address (7'h7E) for both 1-byte and 2-byte Blocking mode.

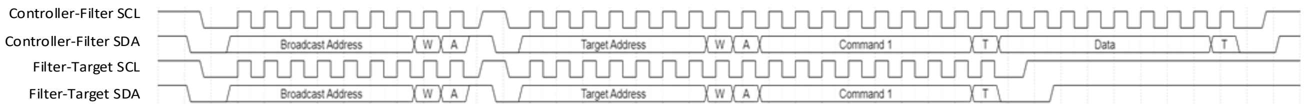


Figure 2.7. I3C Write with Broadcast Address in 1-Byte Blocking Mode

2.5.4. I2C/I3C Write in 2-Byte Blocking Mode

In 2-byte blocking mode, if *Command 1* or *Command 2* in below figure is blocked (set Target to 0 in assigned allow list), the I3C Filter IP will generate a STOP condition and block the remaining I3C traffic.

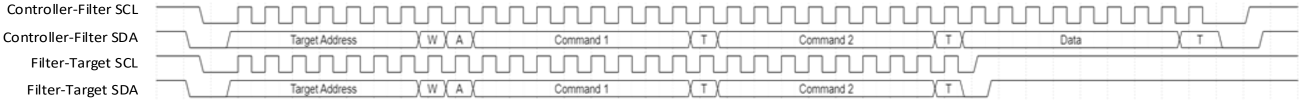


Figure 2.8. I2C/I3C Write in 2-Byte Blocking Mode

Figure 2.8 shows the case where mismatch occurred in *Command 2*. If mismatch occurred in *Command 1*, the I3C Filter IP will immediately generate STOP condition after the command (like in I2C/I3C Write in 1-Byte Blocking Mode).

2.5.5. Other I3C Transactions

All the following I3C transactions are passed through the I3C Filter IP:

- Dynamic address assignment
- All CCCs
- In-band interrupt
- Hot-join request

2.6. I2C Filter Mode

When the *Filter Selection* attribute is set to I2C Filter, the following features are supported:

- Configurable filtering disable
- Optional bidirectional filtering

2.6.1. Filtering Disable

This feature allows temporary disabling of the filtering mechanism through register control. For details on how to configure this, refer to the Filter Configuration (Address 0x1000) section.

- When filtering is enabled, the IP checks incoming I2C write transactions against the configured allow list commands and target assignments. If a transaction matches a blocked command, it is rejected.
- When filtering is disabled, the IP bypasses the allow list check and permits all transactions, including those set as blocked.

2.6.2. Bidirectional Filtering

Bidirectional filtering enables the IP to monitor and filter I2C transactions from a single Controller on each side of the IP interface. This feature is designed for systems where only one Controller is present per side of the IP:

- The IP detects which Controller currently has control of the bus.
- Based on this detection, it applies the filtering rules (allow list and target assignment) to transactions of the active Controller.
- When filtering is enabled, any transaction initiated by the active Controller that matches a blocked command is blocked by the IP.

2.7. I3C Filter IP Flow

Figure 2.9 describes how the I3C Filter IP verifies command received from the Controller and how the IP handles traffic depending on whether the command is allowed or not.

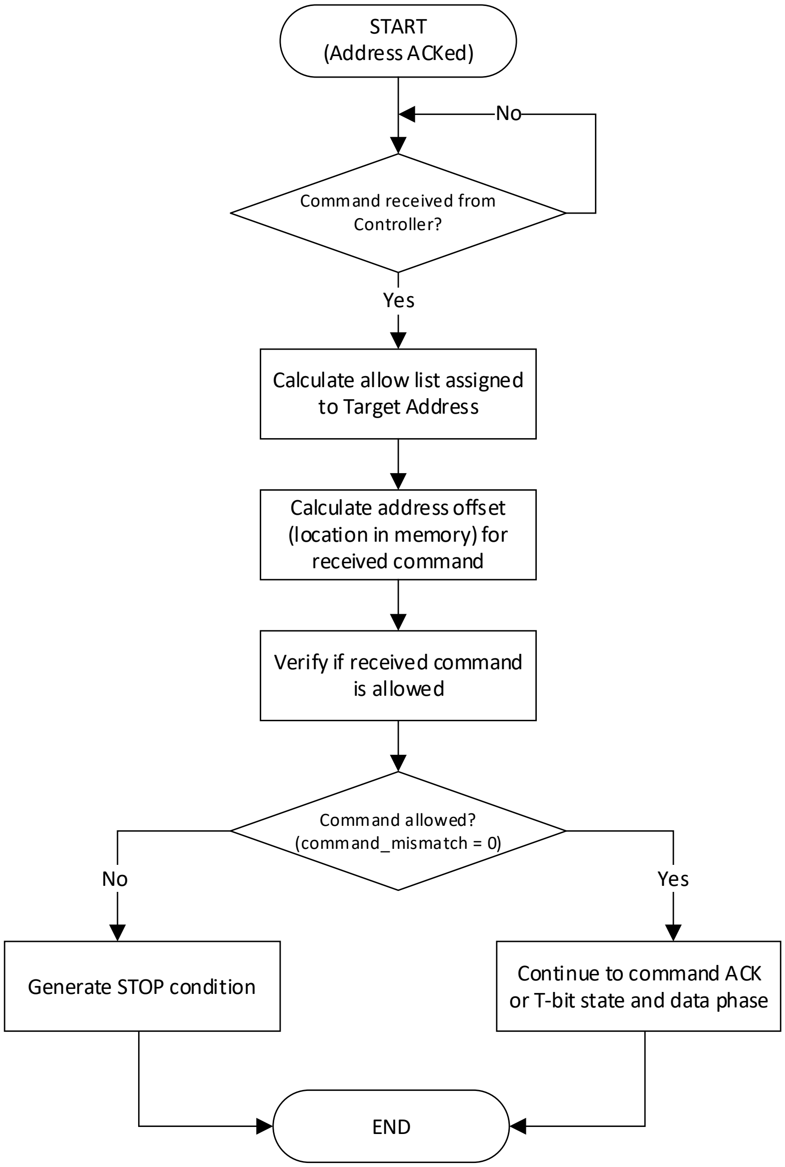

START
(Address ACKed)
No
Command received from Controller?
Yes
Calculate allow list assigned to Target Address
Calculate address offset (location in memory) for received command
Verify if received command is allowed
Command allowed? (command_mismatch = 0)
No
Yes
Generate STOP condition
Continue to command ACK or T-bit state and data phase
END

Figure 2.9. I3C Filter IP Flow

2.8. Programming Flow

This section provides an example on how to configure the IP blocking mode. Consider a command byte from a set of 256 different commands, each represented by a unique hexadecimal value from 8’h00 to 8’hFF. This IP core allows you to choose which commands are allowed or blocked. This can be done by setting up an allow list with 8 different address offsets. Each offset is 32 bits long, and each bit corresponds to a command. If a bit is set to 1, the corresponding command is allowed. Refer to the Register Description section for more information on the register definitions and address offset range.

The steps below describe how to perform 1-byte blocking using target address 7’h34 and Allow List 1.

1. Initialize the allow list for the set of allowed commands by writing to the corresponding address offset.
   The address offset for Allow List 1 is from 0xA0 to 0xBC. In the sample command mapping (shown in binary) in Table 2.2, the following commands are set to 1 in the allow list:

   Commands: 8’h00 - 8’h10 - 8’h39 - 8’h4B - 8’h75 - 8’h88 - 8’h89 - 8’hB2 - 8’hFF

Table 2.2. Allow List 1 Address and Command Mapping

Address Offset	Command Range	[31:24] – 4th Byte	[23:16] – 3rd Byte	[15:8] – 2nd Byte	[7:0] – 1st Byte
0xA0	8’h00 to 8’h1F	0000_0000	**0000_0001**	0000_0000	**0000_0001**
0xA4	8’h20 to 8’h3F	**0000_0010**	0000_0000	0000_0000	0000_0000
0xA8	8’h40 to 8’h5F	0000_0000	0000_0000	**0000_1000**	0000_0000
0xAC	8’h60 to 8’h7F	0000_0000	**0010_0000**	0000_0000	0000_0000
0xB0	8’h80 to 8’h9F	0000_0000	0000_0000	**0000_0011**	0000_0000
0xB4	8’hA0 to 8’hBF	0000_0000	**0000_0100**	0000_0000	0000_0000
0xB8	8’hC0 to 8’hDF	0000_0000	0000_0000	0000_0000	0000_0000
0xBC	8’hE0 to 8’hFF	**1000_0000**	0000_0000	0000_0000	0000_0000

Note: Bits set to 1 indicate which commands are to be passed through. Otherwise, the commands are blocked.

2. Assign allow list to target address by writing the desired allow list number to the corresponding address offset of the Target.

   Example: To assign Allow List 1 to Target Address 7’h34, write 8’h01 to Bits[7:0] of address 0x34.
3. Enable the interrupt for blocked command by writing 32’h1 to Address 0x1014.
4. The IP is now configured to pass the I3C/I2C write with command listed in *Step 1* to Target Address 7’h34.
   Confirm that I3C/I2C write proceeds when Controller sends a Write transaction with command 8’h4B (or any command set to 1 in *Step 2*). SDA/SCL on the Controller side will be equal to the SDA/SCL on the Target side.
5. When Controller sends a Write transaction with command 8’h4C (or any command set to 0 in *Step 2*), the transaction will be blocked. The I3C Filter IP will generate a STOP condition on the SDA/SCL at Target side. Interrupt signal int_o will be asserted. When you read from cmd_blocked_int status register (Address 0x1018), the corresponding status bit will be High. Read from Address 0x1020 for more information about blocked address and command.

Note: There is no need to write to 1byte0_2byte1 register at every step above because this configuration can be set through the GUI. Write to this register when you intend to change the N-byte blocking mode during operation.

2.9. Allow List Configuration

The following procedure describes how to calculate the address offset and bit location of each command per allow list. Refer to the Register Description section for details on register definitions and address offset range.

1. Calculate the address offset based on the intended allow list number and command using the following equation:

   Equation: address_offset = base_address + {allow_list_X, 5’b0} + command[7:3]

   Where,

   base_address – ‘h80 is base address for 1-byte filtering, ‘h800 for 2-byte filtering

   {allow_list_X, 5’d0} – Allow list number from ‘d0 to ‘d59 appended with 5’d0

   command – 8-bit command in HEX (Only bits [7:3] is used in calculation)

   Example:

   - For 1-byte filter, allow list ‘d28, and command ‘h36:

     base_address = ‘h80 → ‘b1000 0000

     {allow_list_X, 5’d0} = ‘d28 → ‘b0001 1100 → append 5’b0 for calculation ‘b0 0011 1000 0000

     command = ‘h36 → ‘b0011 0110 → use only command[7:3] for calculation ‘b0 0110

$$\begin{array}{cc|cccc|cccc|cccc|l}
 & & & & & & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rightarrow \text{base\_address} \\
+ & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rightarrow \{\text{allow\_list\_28, 5'd0}\} \\
+ & & & & & & & & & 0 & 0 & 1 & 1 & 0 & \rightarrow \text{command[7:3]} \\
\hline
 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & \rightarrow \text{address\_offset}
\end{array}$$

   Resulting address offset is 0x406. This is third byte in address offset ‘h404 (see Table 2.3).

2. Identify bit location in the address offset calculated from *Step 1*. Bit location is the last 3 bits of target command. If command to be set is ‘h36, the bit location is command[2:0] equal to ‘h6.
3. For command ‘h36 to be passed through to Target assigned to Allow List 28, write 1 to Bit6 of third byte in address ‘h404. Otherwise, set to 0. Table 2.3 shows Allow List 28 with blocked command ‘h36. The I3C Filter IP will generate STOP condition when Private Write with data equal to ‘h36 is sent to an I3C Target assigned to Allow List 28.

Table 2.3. Allow List 28 Configuration Example

Address Offset	Command Range	4th Byte	3rd Byte	2nd Byte	1st Byte
0x400	8’h00 to 8’h1F	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX
0x404	8’h20 to 8’h3F	XXXX_XXXX	**X0XX_XXXX**	XXXX_XXXX	XXXX_XXXX
0x408	8’h40 to 8’h5F	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX
0x40C	8’h60 to 8’h7F	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX
0x410	8’h80 to 8’h9F	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX
0x414	8’hA0 to 8’hBF	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX
0x418	8’hC0 to 8’hDF	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX
0x41C	8’hE0 to 8’hFF	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX	XXXX_XXXX

3. IP Parameter Description

The configurable attributes of the I3C Filter IP are shown in Table 3.1. You can configure the IP by setting the attributes in the IP Catalog's Module/IP wizard of the Lattice Radiant software accordingly.

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

3.1. General

Table 3.1. General Attributes

Attribute	Selectable Values	Description
General		
Filter Selection	**I3C Filter**, I2C Filter	Indicates if I3C or I2C filter is selected. Note that the I3C filter can still filter I2C traffic given the limitations in I2C support of I3C.
Enable IO Buffer	Checked, **Unchecked**	Indicates if I/O buffers are instantiated for SCL and SDA lines.
Filter Initialization		
Blocking Mode	**Pass Through**, Blocking	Indicates filtering mode after reset. Pass Through – All commands are allowed after reset. Blocking – All commands are blocked after reset.
N-Byte Blocking	**1-byte**, 2-byte	Indicates if IP defaults to 1-byte or 2-byte blocking mode.
I2C Filter Mode		
Enable Clock Stretching	**Checked**, Unchecked	Indicates if clock stretching is enabled in I2C Mode. Selectable only when Filter Selection is I2C Filter.
Enable Glitch Filter	**Checked**, Unchecked	Indicates if glitch filtering is enabled in I2C Mode. Selectable only when Filter Selection is I2C Filter.
Enable I2C SCL Frequency Register	**Checked,** Unchecked	Indicates if I2C Clock Frequency can be set through register or port scl_speed_i. Selectable only when Filter Selection is I2C Filter.
Enable Bidirectional Filtering	Checked**, Unchecked**	Indicates whether IP filtering is applied only in the Controller-to-Target direction or allowed bidirectionally between the Controller and Target. Selectable only when the *Filter Selection* is set to *I2C Filter*.
Clock		
System Clock Frequency (MHz)	50–125	System clock frequency. The default value is 100.

3.2. IP Parameter Settings for Example Use Cases

Table 3.2 shows IP configuration for I3C or I2C filter initialized in non-blocking mode and no additional external logic for SCL and SDA (I/O buffer is used).

Table 3.2. IP Parameter Settings for Example Use Cases

Target Use Case	I3C Controller (Pure/Mixed Fast Bus/Mixed Slow Bus)	I2C Controller (Targets with True 50 ns Spike Filter)	I2C Controller (Targets without True 50 ns Spike Filter)
General			
Filter Selection	I3C	I2C	I2C
Enable IO Buffer	Checked	Checked	Checked
Filter Initialization			
Filter Initial Mode	Pass Through	Pass Through	Pass Through
N-Byte Blocking	1-byte	1-byte	1-byte
I2C Filter Mode			
Enable Clock Stretching	—	Checked	Checked
Enable Glitch Filter	—	Unchecked	Checked
Enable I2C SCL Frequency Register	—	Checked	Checked
Enable Bidirectional Filtering	—	Unchecked	Checked
Clock			
System Clock Frequency (MHz)	100	100	100

4. Signal Description

Table 4.1 describes the I3C Filter IP ports.

Table 4.1. Signal Description

Signal Name	Direction	Width	Description
Clock and Reset			
clk_i	Input	1	User Clock Input. Refer to the Clocking Overview section for more details.
rstn_i	Input	1	Asynchronous active low reset
scl_speed_i[1]	Input	2	I2C Clock Frequency to be sent to Target devices: 2'd0, 2'd1 – 100 kHz (default) 2'd2 – 400 kHz 2'd3 – 1000 kHz
Interrupt			
int_o	Output	1	Interrupt signal. Reset value is 1'b0.
I3C Signals (Enable IO Buffer = 1)[2]			
scl_c_io	Input/Output	1	I3C/I2C serial clock from Controller
sda_c_io	Input/Output	1	I3C/I2C serial data from Controller
scl_t_io	Input/Output	1	I3C/I2C serial clock to Target
sda_t_io	Input/Output	1	I3C/I2C serial data to Target
I3C Signals (Enable IO Buffer = 0)[2]			
scl_c_i	Input	1	I3C/I2C serial clock input from Controller
scl_c_oe_o	Output	1	I3C/I2C serial clock active high output enable for Controller
scl_c_o	Output	1	I3C/I2C serial clock output for Controller
sda_c_i	Input	1	I3C/I2C serial data input from Controller
sda_c_oe_o	Output	1	I3C/I2C serial data active high output enable for Controller
sda_c_o	Output	1	I3C/I2C serial data output for Controller
scl_t_i	Input	1	I3C/I2C serial clock input from Target
scl_t_oe_o	Output	1	I3C/I2C serial clock active high output enable for Target
scl_t_o	Output	1	I3C/I2C serial clock output for Target
sda_t_i	Input	1	I3C/I2C serial data input from Target
sda_t_oe_o	Output	1	I3C/I2C serial data active high output enable for Target
sda_t_o	Output	1	I3C/I2C serial data output for Target
AHB-Lite Interface			
ahbl_hsel_i	Input	1	AHB-Lite Select signal. Indicates the device is selected and transfer is required.
ahbl_hready_i	Input	1	AHB-Lite Ready Input signal. Indicates data phase of previous transfer is completed.
ahbl_haddr_i	Input	32	AHB-Lite Address signal
ahbl_hburst_i	Input	3	AHB-Lite Burst Type signal. Indicates if the transfer is a single transfer or forms part of a burst. Only 3'd0 is supported.
ahbl_hsize_i	Input	3	AHB-Lite Transfer Size signal. Indicates the size of the transfer that is a byte, halfword or word. Only 3'd2 is supported.
ahbl_hmastlock_i	Input	1	AHB-Lite Lock signal. This signal is unused.
ahbl_hprot_i	Input	4	AHB-Lite Protection Control signal. This signal is unused.
ahbl_htrans_i	Input	2	AHB-Lite Transfer Type signal. Indicates the transfer type of the current transfer.
ahbl_hwrite_i	Input	1	AHB-Lite Direction signal. Write = High, Read = Low.
ahbl_hwdata_i	Input	32	AHB-Lite Write Data signal.
ahbl_hreadyout_o	Output	1	AHB-Lite Ready Output signal. Indicates transfer completion. Reset value is 1'b1.
ahbl_hrdata_o	Output	32	AHB-Lite Read Data signal. Reset value is 0.

Signal Name	Direction	Width	Description
ahbl_hresp_o	Output	1	AHB-Lite Transfer Response signal. This signal is tied to 1’b0.

Notes:

1. The scl_speed_i signal is only applicable when I2C filter is selected and the *Enable I2C SCL Frequency Register* parameter is unchecked.
2. SCL and SDA ports depend on the *Enable IO Buffer* parameter. When checked, SCL and SDA signals are passed through bidirectional buffers in the internal logic before connecting to the IP ports. When unchecked, the control signals for bidirectional buffers are exposed at the IP ports to enable additional logic for SCL and SDA lines outside the IP core.

5. Register Description

This section defines the configuration, control, and status registers of the I3C Filter IP. The total address space of the IP is 8 KiB.

Table 5.1 lists the register access types. Table 5.2 shows the mapping of registers and register groups to their addresses. These registers are accessible through the user interface (AHB-Lite).

Table 5.1. Register Access Types

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

Table 5.2. I3C Filter IP Register Summary

Address Offset/Range[1]	Register Name	Description	Access Type	Reset
0x00–0x7C	allow_list_addrN	Assigned commands allow list for Target Address *N*, where *N* ranges from 0 to 127 (supporting up to 128 Target devices). Each Target Address has a dedicated allow list field.	RW	0x0
0x80–0x7FC	allow_list1b_X_ZtoY	Defines bits Z to Y of allow list X for 1-byte command filtering. When 1-byte filtering is enabled, this field represents the full 1-byte command. When 2-byte filtering is enabled, this field represents the first byte of the command. Address Offsets: • 0x80: allow_list1b_0_31to0 – Bits 31 to 0 of allow list 0. • 0x84: allow_list1b_0_63to32 – Bits 63 to 32 of allow list 0.[2] • ... • Up to 0x7FC for the full range of allow list entries.[3]	RW	0x0
0x800–0xF7C	allow_list2b_X_ZtoY	Defines bits Z to Y of allow list X for 2-byte command filtering. When 2-byte filtering is enabled, this field represents the second byte of the command. Address Offsets: • 0x800: allow_list1=2b_0_31to0 – Bits 31 to 0 of allow list 0. • 0x804: allow_list2b_0_63to32 – Bits 63 to 32 of allow list 0.[2] • ... • Up to 0xF7C for the full range of allow list entries.[3]	RW	0x0
0xF80–0xFFC	reserved	Reserved address	RO	0x0
0x1000	filter_cfg	Indicates filter configuration signals	RW	0x0
0x1004–0x1010	i3c0_i2c1	Indicates if device is I3C or I2C	RW	0x0
0x1014	int_en	Interrupt enable signals	RW	0x0
0x1018	int_stat	Interrupt status signals	RW1C	0x0
0x101C	int_set	Interrupt set signals	WO	0x0
0x1020	blocked_cmd_info	Information register for blocked commands	RO	0x0
0x1024	bus_status	Information register for bus status	RO	0x1
0x1028–0x1FFC	reserved	Reserved address	RO	0x0

Notes:

1. Address offset starts at 0x0 and increments by 4.
2. Each allow list occupies 8 address offsets, storing up to 256 bits.
3. The number of allow lists X ranges from 0 to 59.

5.1. Target Device Allow List Assignment (Address 0x0 to 0x7C)

Target Device Allow List Assignment (Address 0x0).

Table 5.3. Target Device Allow List Assignment

Field	Name	Description	Access	Reset
[31:24]	allow_list_addr3	Assigned commands allow list for Target Address 7'h3	RW	8'h0
[23:16]	allow_list_addr2	Assigned commands allow list for Target Address 7'h2	RW	8'h0
[15:8]	allow_list_addr1	Assigned commands allow list for Target Address 7'h1	RW	8'h0
[7:0]	allow_list_addr0	Assigned commands allow list for Target Address 7'h0	RW	8'h0

Notes:

- Address 0x4 is for Target Address 7'h4 to 7'h7.
- Address 0x8 is for Target Address 7'h8 to 7'hB.
- Same mapping is used up to Address 0x7C for Target Addresses 7'hC to 7'h7F.

5.2. 1-Byte Filtering Allow List 0 (Address 0x80 to 0x9C)

Mapping of 256-bit value for allow list 0 of 1-byte.

Table 5.4. 1-Byte Filtering Allow List 0

Address Offset	Name	Description	Access	Reset
0x80	allow_list1b_0_31to0	Bits [31:0] of commands in Allow List 0	RW	Depending on *Blocking Mode* parameter setting in GUI: • 32'h0000_0000 if 1-byte Blocking. • 32'hFFFF_FFFF if Pass Through or 2-byte Blocking.
0x84	allow_list1b_0_63to32	Bits [63:32] of commands in Allow List 0	RW	
0x88	allow_list1b_0_95to64	Bits [95:64] of commands in Allow List 0	RW	
0x8C	allow_list1b_0_127to96	Bits [127:96] of commands in Allow List 0	RW	
0x90	allow_list1b_0_159to128	Bits [159:128] of commands in Allow List 0	RW	
0x94	allow_list1b_0_191to160	Bits [191:160] of commands in Allow List 0	RW	
0x98	allow_list1b_0_216to192	Bits [216:192] of commands in Allow List 0	RW	
0x9C	allow_list1b_0_255to224	Bits [255:224] of commands in Allow List 0	RW	

Notes:

- Address offset for allow_list1b_1 is from 0xA0 to 0xBC.
- Address offset for allow_list1b_2 is from 0xC0 to 0xDC.
- Same mapping is used up to allow_list1b_59 with offset 0x7E0 to 0x7FC.

If a bit is set to 1, the command is passed through from Controller.

If a bit is set to 0, the command is filtered and blocked from Controller.

For example, if bit 54 in allow list 0 (bit 22 of address 0x84) is set to 1, transaction from Controller with command byte equal to 8'd54 will be passed through to Target with allow list assigned to 0.

5.3. 2-Byte Filtering Allow List 0 (Address 0x800 to 0x81C)

Mapping of 256-bit value for allow list 0 of 2-byte filter.

Table 5.5. 2-Byte Filtering Allow List 0

Address Offset	Name	Description	Access	Reset
0x800	allow_list2b_0_31to0	Bits [31:0] of commands in Allow List 0	RW	Depending on *Blocking Mode* parameter setting in GUI: • 32'h0000_0000 if 2-byte Blocking. • 32'hFFFF_FFFF if Pass Through or 1-byte
0x804	allow_list2b_0_63to32	Bits [63:32] of commands in Allow List 0	RW	
0x808	allow_list2b_0_95to64	Bits [95:64] of commands in Allow List 0	RW	
0x80C	allow_list2b_0_127to96	Bits [127:96] of commands in Allow List 0	RW	
0x810	allow_list2b_0_159to128	Bits [159:128] of commands in Allow List 0	RW	
0x814	allow_list2b_0_191to160	Bits [191:160] of commands in Allow List 0	RW	

Address Offset	Name	Description	Access	Reset
0x818	allow_list2b_0_216to192	Bits [216:192] of commands in Allow List 0	RW	Blocking.
0x81C	allow_list2b_0_255to224	Bits [255:224] of commands in Allow List 0	RW	

Notes:

- Address offset for allow_list2b_1 is from 0x820 to 0x83C.
- Address offset for allow_list2b_2 is from 0x840 to 0x85C.
- Same mapping is used up to allow_list2b_59 with offset 0xF60 to 0xF7C.

If a bit is set to 1, the command is passed through from Controller.

If a bit is set to 0, the command is filtered and blocked from Controller.

For example, if:

- bit 54 in 1-byte filter allow list 0 (bit 22 of address 0x84) is set to 1; and
- bit 8 in 2-byte filter allow list (bit 8 of address 0x800) is set to 1,

transaction from Controller with command bytes equal to 8’d54 for first command or 8’d8 for second command will be passed through to Target with allow list assigned to 0.

If bit 8 in 2-byte filter allow list (bit 8 of address 0x800) is set to 0 instead and the same transaction is sent from Controller, the first command will be passed through, and the I3C Filter IP will generate a STOP condition after the second command.

5.4. Filter Configuration (Address 0x1000)

Filter configuration for blocking mode and I2C SCL frequency.

Table 5.6. Filter Configuration

Field	Name	Description	Access	Reset
[31:10]	reserved	Reserved	RO	22’h0
[9:8]	i2c_freq	I2C SCL Frequency. Applicable only when I2C Filter is selected. Fixed to 0 if I3C Filter is selected or I2C_SCL_FREQ_REG_EN is disabled: • 2’d0, 2’d1 – 100 kHz • 2’d2 – 400 kHz • 2’d3 – 1 MHz In case of multiple targets, set this register to the speed of the slowest device.	RW	2’h0
[7:2]	reserved	Reserved	RO	6’h0
[1]	filter_dis	Indicates whether filtering is enabled or disabled for configured blocked commands. Applicable only when the I2C Filter is selected: • 1’b0 – Filtering enabled • 1’b1 – Filtering disabled Set this register only when the bus is idle. If set while the bus is active, the filtering behavior is not guaranteed. Refer to the Bus Status (0x1024) section to check the bus idle status.	RW	1’b0
[0]	1byte0_2byte1	Indicates if blocking mode is 1-byte or 2-byte: • 1’b0 – 1-byte blocking • 1’b1 – 2-byte blocking	RW	Depending on *N-Byte Blocking* parameter setting in GUI: • 1’b0 – 1-byte blocking • 1’b1 – 2-byte blocking

5.5. Target Device Support (Address 0x1004 to 0x1010)

Mapping of I3C or I2C device support based on Target Address. I3C Filter IP supports 128 target addresses. Applicable only when *I3C Filter* is selected.

Table 5.7. Target Device Support

Address Offset	Name	Description	Access	Reset
0x1004	i3c0_i2c1_target_31to0	Indicates if target 31 to 0 is I3C or I2C device	RW	32’h0
0x1008	i3c0_i2c1_target_63to32	Indicates if target 63 to 32 is I3C or I2C device	RW	32’h0
0x100C	i3c0_i2c1_target_95to64	Indicates if target 95 to 64 is I3C or I2C device	RW	32’h0
0x1010	i3c0_i2c1_target_127to96	Indicates if target 127 to 96 is I3C or I2C device	RW	32’h0

0 – I3C device

1 – I2C device

For example, if bit 63 is set to 1 (most significant bit of address 0x1008), then Target Device 63 is an I2C device.

5.6. Interrupt Enable (Address 0x1014)

Interrupt Enable. When set to high, it enables the corresponding interrupt status signal to cause the assertion of interrupt port signal (int_o).

Table 5.8. Interrupt Enable

Field	Name	Description	Access	Reset
[31:1]	reserved	Reserved	RO	31’h0
[0]	cmd_blocked_int_en	Enable for blocked command interrupt	RW	1’h0

5.7. Interrupt Status (Address 0x1018)

Interrupt Status. When high, it indicates an event and when the corresponding interrupt enable bit is enabled, it will cause the assertion of interrupt port (int_o).

Table 5.9. Interrupt Status

Field	Name	Description	Access	Reset
[31:1]	reserved	Reserved	RO	31’h0
[0]	cmd_blocked_int	Blocked command interrupt. Write 1 to clear status.	RW1C	1’h0

5.8. Interrupt Set (Address 0x101C)

Interrupt Set. This is a dummy register that is used to test the assertion of interrupt status. When set to high, it triggers the corresponding interrupt status signal.

Table 5.10. Interrupt Set

Field	Name	Description	Access	Reset
[31:1]	reserved	Reserved	RO	31’h0
[0]	cmd_blocked_int_set	Manually set blocked command interrupt	WO	1’h0

5.9. Blocked Command Information (0x1020)

Information register for blocked command. Contains corresponding target address and command when blocked command interrupt is asserted.

Table 5.11. Blocked Command Information

Field	Name	Description	Access	Reset
[31]	blocked_addr_valid	Indicates if blocked address information is valid	RO	1'h0
[30]	blocked_cmd1_valid	Indicates if blocked first command information is valid	RO	1'h0
[29]	blocked_cmd2_valid	Indicates if blocked second command information is valid	RO	1'h0
[28:23]	reserved	Reserved	RO	6'h0
[22:16]	blocked_addr	Blocked address	RO	7'h0
[15:8]	blocked_cmd1	Blocked first command	RO	8'h0
[7:0]	blocked_cmd2	Blocked second command	RO	8'h0

If interrupt for blocked command cmd_blocked_int is asserted, you can read from this register for more information.

If blocked_addr_valid is High, blocked_addr register contains the address of the Target with the last blocked command.

If blocked_cmd1_valid is High, blocked_cmd1 register contains the first byte of the last blocked command.

If blocked_cmd2_valid is High, blocked_cmd2 register contains the second byte of the last blocked command.

For 1-byte blocking, only blocked_addr_valid and blocked_cmd1_valid registers are expected to be asserted when a blocked command event occurs.

For 2-byte blocking, blocked_addr_valid is expected to be asserted when a blocked command event occurs. Depending on which command byte triggered the blocked command event, either blocked_cmd1_valid or blocked_cmd2_valid register is expected to be asserted.

5.10. Bus Status (0x1024)

Information Register for Bus Status.

Table 5.12. Bus Status

Field	Name	Description	Access	Reset
[31:1]	Reserved	Reserved	RO	31'h0
[0]	bus_idle	Indicates if the I3C bus is Idle: • 1'b1 – Bus is Idle • 1'b0 – Bus is Active	RO	1'h1

6. Example Design

The I3C Filter example design allows you to compile, simulate, and test the I3C Filter IP on the MachXO5-NX Development Board.

6.1. Example Design Supported Configuration

Table 6.1. I3C Filter IP Configuration Supported by the Example Design

I3C Filter IP GUI Parameter	I3C Filter IP Configuration Supported in Example Demo Design
General	
Filter Selection	I3C
Enable IO Buffer	Checked
Filter Initialization	
Blocking Mode	Blocking
N-Byte Blocking	2-byte
I2C Filter Mode	
Enable Clock Stretching	—
Enable Glitch Filter	—
Enable I2C SCL Frequency Register	—
Enable Bidirectional Filtering	—
Clock	
System Clock Frequency (MHz)	100 MHz

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

I3C Controller, I3C Target, and I3C Filter IPs are instantiated and connected in the project as shown in Figure 6.1. In this example, I3C Controller and I3C Target IP are instantiated in the same system with I3C Filter IP.

Refer to the I3C Controller IP User Guide (FPGA-IPUG-02228) and I3C Target IP User Guide (FPGA-IPUG-02227) to generate the respective IP. In actual hardware or use case, you can connect the I3C Filter to different I3C Controller and I3C/I2C Target devices.

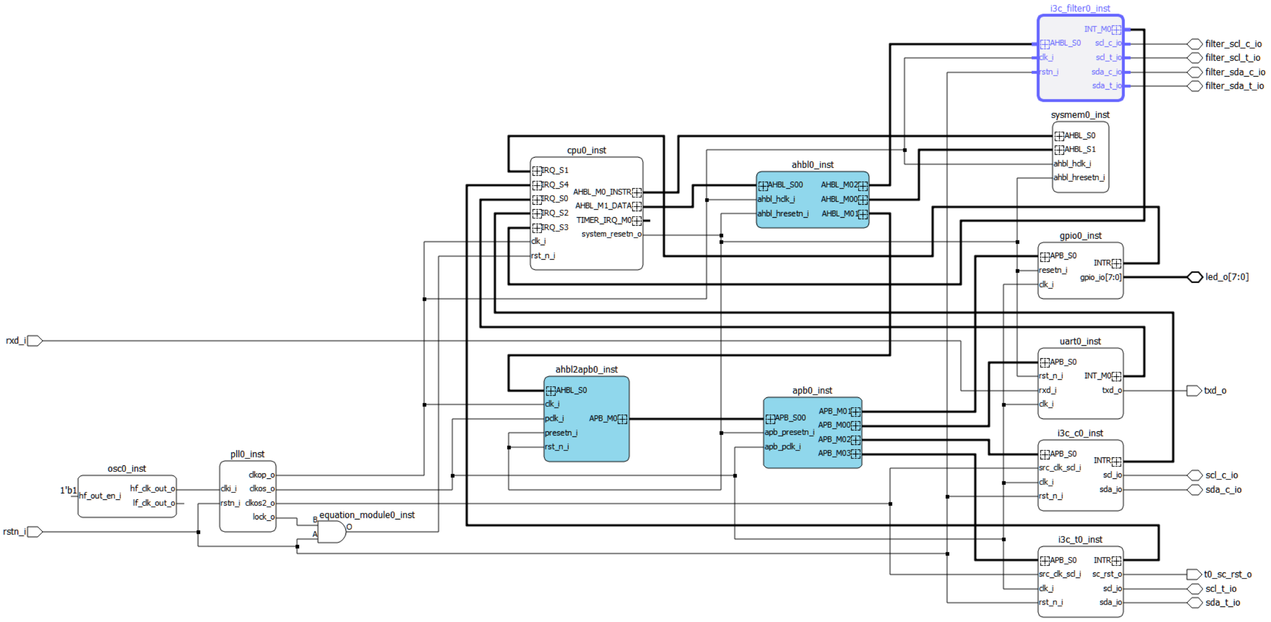
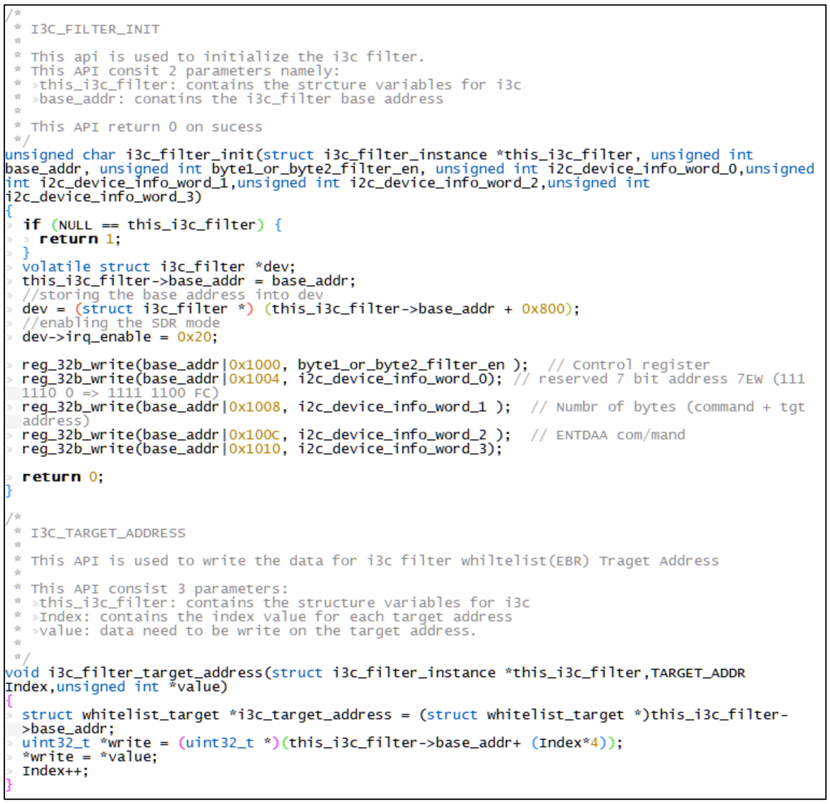


Figure 6.1. I3C Filter IP in Propel SoC Project

An Embedded C/C++ Project is also created in the Propel software to enable developing and debugging application code for different IP features. I3C Filter features can be tested by sending I3C Commands from the I3C Controller to the I3C Target. Runtime configuration of IP and feature testing can be done through C-Code Test Routine. Figure 6.2 shows an example routine for initializing the filter and setting the allow list for each Target.



```
/*
 * I3C_FILTER_INIT
 *
 * This api is used to initialize the i3c filter.
 * This API consit 2 parameters namely:
 *  this_i3c_filter: contains the strcture variables for i3c
 *  base_addr: conatins the i3c_filter base address
 *
 * This API return 0 on sucess
 */
unsigned char i3c_filter_init(struct i3c_filter_instance *this_i3c_filter, unsigned int
base_addr, unsigned int byte1_or_byte2_filter_en, unsigned int i2c_device_info_word_0,unsigned
int i2c_device_info_word_1,unsigned int i2c_device_info_word_2,unsigned int
i2c_device_info_word_3)
{
  if (NULL == this_i3c_filter) {
    return 1;
  }
  volatile struct i3c_filter *dev;
  this_i3c_filter->base_addr = base_addr;
  //storing the base address into dev
  dev = (struct i3c_filter *) (this_i3c_filter->base_addr + 0x800);
  //enabling the SDR mode
  dev->irq_enable = 0x20;

  reg_32b_write(base_addr|0x1000, byte1_or_byte2_filter_en );  // Control register
  reg_32b_write(base_addr|0x1004, i2c_device_info_word_0); // reserved 7 bit address 7EW (111
  1110 0 => 1111 1100 FC)
  reg_32b_write(base_addr|0x1008, i2c_device_info_word_1 );  // Numbr of bytes (command + tgt
  address)
  reg_32b_write(base_addr|0x100C, i2c_device_info_word_2 );  // ENTDAA com/mand
  reg_32b_write(base_addr|0x1010, i2c_device_info_word_3);

  return 0;
}

/*
 * I3C_TARGET_ADDRESS
 *
 * This API is used to write the data for i3c filter whiltelist(EBR) Traget Address
 *
 * This API consist 3 parameters:
 *  this_i3c_filter: contains the structure variables for i3c
 *  Index: contains the index value for each target address
 *  value: data need to be write on the target address.
 *
 */
void i3c_filter_target_address(struct i3c_filter_instance *this_i3c_filter,TARGET_ADDR
Index,unsigned int *value)
{
  struct whitelist_target *i3c_target_address = (struct whitelist_target *)this_i3c_filter-
  >base_addr;
  uint32_t *write = (uint32_t *)(this_i3c_filter->base_addr+ (Index*4));
  *write = *value;
  Index++;
}
```

Figure 6.2. Sample C-Code Test Routine

6.3. Example Design Components

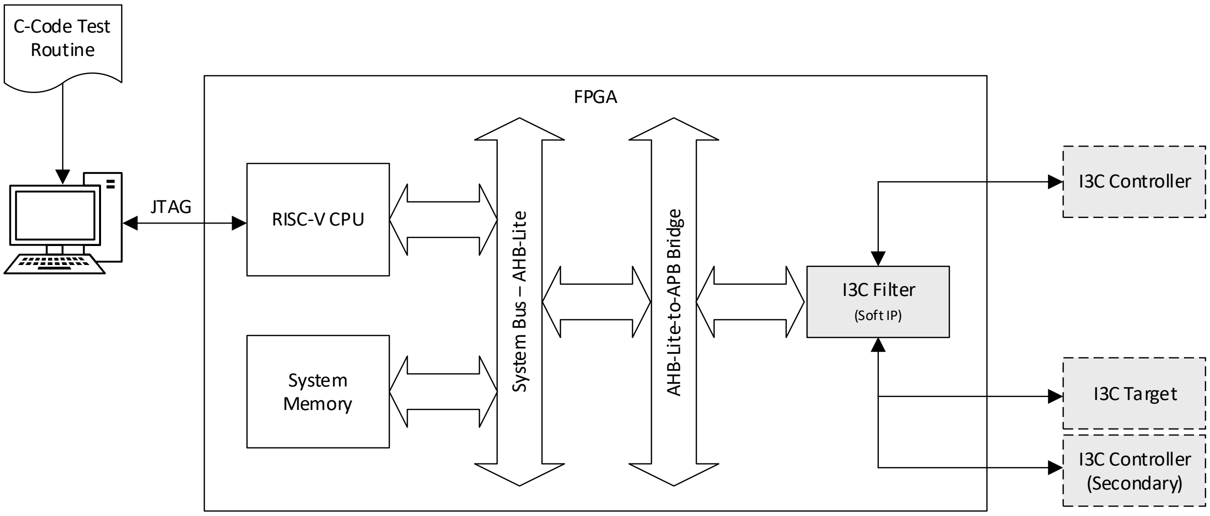


Figure 6.3. I3C Filter Example Design Block Diagram

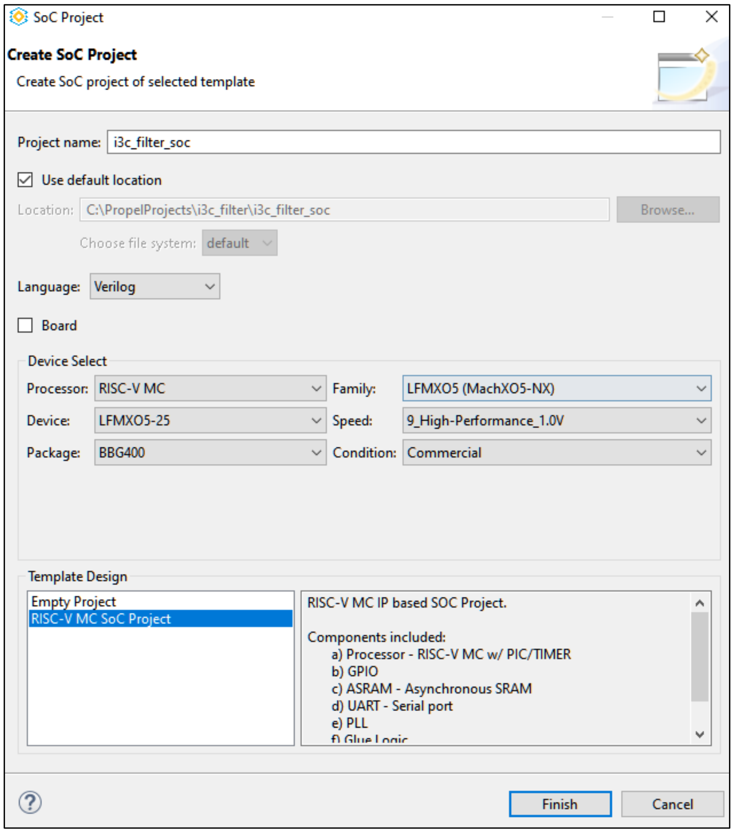
The I3C Filter example design includes the following blocks:

- 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
- I3C Filter IP – IP instance connected to I3C Bus (SCL and SDA)
- I3C Controller/Target Devices

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 Propel Software, create a new Lattice SoC Design Project by navigating to **File** > **New** > **Lattice SoC Design Project**.
3. The **Create SC Project** window will open:
   - In **Device Select** section, indicate the correct details of the device or board that you will use. 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.



SoC Project

Create SoC Project

Create SoC project of selected template

Project name: i3c_filter_soc

☑ Use default location

Location: C:\PropelProjects\i3c_filter\i3c_filter_soc

Choose file system: default

Language: Verilog

☐ Board

Device Select

Processor:	RISC-V MC	Family:	LFMXO5 (MachXO5-NX)
Device:	LFMXO5-25	Speed:	9_High-Performance_1.0V
Package:	BBG400	Condition:	Commercial

Template Design

Empty Project
RISC-V MC SoC Project

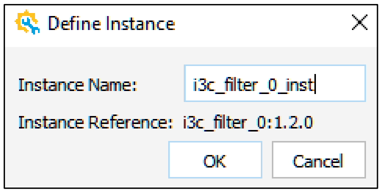
RISC-V MC IP based SOC Project.

Components included:
a) Processor - RISC-V MC w/ PIC/TIMER
b) GPIO
c) ASRAM - Asynchronous SRAM
d) UART - Serial port
e) PLL
f) Glue Logic

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 following IPs:
   - I3C Controller
   - I3C Target/I3C Secondary Controller
   - I3C Filter

   Refer to the Generating and Instantiating the IP section for more details.
6. After generating each IP, the **Define Instance** window will open. Modify instance name if needed, then click **OK**.



Define Instance

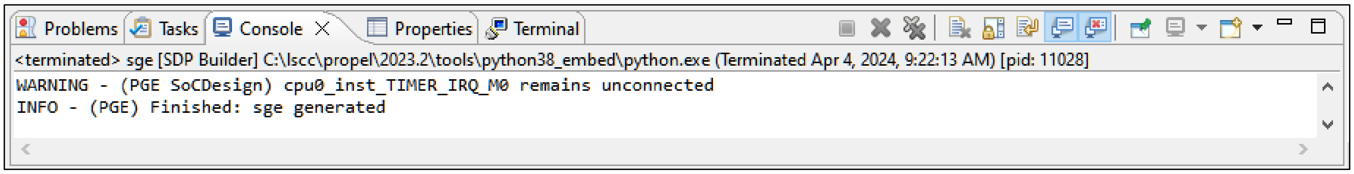
Instance Name: i3c_filter_0_inst

Instance Reference: i3c_filter_0:1.2.0

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**.

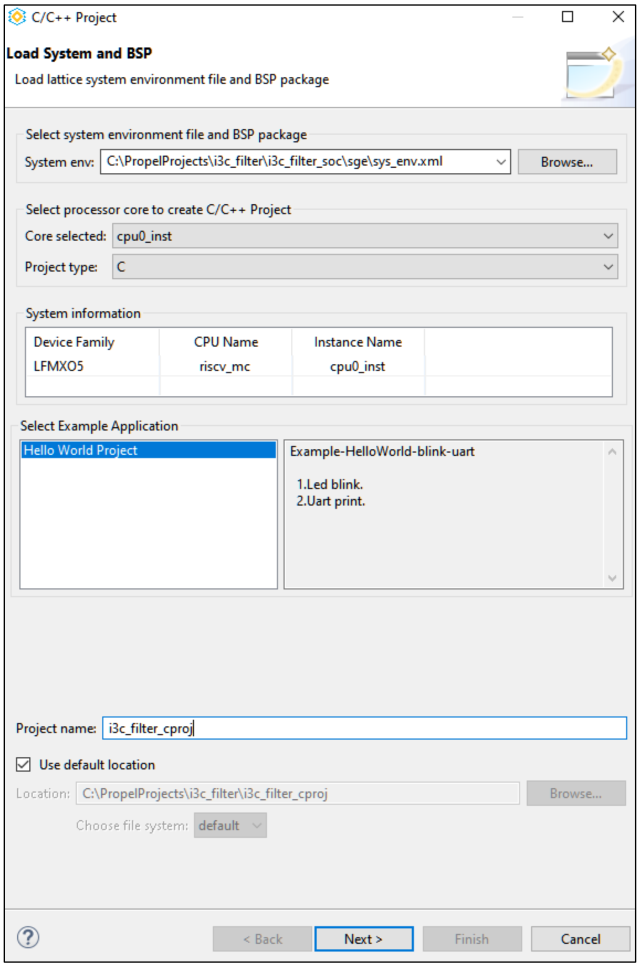
11. Check the build result from the **Console** view.



```
<terminated> sge [SDP Builder] C:\lscc\propel\2023.2\tools\python38_embed\python.exe (Terminated Apr 4, 2024, 9:22:13 AM) [pid: 11028]
WARNING - (PGE SoCDesign) cpu0_inst_TIMER_IRQ_M0 remains unconnected
INFO - (PGE) Finished: sge generated
```

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**, then click **Finish**.



Load System and BSP

Load lattice system environment file and BSP package

Select system environment file and BSP package

System env: C:\PropelProjects\i3c_filter\i3c_filter_soc\sge\sys_env.xml

Select processor core to create C/C++ Project

Core selected: cpu0_inst

Project type: C

System information

Device Family	CPU Name	Instance Name
LFMXO5	riscv_mc	cpu0_inst

Select Example Application

Hello World Project

Example-HelloWorld-blink-uart

1.Led blink.
2.Uart print.

Project name: i3c_filter_cproj

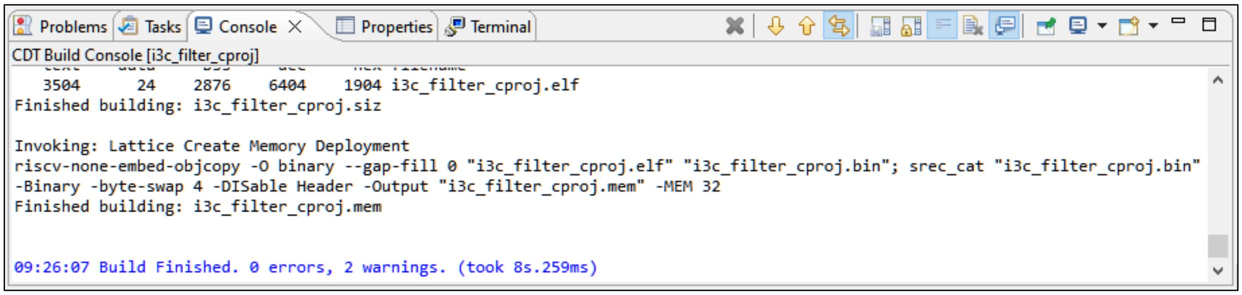
☑ Use default location

Location: C:\PropelProjects\i3c_filter\i3c_filter_cproj

Choose file system: default

Figure 6.7. Lattice C/C++ Design Project

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



```
   3504      24    2876    6404    1904 i3c_filter_cproj.elf
Finished building: i3c_filter_cproj.siz

Invoking: Lattice Create Memory Deployment
riscv-none-embed-objcopy -O binary --gap-fill 0 "i3c_filter_cproj.elf" "i3c_filter_cproj.bin"; srec_cat "i3c_filter_cproj.bin"
-Binary -byte-swap 4 -DISable Header -Output "i3c_filter_cproj.mem" -MEM 32
Finished building: i3c_filter_cproj.mem


09:26:07 Build Finished. 0 errors, 2 warnings. (took 8s.259ms)
```

Figure 6.8. Build C/C++ Project Result

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

6.5. Simulating the Example Design

Refer to the System Simulation Flow section of the Lattice Propel SDK User Guide.

6.6. Hardware Testing

6.6.1. Hardware Testing Setup

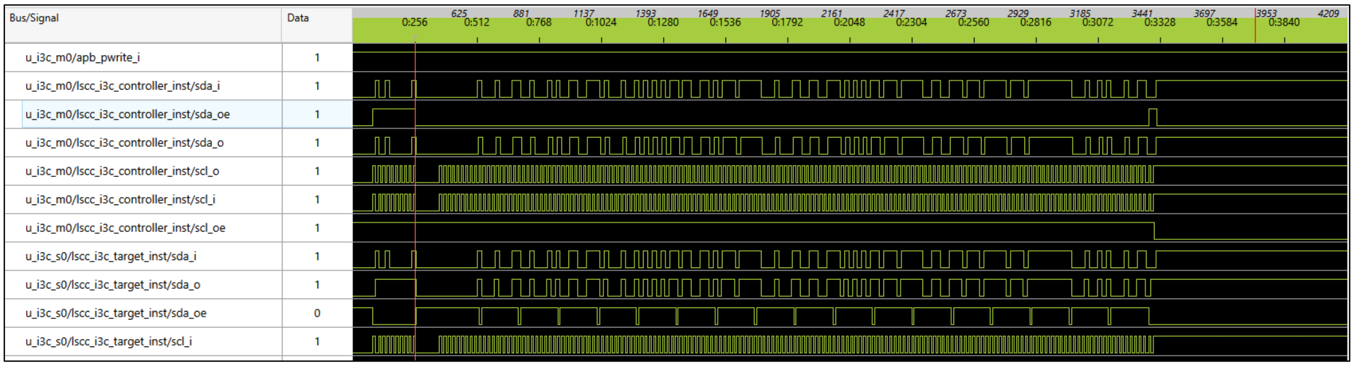
Download the generated bitstream file from the Generating the Example Design section to the MachXO5-NX Development Board through the Lattice Radiant Programmer.

Before running tests, ensure the following connections are properly configured:

- Connect the I3C Filter to external I3C/I2C Target or Secondary Controller devices using flywire.
- Verify the correct pin mapping between the controller and the external device to avoid communication errors or hardware damage.
- Keep wire lengths short to maintain signal integrity and minimize noise or crosstalk.
- Add pull-up resistors to the SDA and SCL lines if required by your I3C/I2C configuration.

6.6.2. Expected Output

Below is a sample waveform captured via Reveal Inserter and Reveal Analyzer tools. Refer to the relevant sections in the Lattice Radiant Software User Guide for more information on how to use the Reveal Inserter and Reveal Analyzer tools.



Bus/Signal	Data
u_i3c_m0/apb_pwrite_i	1
u_i3c_m0/lscc_i3c_controller_inst/sda_i	1
u_i3c_m0/lscc_i3c_controller_inst/sda_oe	1
u_i3c_m0/lscc_i3c_controller_inst/sda_o	1
u_i3c_m0/lscc_i3c_controller_inst/scl_o	1
u_i3c_m0/lscc_i3c_controller_inst/scl_i	1
u_i3c_m0/lscc_i3c_controller_inst/scl_oe	1
u_i3c_s0/lscc_i3c_target_inst/sda_i	1
u_i3c_s0/lscc_i3c_target_inst/sda_o	1
u_i3c_s0/lscc_i3c_target_inst/sda_oe	0
u_i3c_s0/lscc_i3c_target_inst/scl_i	1

Figure 6.9 Expected Output for I3C Private Read

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.

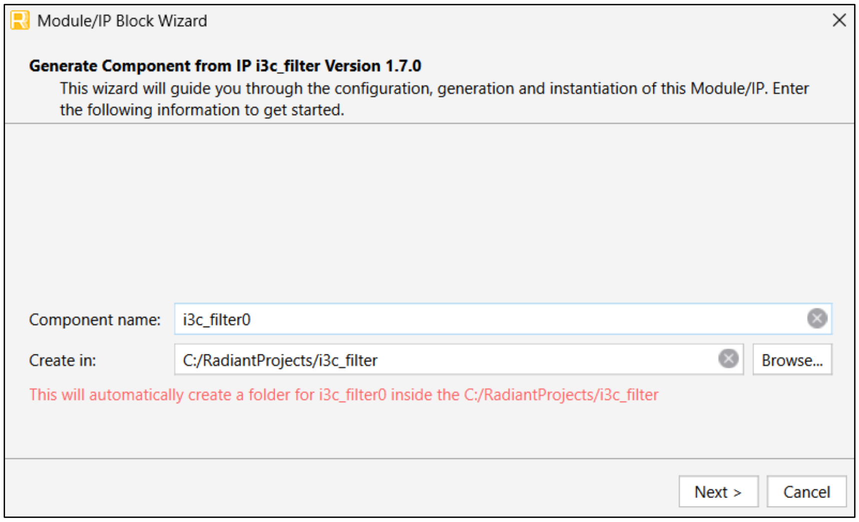
Note: The screenshots provided are for reference only. Details may vary depending on the version of the IP or software being used. If there have been no significant changes to the GUI, a screenshot may reflect an earlier version of the IP.

7.1. Generating and Instantiating the IP

You can use the Lattice Radiant software to generate IP modules and integrate them into the device architecture. The steps below describe how to generate the I3C Filter IP in the Lattice Radiant software.

To generate the I3C Filter IP:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click **I3C Filter** 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**.



Module/IP Block Wizard

Generate Component from IP i3c_filter Version 1.7.0

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

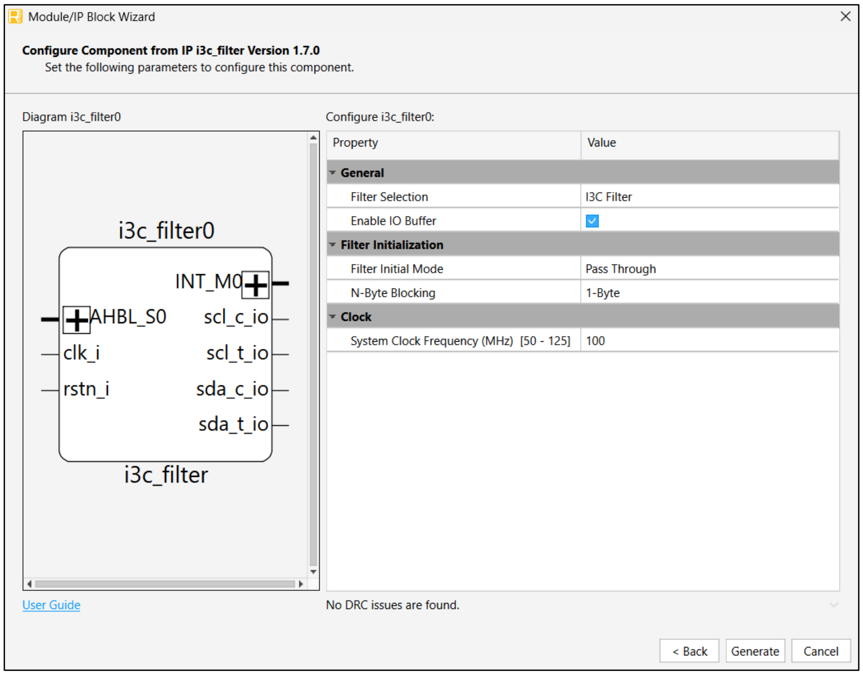
Component name: i3c_filter0

Create in: C:/RadiantProjects/i3c_filter

This will automatically create a folder for i3c_filter0 inside the C:/RadiantProjects/i3c_filter

Figure 7.1. Module/IP Block Wizard

3. In the next **Module/IP Block Wizard** window, customize the selected I3C Filter IP using drop-down lists and check boxes. Figure 7.2 shows an example configuration of the I3C Filter IP. For details on the configuration options, refer to the IP Parameter Description section.



Module/IP Block Wizard

Configure Component from IP i3c_filter Version 1.7.0
Set the following parameters to configure this component.

Diagram i3c_filter0

Configure i3c_filter0:

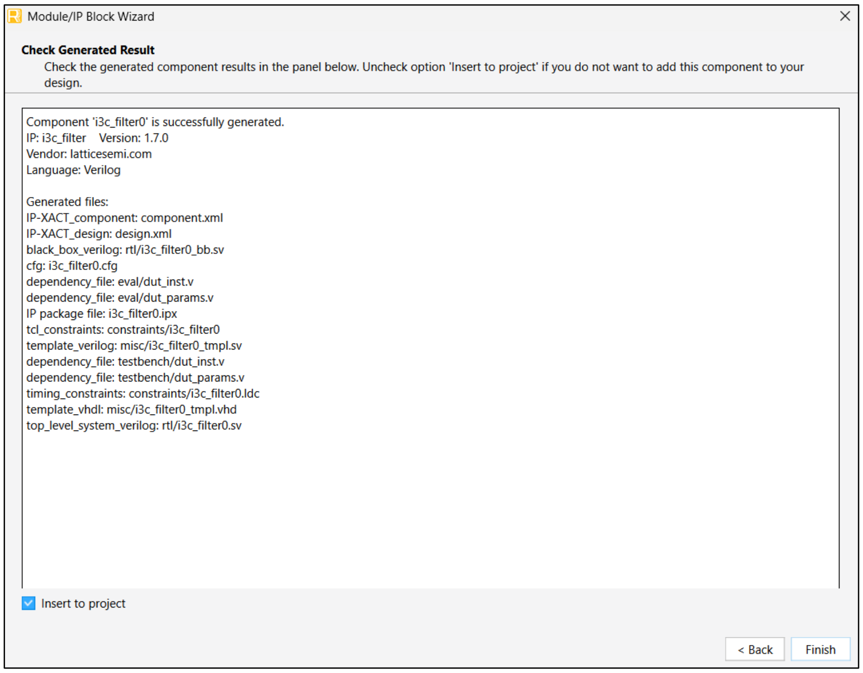
Property	Value
General	
Filter Selection	I3C Filter
Enable IO Buffer	☑
Filter Initialization	
Filter Initial Mode	Pass Through
N-Byte Blocking	1-Byte
Clock	
System Clock Frequency (MHz) [50 - 125]	100

User Guide

No DRC issues are found.

Figure 7.2. IP Configuration

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



Module/IP Block Wizard

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 'i3c_filter0' is successfully generated.
IP: i3c_filter    Version: 1.7.0
Vendor: latticesemi.com
Language: Verilog

Generated files:
IP-XACT_component: component.xml
IP-XACT_design: design.xml
black_box_verilog: rtl/i3c_filter0_bb.sv
cfg: i3c_filter0.cfg
dependency_file: eval/dut_inst.v
dependency_file: eval/dut_params.v
IP package file: i3c_filter0.ipx
tcl_constraints: constraints/i3c_filter0
template_verilog: misc/i3c_filter0_tmpl.sv
dependency_file: testbench/dut_inst.v
dependency_file: testbench/dut_params.v
timing_constraints: constraints/i3c_filter0.ldc
template_vhdl: misc/i3c_filter0_tmpl.vhd
top_level_system_verilog: rtl/i3c_filter0.sv
```

☑ Insert to project

Figure 7.3. Check Generated Result

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

7.1.1. Generated Files and File Structure

The generated I3C Filter module package includes the closed-box (<Component name>_bb.v) and instance templates (<Component name>_tmpl.v/vhd) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file (<Component name>.v) that can be used as an instantiation template for the module is also provided. You may also use this top-level reference as the starting template for the top-level for their complete design. The generated files are listed in Table 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 module.
rtl/<Component name>_bb.v	This file provides the synthesis closed-box.
misc/<Component name>_tmpl.v misc /<Component name>_tmpl.vhd	These files provide instance templates for the module.
eval/constraint.pdc	This file provides information on how to constrain this IP in your design. Refer to the Timing and Physical Constraints section on how to use this file.

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.

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

Refer to the relevant sections in the Lattice Radiant Software User Guide for more information on how to create or edit constraints and how to use the Device Constraint Editor.

7.3. Timing and Physical Constraints

A sample constraint file is included in the output files after generating and instantiating the IP. Add the content of the following IP constraint file to your design constraints:

<IP_Instance_Path>/<IP_Instance_Name>/eval/constraint.pdc.

This 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, copy the content of *constraint.pdc* to the top-level design constraint for post-synthesis.

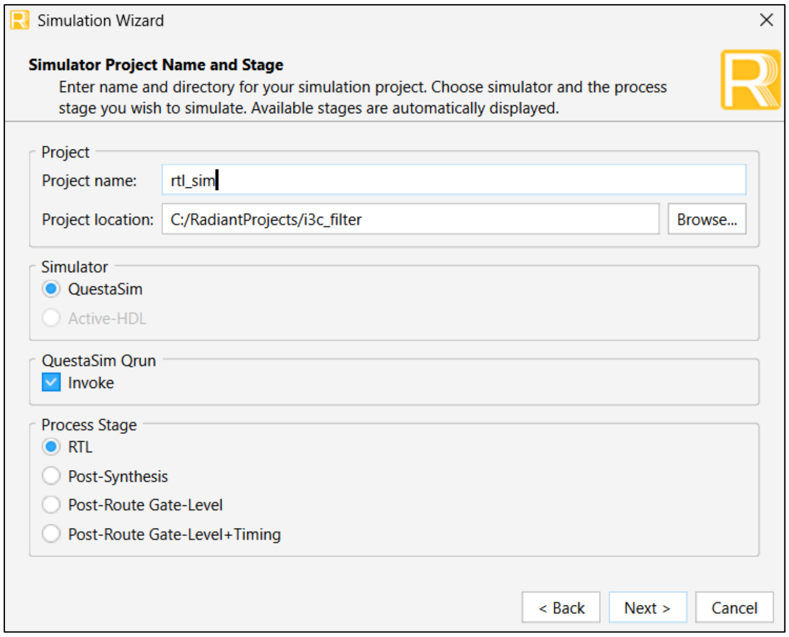
Refer to Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059) for details on how to constrain your design.

7.4. Running Functional Simulation

You can run functional simulation after the IP is generated.

To run functional simulation:

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



Simulation Wizard

Simulator Project Name and Stage

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

Project

Project name: rtl_sim

Project location: C:/RadiantProjects/i3c_filter

Simulator

● QuestaSim

○ Active-HDL

QuestaSim Qrun

☑ Invoke

Process Stage

● RTL

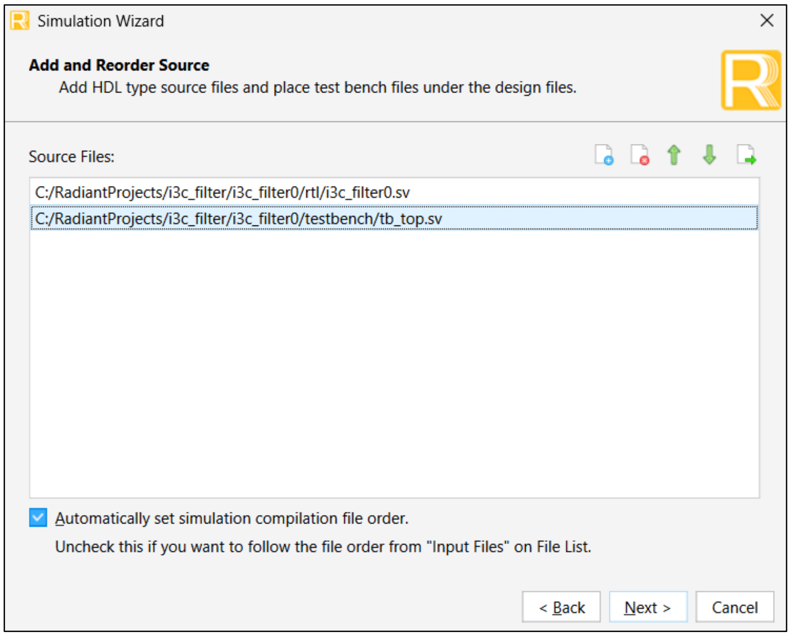
○ Post-Synthesis

○ Post-Route Gate-Level

○ Post-Route Gate-Level+Timing

Figure 7.4. Simulation Wizard

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



Simulation Wizard

Add and Reorder Source

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

Source Files:

C:/RadiantProjects/i3c_filter/i3c_filter0/rtl/i3c_filter0.sv

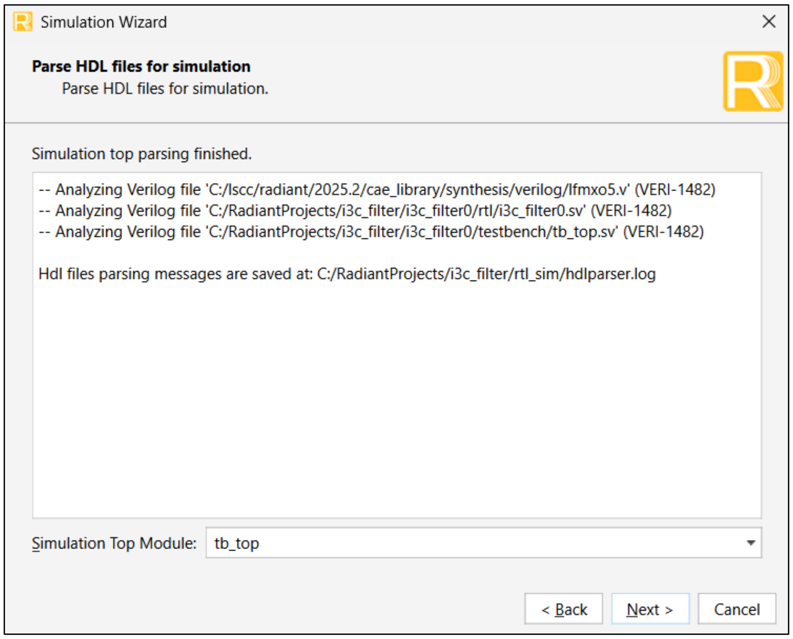
C:/RadiantProjects/i3c_filter/i3c_filter0/testbench/tb_top.sv

☑ Automatically set simulation compilation file order.

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

Figure 7.5. Adding and Reordering Source

3. Click **Next**. Select **tb_top** as **Simulation Top Module** in the **Parse HDL files for simulation** window as shown in Figure 7.6.



Parse HDL files for simulation
Parse HDL files for simulation.

Simulation top parsing finished.

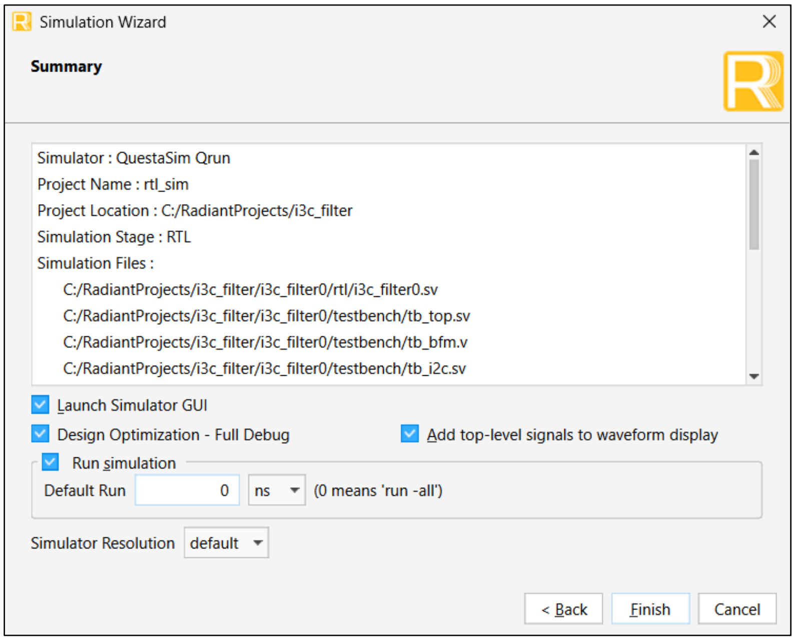
```
-- Analyzing Verilog file 'C:/lscc/radiant/2025.2/cae_library/synthesis/verilog/lfmxo5.v' (VERI-1482)
-- Analyzing Verilog file 'C:/RadiantProjects/i3c_filter/i3c_filter0/rtl/i3c_filter0.sv' (VERI-1482)
-- Analyzing Verilog file 'C:/RadiantProjects/i3c_filter/i3c_filter0/testbench/tb_top.sv' (VERI-1482)

Hdl files parsing messages are saved at: C:/RadiantProjects/i3c_filter/rtl_sim/hdlparser.log
```

Simulation Top Module: tb_top

Figure 7.6. Selecting Simulation Top Module

4. Click **Next**. The **Summary** window is shown.



Summary

Simulator : QuestaSim Qrun
Project Name : rtl_sim
Project Location : C:/RadiantProjects/i3c_filter
Simulation Stage : RTL
Simulation Files :
  C:/RadiantProjects/i3c_filter/i3c_filter0/rtl/i3c_filter0.sv
  C:/RadiantProjects/i3c_filter/i3c_filter0/testbench/tb_top.sv
  C:/RadiantProjects/i3c_filter/i3c_filter0/testbench/tb_bfm.v
  C:/RadiantProjects/i3c_filter/i3c_filter0/testbench/tb_i2c.sv

☑ Launch Simulator GUI
☑ Design Optimization - Full Debug ☑ Add top-level signals to waveform display
☑ Run simulation
Default Run 0 ns (0 means 'run -all')

Simulator Resolution default

Figure 7.7. Simulation Wizard Summary

5. Click **Finish** to run the simulation. See the Simulation Results section for example results.

7.4.1. Simulation Results

Figure 7.8 shows the top signals of I3C Filter IP in pass through operation. In pass through operation, SDA and SCL signals are passed through from Controller to Target. *sda_c_io* and *scl_c_io* are equivalent to *sda_t_io* and *scl_t_io* respectively.

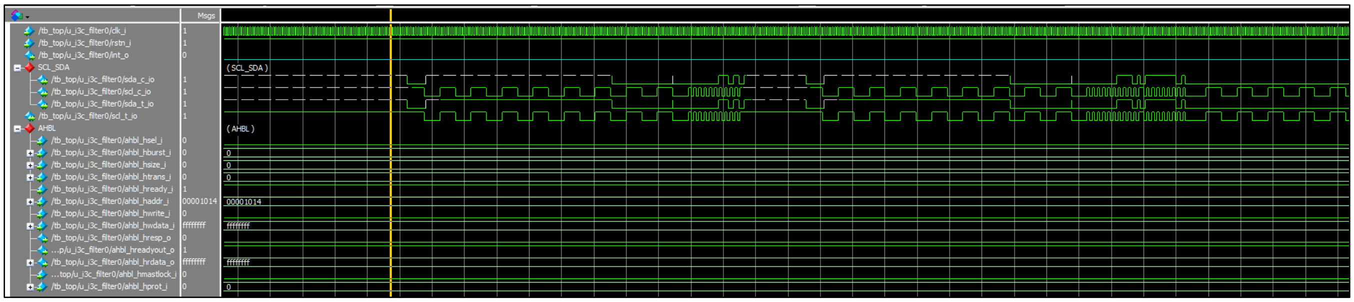

Msgs
/tb_top/u_i3c_filter0/clk_i 1
/tb_top/u_i3c_filter0/rstn_i 1
/tb_top/u_i3c_filter0/int_o 0
SCL_SDA
/tb_top/u_i3c_filter0/sda_c_io 1
/tb_top/u_i3c_filter0/scl_c_io 1
/tb_top/u_i3c_filter0/sda_t_io 1
/tb_top/u_i3c_filter0/scl_t_io 1
AHBL
/tb_top/u_i3c_filter0/ahbl_hsel_i 0
/tb_top/u_i3c_filter0/ahbl_hburst_i 0
/tb_top/u_i3c_filter0/ahbl_hsize_i 0
/tb_top/u_i3c_filter0/ahbl_htrans_i 0
/tb_top/u_i3c_filter0/ahbl_hready_i 1
/tb_top/u_i3c_filter0/ahbl_haddr_i 00001014
/tb_top/u_i3c_filter0/ahbl_hwrite_i 0
/tb_top/u_i3c_filter0/ahbl_hwdata_i ffffffff
/tb_top/u_i3c_filter0/ahbl_hresp_o 0
...p/u_i3c_filter0/ahbl_hreadyout_o 1
/tb_top/u_i3c_filter0/ahbl_hrdata_o ffffffff
...top/u_i3c_filter0/ahbl_hmastlock_i 0
/tb_top/u_i3c_filter0/ahbl_hprot_i 0
(SCL_SDA)
(AHBL)
0
0
0
00001014
ffffffff
ffffffff
0

Figure 7.8. Simulation Waveform for Pass Through Mode

Figure 7.9 shows the waveform when I3C filters blocks I3C command. In the waveform, I3C write to Address 7'h2D with command 8'h39. SDA and SCL from Controller are equivalent to SDA and SCL to Target from START – Broadcast Address – Repeated Start – Target Address/W – ACK – Command. After the Command, I3C Filter generated a STOP condition at the SDA and SCL to Target. The int_o signal is also asserted to indicate that a blocked command event occurred.

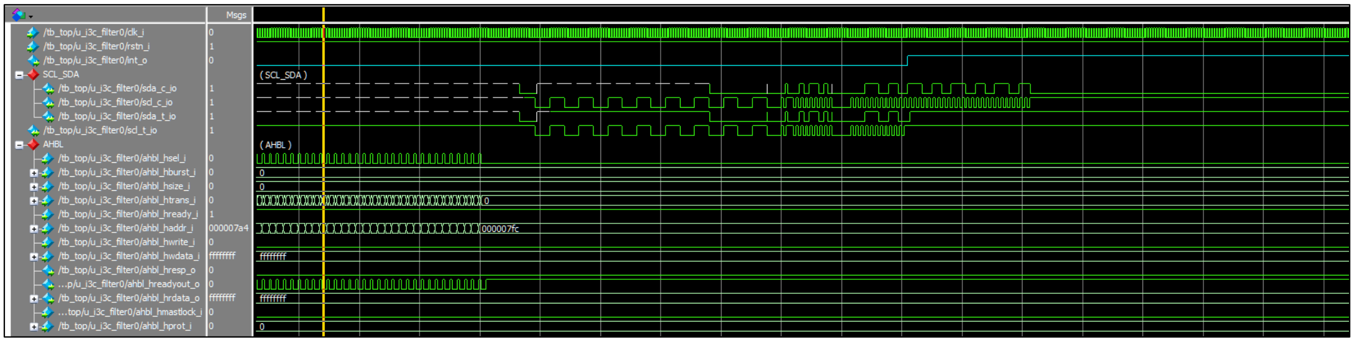

Msgs
/tb_top/u_i3c_filter0/clk_i 0
/tb_top/u_i3c_filter0/rstn_i 1
/tb_top/u_i3c_filter0/int_o 0
SCL_SDA
/tb_top/u_i3c_filter0/sda_c_io 1
/tb_top/u_i3c_filter0/scl_c_io 1
/tb_top/u_i3c_filter0/sda_t_io 1
/tb_top/u_i3c_filter0/scl_t_io 1
AHBL
/tb_top/u_i3c_filter0/ahbl_hsel_i 0
/tb_top/u_i3c_filter0/ahbl_hburst_i 0
/tb_top/u_i3c_filter0/ahbl_hsize_i 0
/tb_top/u_i3c_filter0/ahbl_htrans_i 0
/tb_top/u_i3c_filter0/ahbl_hready_i 1
/tb_top/u_i3c_filter0/ahbl_haddr_i 000007a4
/tb_top/u_i3c_filter0/ahbl_hwrite_i 0
/tb_top/u_i3c_filter0/ahbl_hwdata_i ffffffff
/tb_top/u_i3c_filter0/ahbl_hresp_o 0
...p/u_i3c_filter0/ahbl_hreadyout_o 0
/tb_top/u_i3c_filter0/ahbl_hrdata_o ffffffff
...top/u_i3c_filter0/ahbl_hmastlock_i 0
/tb_top/u_i3c_filter0/ahbl_hprot_i 0
(SCL_SDA)
(AHBL)
0
0
0
000007fc
ffffffff
ffffffff
0

Figure 7.9. Simulation Waveform for Blocking Mode

Appendix A. Resource Utilization

Table A.1 shows the resource utilization of the I3C Filter IP Core on the LFMXO5-25-7BBG256C device using Synplify Pro of the Lattice Radiant Software 2024.1.

Table A.1. LFMXO5-25-7BBG256C Device Resource Utilization

IP Configuration	Clock Frequency (MHz)[1]	LUTs[2]	Registers	EBR
Filter Selection = I3C, Enable IO Buffer = True	141.70	569/23040	684/23637	2/80
Filter Selection = I2C, Enable IO Buffer = True	148.28	396/23040	277/23637	2/80

Notes:

1. Fmax is generated when the FPGA design contains only the I3C Filter module and the target frequency is 100 MHz. These values may be reduced when user logic is added to the FPGA design.
2. The distributed RAM utilization is accounted for in the total LUT4s utilization. The actual LUT4 utilization is distributed among logic, distributed RAM, and ripple logic.

References

- MachXO5-NX web page
- I3C Filter 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
- MachXO5-NX Development Board web page
- MIPI I3C Specification web page
- I2C Bus Specification and User Manual
- AMBA 3 AHB-Lite Protocol Specification
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- I3C Controller IP User Guide (FPGA-IPUG-02228)
- I3C Target IP User Guide (FPGA-IPUG-02227)
- I3C Filter IP Release Notes (FPGA-RN-02004)
- I3C Filter Driver API Reference (FPGA-TN-02390)
- 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 1.4, IP v1.7.0, December 2025

Section	Change Summary
All	• Updated the IP version on the cover page. • Added a note on the IP version in the *Quick Facts* and *Revision History* sections. • Made editorial fixes.
Introduction	• Updated *Lattice Implementation* in Table 1.1. Summary of the I3C Filter IP. • Added a note on system clock frequency to Table 1.2. I3C Filter IP Support Readiness. • In the Features section: • Added bidirectional filtering. • Updated the descriptions for read and write access. • Updated the Licensing and Ordering Information section and removed the *Ordering Part Number* section. • Added the Attribute Names section.
Functional Description	• Updated Figure 2.1. I3C Filter Core Block Diagram. • Added control and status registers, and memory blocks to the IP Architecture Overview section. • Updated the Reset and User Interfaces sections. • Added the I2C Filter Mode section. • Added */I2C* to step 4 in the Programming Flow section.
IP Parameter Description	• In Table 3.1. General Attributes: • Updated the selectable values for *Filter Selection*. • Added the *Enable Bidirectional Filtering* attribute. • Removed the *Enable Debug Ports* attribute. • Added the *Enable Bidirectional Filtering* attribute to Table 3.2. IP Parameter Settings for Example Use Cases.
Signal Description	In Table 4.1. Signal Description: • Updated the descriptions for *scl_speed_i*, *ahbl_hburst_i*, and *ahbl_hsize_i*. • Removed all debug signals.
Register Description	• Updated the introductory paragraph in this section. • Added table captions to all tables. • Added the following tables: • Table 5.1. Register Access Types • Table 5.12. Bus Status • Updated the following tables: • Table 5.2. I3C Filter IP Register Summary • Table 5.6. Filter Configuration • Renamed the *Default* column to *Reset* column in the following tables: • Table 5.3. Target Device Allow List Assignment • Table 5.4. 1-Byte Filtering Allow List 0 (updated the parameter setting) • Table 5.5. 2-Byte Filtering Allow List 0 (updated the parameter setting) • Table 5.7. Target Device Support • Table 5.8. Interrupt Enable • Table 5.9. Interrupt Status • Table 5.10. Interrupt Set • Table 5.11. Blocked Command Information
Example Design	• Added *Enable Bidirectional Filtering* to Table 6.1. I3C Filter IP Configuration Supported by the Example Design. • Updated the Hardware Testing Setup section.

Section	Change Summary
Designing with the IP	- Added a note on screenshots in this section. - Updated Figure 7.1. Module/IP Block Wizard – Figure 7.7. Simulation Wizard Summary.
References	Added the *Lattice Solutions Reference Designs* web page and removed the *Lattice Radiant Software 2023.2 User Guide*.

Revision 1.3, IP v1.6.0, July 2025

Section	Change Summary
All	Updated the IP version information on the cover page.
Introduction	- In Table 1.1. Summary of the I3C Filter IP: - Updated *Supported FPGA Family* to *Supported Devices*. - Removed *Targeted Devices*. - Updated *Lattice Implementation*. - In Table 1.3. Ordering Part Number: - Updated *Single Machine Annual* to *Single Seat Annual*. - Updated *Multi-Site Perpetual* to *Single Seat Perpetual*.

Revision 1.2, IP v1.5.0, February 2025

Section	Change Summary
All	Updated the IP version information on the cover page.
Introduction	Updated the IP version information in Table 1.1. Summary of the I3C Filter IP.
References	Added *I3C Filter Driver API Reference (FPGA-TN-02390)*.

Revision 1.1, IP v1.4.0, December 2024

Section	Change Summary
All	Added the IP version information on the cover page.
Abbreviations in This Document	Added *In-Band Interrupt* (*IBI*).
Introduction	- Updated Table 1.1. Summary of the I3C Filter IP: - Removed *IP Version*. - Added *IP Changes* and *Resources*. - Updated *Targeted Devices* and *Lattice Implementation*. - Replaced the *IP Validation Summary* section with the IP Support Summary section. - Added the Hardware Support section.
Functional Description	Updated Figure 2.1. I3C Filter Core Block Diagram.
IP Parameter Description	Added the *Enable Debug Ports* attribute to Table 3.1. General Attributes.
Signal Description	Added *Debug Signals* to Table 4.1. Signal Description.
Example Design	Added an introductory paragraph to list the evaluation boards used for the example design.
Designing with the IP	Updated Figure 7.1. Module/IP Block Wizard, Figure 7.2. IP Configuration, and Figure 7.3. Check Generated Result.
Resource Utilization	Made editorial fixes.
References	Added the *MachXO5-NX Development Board* web page.

Revision 1.0, June 2024

Section	Change Summary
All	Initial release.