Lattice Sentry I2C Filter IP

IP Version: v1.4.0

User Guide

FPGA-IPUG-02166-1.4

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

Contents 3
Acronyms in This Document 5
1. Introduction 6
1.1. Features 6
1.2. Conventions 6
1.3. Licensing Information 6
2. Functional Description 7
2.1. Overview 7
2.2. Read Transaction 7
2.3. Non-blocked Write Transaction 8
2.4. Blocked Write Transaction 8
2.5. Modules Description 9
2.5.1. Packaged SMBus IP Block Diagrams 9
2.5.2. Signals Description 10
2.5.3. Attributes 11
2.5.4. Register Description 12
3. Program Flow 16
3.1. Initialization 16
3.2. Interrupt Mode 16
3.3. Polling Mode 16
3.4. C Code API 16
4. I2C Filter IP Generation 17
5. Applicable Devices 19
References 20
Technical Support 21
Revision History 22

Figures

Figure 2.1. I2C Filter Topology 7
Figure 2.2. I2C Filter Read Transaction 7
Figure 2.3. I2C Filter Non-blocked Write Transaction 8
Figure 2.4. I2C Filter Blocked Write Transaction 8
Figure 2.5. Packaged I2C Filter Block Diagram 9
Figure 2.6. Functional Block Diagram 10
Figure 3.1. SMBus IP Program Flow 16
Figure 4.1. Entering Component Name 17
Figure 4.2. Configuring Parameters 17
Figure 4.3. Verifying Results 18
Figure 4.4. Specifying Instance Name 18
Figure 4.5. Generated Instance 18

Tables

Table 2.1. Interface Signal Description 10
Table 2.2. Attributes Description 11
Table 2.3. Register Address Map 12

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AHB-Lite	Advanced High-performance Bus – Lite
I2C	Inter-Integrated Circuit
IP	Intellectual Property
SMBus	System management bus
FPGA	Field Programmable Gate Array
RoT	Root of Trust
CPU	Central Processing Unit

1. Introduction

SMBus Relay with filter (named I2C filter in this document) is designed to function as an invisible relay from the point of view of both Controller and Target devices on the bus. It is meant to be directly attached to the Controller port and protect all Target devices against malicious traffic generated from the Controller port based on a allow list of allowable commands set by the host (such as CPU, FPGA RoT design, and others). The filter IP is the subset of the SMBus protocol. SMBSUS# and SMBALERT# are not supported.

The design is implemented in Verilog HDL. It can be configured and generated using the Lattice Propel™ Builder software. It can be implemented using the Lattice Radiant™ software and the Lattice Diamond® Place and Route tool. The module registers are accessed by the host (such as CPU, FPGA RoT design, and others) using a 32-bit Advanced High-performance Bus - Lite (AHB-Lite) interface.

1.1. Features

The I2C filter soft IP has the following features:

- Provides four interfaces, namely, AHB Lite, SMBus controller, SMBus target and Interrupt.
- Connected between a single controller (Primary) and multiple target (Secondary) devices.
- Protects the secondary devices from malicious traffic generated from the controller.
- Does not violate SMBus protocol and is transparent between the Primary and Secondary devices.
- Allows all the Read access.
- Verifies all the write access against a allow list of allowable opcodes (SMBus command) set inside the memory.
- The opcodes in the memory can be initialized and/or 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 secondary devices.
- Supports clock stretching from both primary and secondary devices.
- Supports glitch filter from both primary and secondary devices.
- Each bit in the memory represents one opcode (SMBus command) equivalent to the address. Write is allowed if the bit value is *1* and not allowed if the bit value is *0*. 256 bits required to be supported for each Target device.
- Supports up to 128 target-devices on the bus (7-bit addressing supported only).
- When a block event occurs, the write transaction is halted, an interrupt is sent to the host, and the status register is updated with the blocked command for the corresponding target address.

1.2. Conventions

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

1.3. Licensing Information

The Lattice Sentry I2C Filter IP is provided at no additional cost with the Lattice Radiant and Lattice Propel software.

2. Functional Description

2.1. Overview

The Relay logic is required to pass the communication between Controller and Target devices that does not violate SMBus protocol and is transparent from both Controller and Target devices attached on the bus. This is achieved through a mix of oversampling the bus as well as comprehension of direction changes during transmitting of SMBus frames. The Filter logic allows all read accesses. Any write access is required to be verified against a allow list of allowable opcodes. Based on SMBus protocol, the first byte after start bit is the desired 7-bit target address along with the operation bit : 0 = write, 1 = read. The second byte is the SMBus command. The following bytes are data. The write access is verified by checking the command byte according to the target's allow list.

As shown in Figure 2.1, the I2C filter uses one AHB-Lite interface for filter allow list configuration and register access. The SMBus controller port is used to connect by SMBus controller while SMBus target port is used to connect by SMBus target. There is only one SMBus controller and up to 128 targets on one bus.

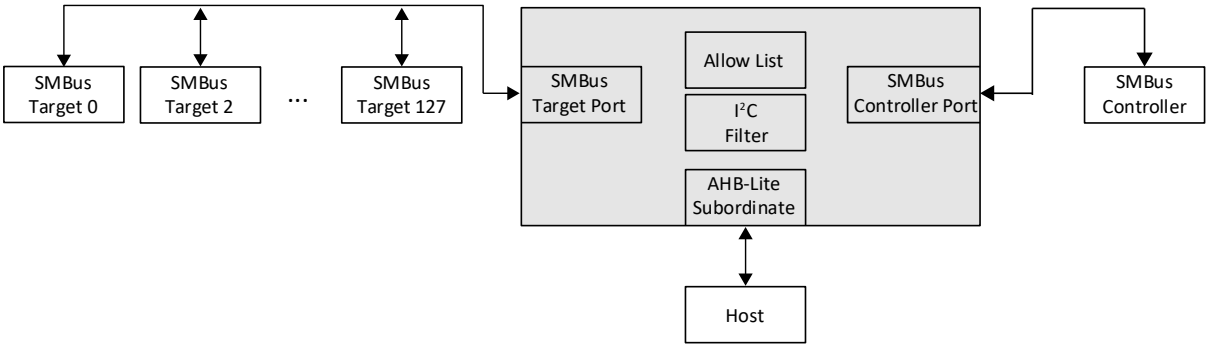

Figure 2.1. I2C Filter Topology

2.2. Read Transaction

Figure 2.2 shows an example of a read transaction relayed through the component. The skew visible between Controller (SMBus controller device) --> filter (I2C filter controller port) and filter (I2C filter target port) --> Target (SMBus target device) lines is the result of passing through the Relay and is within margins not violating SMBus signaling. Read transaction is always allowed by the I2C filter, ignoring the allow list.

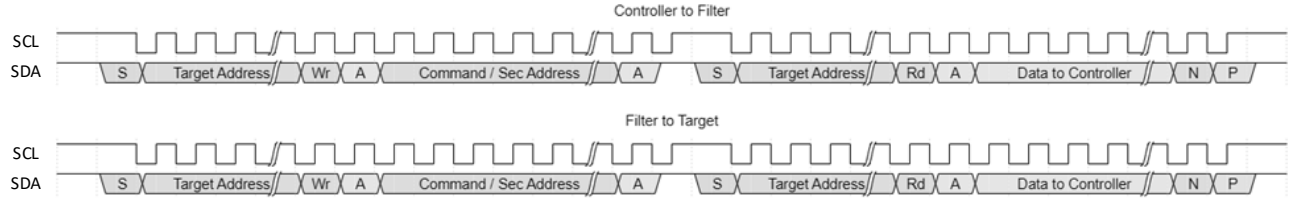

Figure 2.2. I2C Filter Read Transaction

2.3. Non-blocked Write Transaction

Figure 2.3 shows an example of a valid write transaction relayed through the component. The write transaction is not blocked because the command transferred is in the allow list.

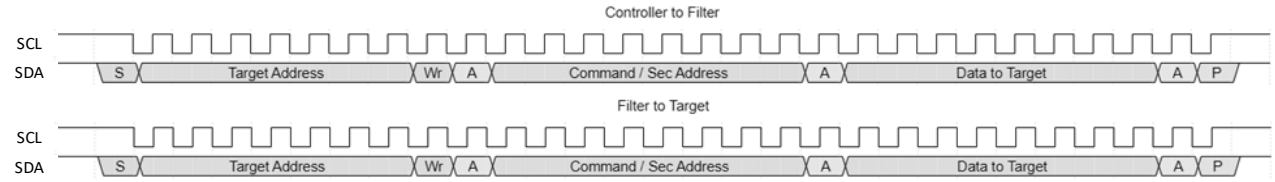

Figure 2.3. I2C Filter Non-blocked Write Transaction

2.4. Blocked Write Transaction

Figure 2.4 below shows an example of invalid write transaction filtered out through the component, in this case the I2C filter asserts SMBus Stop condition (P) leading to an abort on the Target device, while also responding with a NACK to the Controller device signaling an error in the write transaction. Stop Target device occurs at command phase while NACK to Controller device occurs at first byte data phase. The write transaction is blocked due to the command transferred is not in the allow list.

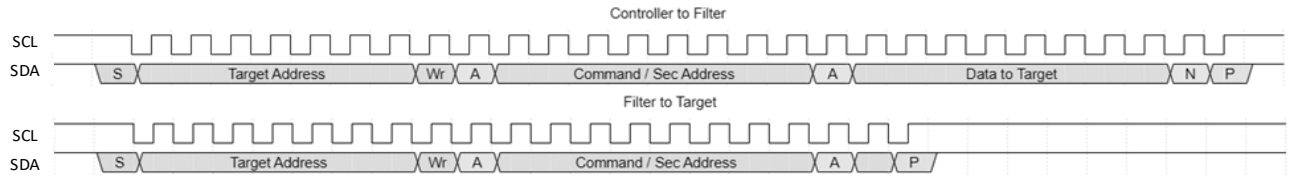

Figure 2.4. I2C Filter Blocked Write Transaction

2.5. Modules Description

2.5.1. Packaged SMBus IP Block Diagrams

When used in the Lattice Propel Builder software, I2C filter is packaged to be compliant with the IP-XACT IEEE_1685 standard. Figure 2.5 shows the packaged IP block diagram. There are four attributes:

- System Clock Frequency (MHz) — the frequency of clk_i (25 MHz to 125 MHz).
- Enable SCL stretch — enables the SMBus SCL stretch function
- Enable Glitch Filter — enables the SMBus SCL/SDA glitch filter function. Glitch filter eliminates the effect of noise glitches. The feature is implemented as a soft logic. The length of the glitches filtered is based on the System Clock Frequency. The effective glitch filter length is calculated as follows:

$$\text{System Clock Period (ns)} = \frac{1000}{\text{System Clock Frequency (MHz)}}$$

$$\text{Effective Glitch Filter (ns)} = \left(\left\lfloor \frac{50}{\text{Sytem Clock Period (ns)}} \right\rfloor + 1\right) \times \text{System Clock Period (ns)}$$

- Remove Tri-State Buffer – removes tri-state buffer and exposes input, output, and output enable ports.

The functional block diagram of the IP core is shown in Figure 2.6. The registers and allow list of the IP core are accessed by the host (such as CPU, FPGA RoT design, and others) through an AHB-lite interface. The allow list is implemented by the EBR block. The *irq* signal is the interrupt signal to host.

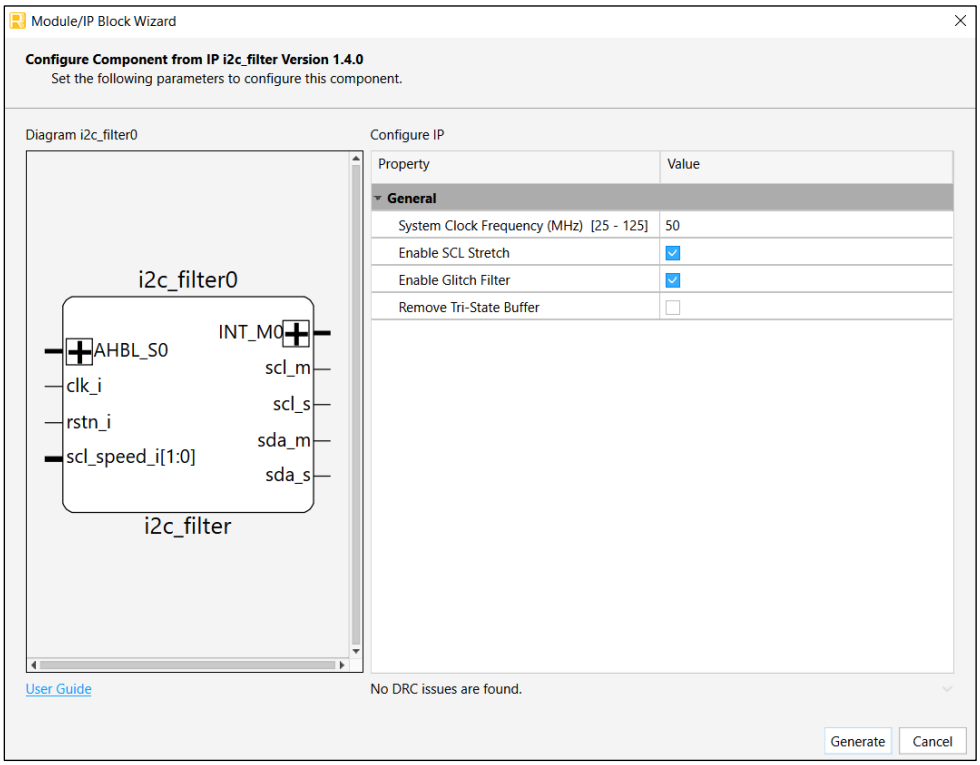

Figure 2.5. Packaged I2C Filter Block Diagram

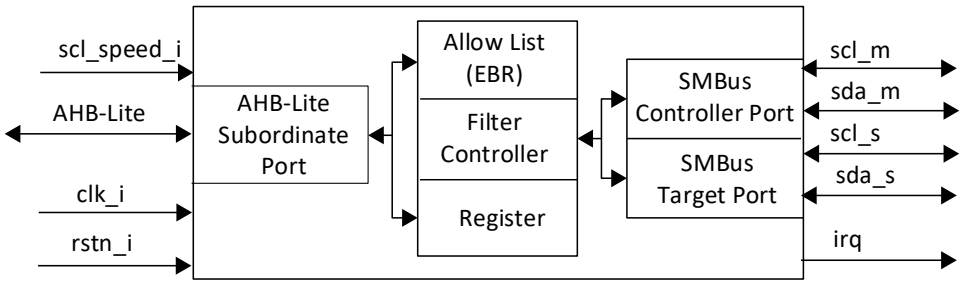

Figure 2.6. Functional Block Diagram

2.5.2. Signals Description

Table 2.1. Interface Signal Description

Signal Name	Width	Direction	Description
Clock and Reset			
clk_i	1	Input	System clock
rstn_i	1	Input	System reset. The reset assertion can be asynchronous but reset negation should be synchronous. This signal is active low, when asserted, output ports and registers are forced to their reset values.
SCL Clock Speed			
scl_speed_i	2	Input	Indicates the SCL clock speed to be sent to the target devices. 2'b00: Reserved 2'b01: 100 KHz 2'b10: 400 KHz 2'b11: 1000 KHz
AHB-lite Bus			
ahbl_hsel_slv_i	1	Input	AHBL Select signal Indicates that the target device is selected and a data transfer is required.
ahbl_haddr_slv_i	32	Input	The system address bus.
ahbl_hburst_slv_i	3	Input	3'b000: SINGLE Single burst 3'b001: INCR Incrementing burst of undefined length (NOT supported) 3'b010: WRAP4 4-bit wrapping burst 3'b011: INCR4 4-bit incrementing burst 4'b100: WRAP8 8-bit wrapping burst 3'b101: INCR8 8-bit incrementing burst 8'b110: WRAP16 16-bit wrapping burst 3'b111: INCR16 16-bit incrementing burst
ahbl_hprot_slv_i	4	Input	ahbl_hprot_slv_i [0] :1'b0 - opcode fetch; 1'b1 - data access ahbl_hprot_slv_i [1]: 1'b0 - user access; 1'b1 - privileged access ahbl_hprot_slv_i [2]: 1'b0 - non-bufferable, 1'b1 - bufferable ahbl_hprot_slv_i [3]: 1'b0 - non-cacheable; 1'b1 - cacheable
ahbl_hsize_slv_i	3	Input	3'b000: 1 byte 3'b001: 2 bytes 3'b010: 4 bytes
ahbl_htrans_slv_i	2	Input	Indicates the transfer type of the current transfer. This can be: 2'b00: IDLE 2'b01: BUSY 2'b10: NONSEQUENTIAL 2'b11: SEQUENTIAL
ahbl_hwdata_slv_i	32	Input	The write data bus
ahbl_hwrite_slv_i	1	Input	When HIGH, this signal indicates a write transfer and when LOW a read transfer.

Signal Name	Width	Direction	Description
ahbl_hready_slv_i	1	Input	This signal should come from AHBL Interconnect. When set to 1, this indicates the previous transfer is complete.
ahbl_hrdata_slv_o	32	Output	The read data bus
ahbl_hreadyout_slv_o	1	Output	When HIGH, this signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
ahbl_hresp_slv_o	1	Output	When LOW, this signal indicates that the transfer status is OKAY. When HIGH, it indicates that the transfer status is ERROR.
Interrupt Signal			
irq	1	Output	Interrupt to host (CPU), reset value is 1'b0.
SMBus Signal (Remove Tri-State Buffer = 0)			
scl_m	1	Input/Output	SMbus clock connect to controller
sda_m	1	Input/Output	SMbus data connect to controller
scl_s	1	Input/Output	SMbus clock connect to target
sda_s	1	Input/Output	SMbus data connect to target
SMBus Signal (Remove Tri-State Buffer = 1)			
scl_m_in	1	Input	SMbus clock input from controller
scl_m_outen	1	Output	SMbus clock output enable to controller 0: Buffer should work as output (tri-state should get scl_m_out) 1: Buffer should work as input
scl_m_out	1	Output	SMbus clock output to controller
sda_m_in	1	Input	SMbus data input from controller
sda_m_outen	1	Output	SMbus data output enable to controller 0: Buffer should work as output (tri-state should get sda_m_out) 1: Buffer should work as input
sda_m_out	1	Output	SMbus data output to controller
scl_s_in	1	Input	SMbus clock input from target
scl_s_outen	1	Output	SMbus clock output enable to target 0: Buffer should work as output (tri-state should get scl_s_out) 1: Buffer should work as input
scl_s_out	1	Output	SMbus clock output to target
sda_s_in	1	Input	SMbus data input from target
sda_s_outen	1	Output	SMbus data output enable to target 0: Buffer should work as output (tri-state should get sda_s_out) 1: Buffer should work as input
sda_s_out	1	Output	SMbus data output to target

2.5.3. Attributes

The configurable attributes of the IP Core are shown in Table 2.2. The attributes can be configured through the IP Catalog's Module/IP wizard of the Lattice Propel Builder. Wherever applicable, default values are in bold.

Table 2.2. Attributes Description

Parameter	Value Range	Description
System Clock Frequency	25~125 MHz, **50MHz**	System clock frequency
Enable SCL Stretch	**Check**, Uncheck	Enable SMBus clock stretch. Check: enable; Uncheck: disable
Enable Glitch Filter	**Check**, Uncheck	Enable SMBus glitch filter. Check: enable; Uncheck: disable
Remove Tri-State Buffer	Check, **Uncheck**	Removes tri-state buffer. Check: no tri-state buffers; Uncheck tri-state buffers are instantiated

2.5.4. Register Description

The register address map, shown in Table 2.3, specifies the available IP core registers. The offset of each register increments by four to allow easy interfacing with the processor and system buses. In this case, each register is 32-bit wide.

Table 2.3. Register Address Map

Offset	Register Name	Access	Reset	Description
Allow List (EBR)				
0x00	Allow list number (range 0 ~ 59). Specify allow list number corresponding to target address. Target addresses vary from 7'd0 to 7'd7f for 7 bits address. Allow list numbers may be 0~59 for this design. But the maximum number could extend to 255 in the future. One target only has one allow list number. But one allow list number can be applied to all targets.	WO	32'h0	Target address 7'd3, 7'd2, 7'd1, 7'd0 allow list number. bit31:24 -> address 7'd3 allow list number bit23:16 -> address 7'd2 allow list number bit15:8 -> address 7'd1 allow list number bit7:0 -> address 7'd0 allow list number
0x04				Target address 7'd7, 7'd6, 7'd5, 7'd4 allow list number. bit31:24 -> address 7'd7 allow list number bit23:16 -> address 7'd6 allow list number bit15:8 -> address 7'd5 allow list number bit7:0 -> address 7'd4 allow list number
0x08				Target address 7'd11, 7'd10, 7'd9, 7'd8 allow list number. bit31:24 -> address 7'd11 allow list number bit23:16 -> address 7'd10 allow list number bit15:8 -> address 7'd9 allow list number bit7:0 -> address 7'd8 allow list number
0x0C				Target address 7'd15, 7'd14, 7'd13, 7'd12 allow list number. bit31:24 -> address 7'd15 allow list number bit23:16 -> address 7'd14 allow list number bit15:8 -> address 7'd13 allow list number bit7:0 -> address 7'd12 allow list number
0x10				Target address 7'd19, 7'd18, 7'd17, 7'd16 allow list number. bit31:24 -> address 7'd19 allow list number bit23:16 -> address 7'd18 allow list number bit15:8 -> address 7'd17 allow list number bit7:0 -> address 7'd16 allow list number
0x14				Target address 7'd23, 7'd22, 7'd21, 7'd20 allow list number. bit31:24 -> address 7'd23 allow list number bit23:16 -> address 7'd22 allow list number bit15:8 -> address 7'd21 allow list number bit7:0 -> address 7'd20 allow list number
0x18				Target address 7'd27, 7'd26, 7'd25, 7'd24 allow list number. bit31:24 -> address 7'd27 allow list number bit23:16 -> address 7'd26 allow list number bit15:8 -> address 7'd25 allow list number bit7:0 -> address 7'd24 allow list number
0x1c				Target address 7'd31, 7'd30, 7'd29, 7'd28 allow list number. bit31:24 -> address 7'd31 allow list number bit23:16 -> address 7'd30 allow list number bit15:8 -> address 7'd29 allow list number bit7:0 -> address 7'd28 allow list number

Offset	Register Name	Access	Reset	Description
0x20				Target address 7'd35~7'd32 allow list number. Bitmap refer to above.
0x24				Target address 7'd39~7'd36 allow list number.
0x28				Target address 7'd43~7'd40 allow list number. Bitmap refer to above.
0x2c				Target address 7'd47~7'd44 allow list number. Bitmap refer to above.
0x30				Target address 7'd51~7'd48 allow list number. Bitmap refer to above.
0x34				Target address 7'd55~7'd52 allow list number. Bitmap refer to above.
0x38				Target address 7'd59~7'd56 allow list number. Bitmap refer to above.
0x3c				Target address 7'd63~7'd60 allow list number. Bitmap refer to above.
0x40				Target address 7'd67~7'd64 allow list number. Bitmap refer to above.
0x44				Target address 7'd71~7'd68 allow list number. Bitmap refer to above.
0x48				Target address 7'd75~7'd72 allow list number. Bitmap refer to above.
0x4c				Target address 7'd79~7'd76 allow list number. Bitmap refer to above.
0x50				Target address 7'd83~7'd80 allow list number. Bitmap refer to above.
0x54				Target address 7'd87~7'd84 allow list number. Bitmap refer to above.
0x58				Target address 7'd91~7'd88 allow list number. Bitmap refer to above.
0x5c				Target address 7'd95~7'd92 allow list number. Bitmap refer to above.
0x60				Target address 7'd99~7'd96 allow list number. Bitmap refer to above.
0x64				Target address 7'd103~7'd100 allow list number. Bitmap refer to above.
0x68				Target address 7'd107~7'd104 allow list number. Bitmap refer to above.
0x6c				Target address 7'd111~7'd108 allow list number. Bitmap refer to above.
0x70				Target address 7'd115~7'd112 allow list number. Bitmap refer to above.
0x74				Target address 7'd119~7'd116 allow list number. Bitmap refer to above.
0x78				Target address 7'd123~7'd120 allow list number. Bitmap refer to above.
0x7c				Target address 7'd127~7'd124 allow list number. Bitmap refer to above.

<table>
<tr><th>Offset</th><th>Register Name</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>0x80</td><td>allow list number 0</td><td rowspan="48">WO</td><td rowspan="48">256’h0</td><td rowspan="48">There are 256 bits in this range. Every bit represents a command allow list. For example, bit9 is the SMBus command 0x9 allow list. If the bit is “1” command 0x9 is passed in write transaction. If the bit is “0” command 0x9 is blocked in write transaction.

<table><tr><th>Offset</th><th>Byte3</th><th>Byte2</th><th>Byte1</th><th>Byte0</th></tr><tr><td>0x00</td><td>Bit31:24</td><td>Bit23:16</td><td>Bit15:8</td><td>Bit7:0</td></tr><tr><td>0x04</td><td>Bit63:56</td><td>Bit55:48</td><td>Bit47:40</td><td>Bit39:32</td></tr><tr><td>0x08</td><td>Bit95:88</td><td>Bit87:80</td><td>Bit79:72</td><td>Bit71:64</td></tr><tr><td>0x0c</td><td>Bit127:120</td><td>Bit119:112</td><td>Bit111:104</td><td>Bit103:96</td></tr><tr><td>0x10</td><td>Bit159:152</td><td>Bit151:144</td><td>Bit143:136</td><td>Bit135:128</td></tr><tr><td>0x14</td><td>Bit191:184</td><td>Bit183:176</td><td>Bit175:168</td><td>Bit167:160</td></tr><tr><td>0x18</td><td>Bit223:216</td><td>Bit215:208</td><td>Bit207:200</td><td>Bit199:192</td></tr><tr><td>0x1c</td><td>Bit255:248</td><td>Bit247:240</td><td>Bit239:232</td><td>Bit231:224</td></tr></table></td></tr>
<tr><td>0xA0</td><td>allow list number 1</td></tr>
<tr><td>0xC0</td><td>allow list number 2</td></tr>
<tr><td>0xE0</td><td>allow list number 3</td></tr>
<tr><td>0x100</td><td>allow list number 4</td></tr>
<tr><td>0x120</td><td>allow list number 5</td></tr>
<tr><td>0x140</td><td>allow list number 6</td></tr>
<tr><td>0x160</td><td>allow list number 7</td></tr>
<tr><td>0x180</td><td>allow list number 8</td></tr>
<tr><td>0x1A0</td><td>allow list number 9</td></tr>
<tr><td>0x1C0</td><td>allow list number 10</td></tr>
<tr><td>0x1E0</td><td>allow list number 11</td></tr>
<tr><td>0x200</td><td>allow list number 12</td></tr>
<tr><td>0x220</td><td>allow list number 13</td></tr>
<tr><td>0x240</td><td>allow list number 14</td></tr>
<tr><td>0x260</td><td>allow list number 15</td></tr>
<tr><td>0x280</td><td>allow list number 16</td></tr>
<tr><td>0x2A0</td><td>allow list number 17</td></tr>
<tr><td>0x2C0</td><td>allow list number 18</td></tr>
<tr><td>0x2E0</td><td>allow list number 19</td></tr>
<tr><td>0x300</td><td>allow list number 20</td></tr>
<tr><td>0x320</td><td>allow list number 21</td></tr>
<tr><td>0x340</td><td>allow list number 22</td></tr>
<tr><td>0x360</td><td>allow list number 23</td></tr>
<tr><td>0x380</td><td>allow list number 24</td></tr>
<tr><td>0x3A0</td><td>allow list number 25</td></tr>
<tr><td>0x3C0</td><td>allow list number 26</td></tr>
<tr><td>0x3E0</td><td>allow list number 27</td></tr>
<tr><td>0x400</td><td>allow list number 28</td></tr>
<tr><td>0x420</td><td>allow list number 29</td></tr>
<tr><td>0x440</td><td>allow list number 30</td></tr>
<tr><td>0x460</td><td>allow list number 31</td></tr>
<tr><td>0x480</td><td>allow list number 32</td></tr>
<tr><td>0x4A0</td><td>allow list number 33</td></tr>
<tr><td>0x4C0</td><td>allow list number 34</td></tr>
<tr><td>0x4E0</td><td>allow list number 35</td></tr>
<tr><td>0x500</td><td>allow list number 36</td></tr>
<tr><td>0x520</td><td>allow list number 37</td></tr>
<tr><td>0x540</td><td>allow list number 38</td></tr>
<tr><td>0x560</td><td>allow list number 39</td></tr>
<tr><td>0x580</td><td>allow list number 40</td></tr>
<tr><td>0x5A0</td><td>allow list number 41</td></tr>
<tr><td>0x5C0</td><td>allow list number 42</td></tr>
<tr><td>0x5E0</td><td>allow list number 43</td></tr>
<tr><td>0x600</td><td>allow list number 44</td></tr>
<tr><td>0x620</td><td>allow list number 45</td></tr>
<tr><td>0x640</td><td>allow list number 46</td></tr>
<tr><td>0x660</td><td>allow list number 47</td></tr>
</table>

Offset	Register Name	Access	Reset	Description
0x680	allow list number 48			
0x6A0	allow list number 49			
0x6C0	allow list number 50			
0x6E0	allow list number 51			
0x700	allow list number 52			
0x720	allow list number 53			
0x740	allow list number 54			
0x760	allow list number 55			
0x780	allow list number 56			
0x7A0	allow list number 57			
0x7C0	allow list number 58			
0x7E0	allow list number 59			
0x800	Interrupt Enable	R/W	8'bxx0x0000	bit0: enable "target no-acked address" interrupt, '1' enable, '0'disable bit1: enable "target no-acked command" interrupt, '1' enable, '0'disable bit2: enable "target no-acked data" interrupt, '1' enable, '0'disable bit3: enable "controller no-acked data" interrupt, '1' enable, '0'disable bit5: enable "command is blocked" interrupt, '1' enable, '0'disable others: not use, ignore
0x804	Interrupt Status	R/W1C*	8'bxx0x0000	bit0: target no-acked address, '1' active, '0' inactive bit1: target no-acked command, '1' active, '0' inactive bit2: target no-acked data, '1' active, '0' inactive bit3: controller no-acked data, '1' active, '0' inactive bit5: command is blocked, '1' active, '0' inactive others: not use, ignore
0x808	Interrupt Set	WO	8'bxx0x0000	bit0: set "target no-acked address" interrupt bit, set '1' trigger interrupt bit1: set "target no-acked command" interrupt bit, set '1' trigger interrupt bit2: set "target no-acked data" interrupt bit, set '1' trigger interrupt bit3: set "controller no-acked data" interrupt bit, set '1' trigger interrupt bit5: set "command is blocked" interrupt bit, set '1' trigger interrupt others: not use, ignore
0x808	Most Recent Address	RO	32'b0	bit31: '1', Data in [6:0] is valid; '0', Data in [6:0] is invalid bit6:0: If valid, most recent received address Register contains the blocked address after a "command block interrupt" has asserted
0x80C	Most Recent Command	RO	32'b0	bit31: '1', Data [7:0] is valid; '0', Data in [7:0] is invalid bit7:0: If valid, most recent received command Register contains the blocked command after a "command block interrupt" has asserted

***Note:** R/W1C, readable and write '1' clear relevant bit.

3. Program Flow

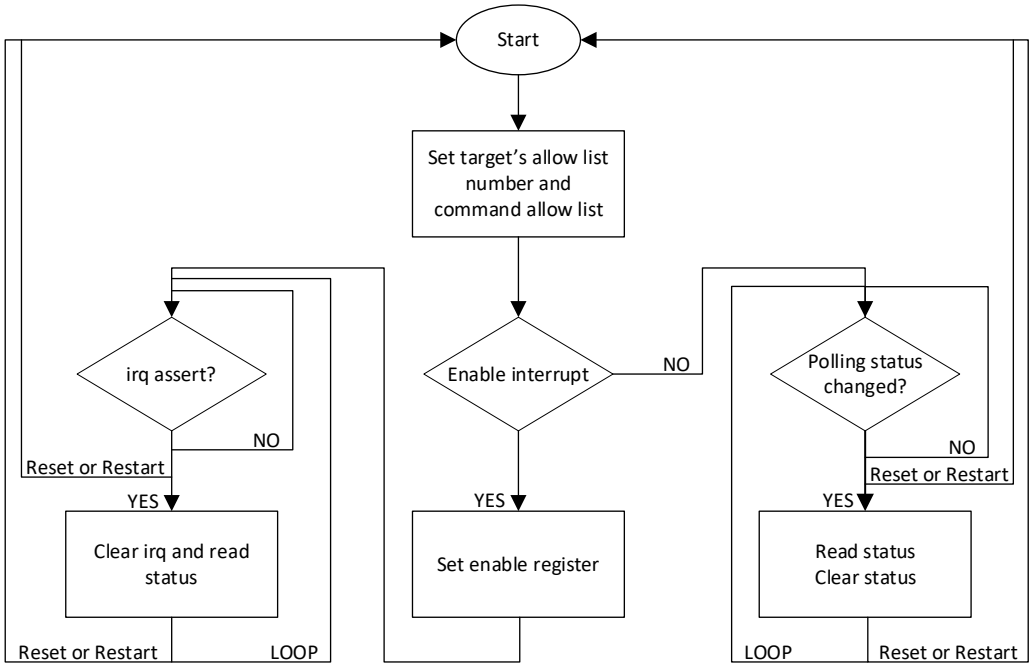

Figure 3.1. SMBus IP Program Flow

3.1. Initialization

When utilizing the IP, the register should be configured first. The targets' allow list number and the allow list contents should be configured through ABH-Lite port. These are two ways of using the IP: Interrupt mode and Polling mode. If using interrupt mode, the interrupt enable register bit should be set accordingly.

3.2. Interrupt Mode

If using interrupt mode, when an interrupt condition is triggered, an interrupt signal (irq) asserts. The host responses to the interrupt, reads the *Interrupt Status* register and writes '1's to clear it. If this is a blocked command event interrupt, the host reads the *Blocked Address* and *Blocked Command* registers to check the blocked address and command. The two registers are valid when bit 31 is set to '1'. Note that new status overrides the current status. So the host needs to handle the interrupt in a timely manner.

3.3. Polling Mode

The SMBus IP can also be used in polling mode. In this mode, after initialization, the host needs to poll the *Interrupt Status* register timely. If a status bit is set, the host also needs to write '1's to clear the relevant bit. Similarly, when a *command is blocked* status is set, the host can read the *Blocked Address* and *Blocked Command* register to check the blocked address and command. The two registers are valid when the bit 31 is set to '1'. Note that the new status overrides the current status. So the host needs to read out the status in a timely manner.

3.4. C Code API

Refer to the IP driver for details.

4. I2C Filter IP Generation

This section provides information on how to generate the I2C Filter IP Core using Propel Builder.

To generate the SMBus IP Core module:

1. In Propel Builder, create a new design. Select the I2C Filter in IP Catalog.
2. Enter the component name as shown in Figure 4.1. Click **Next**.

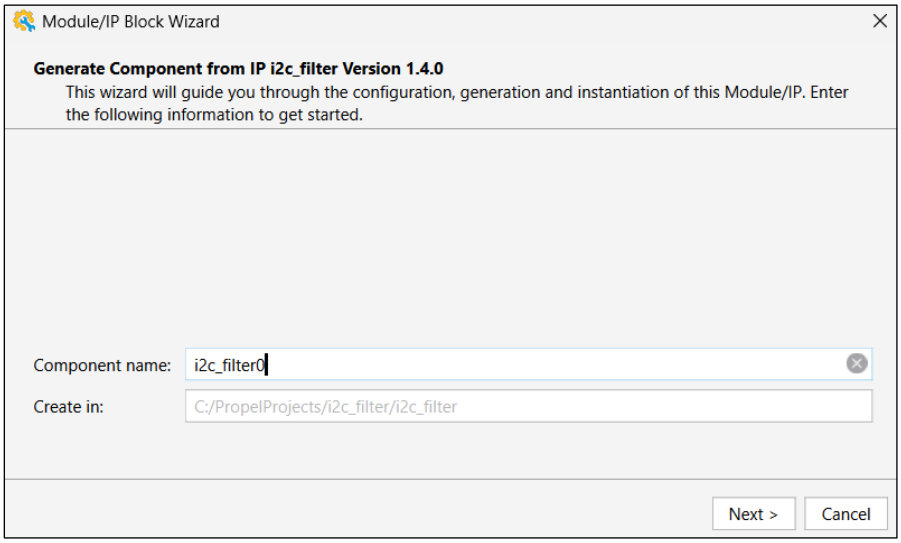

Figure 4.1. Entering Component Name

3. Configure the parameters as shown in Figure 4.2. Click **Generate**.

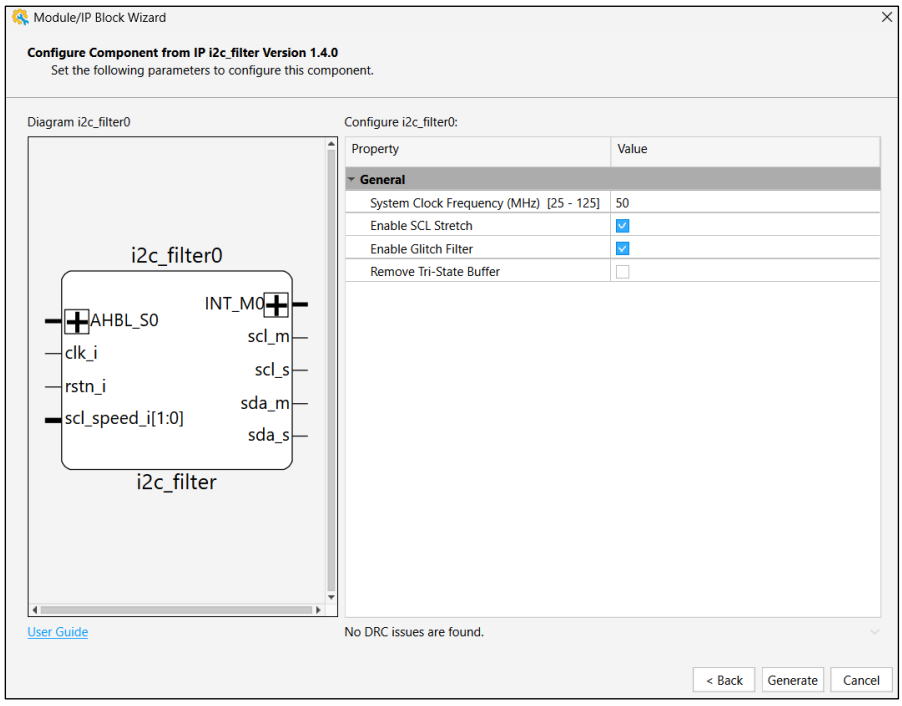

Figure 4.2. Configuring Parameters

4. Verify the information. Click **Finish**.

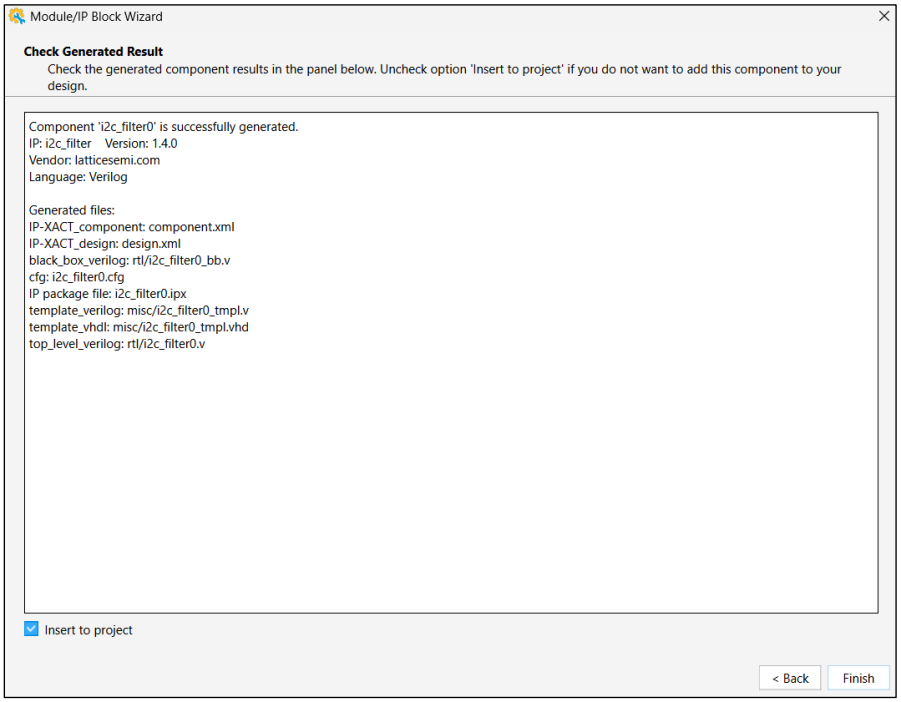

Figure 4.3. Verifying Results

5. Confirm or modify the module instance name. Click **OK**.

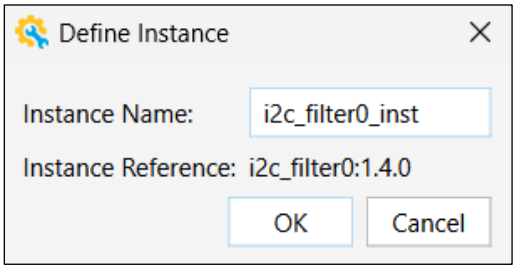

Figure 4.4. Specifying Instance Name

The CPU IP instance is successfully generated as shown in Figure 4.5.

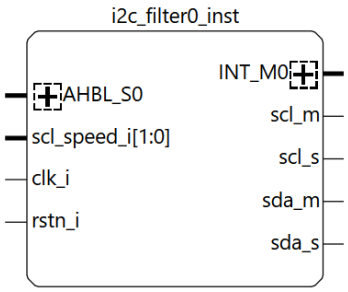

Figure 4.5. Generated Instance

5. Applicable Devices

- MachXO3D™
- Mach™-NX
- MachXO5™-NX
- MachXO4™

References

- Lattice Sentry I2C Filter IP Release Notes (FPGA-RN-02070)
- AMBA 3 AHB-Lite Protocol Specification
- System Management Bus (SMBus) Specification
- Mach-NX web page
- MachXO3D web page
- MachXO4 web page
- MachXO5-NX web page
- Lattice Sentry I2C Filter IP Core web page
- Lattice Radiant Software web page
- Lattice Diamond Software web page
- Lattice Propel Design Environment web page
- Lattice Solutions IP Cores web page
- Lattice Solutions Reference Designs web page
- Lattice Insights web page for Lattice Semiconductor training courses and learning plans

Technical Support

For assistance, submit a technical support case at 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 1.4, IP v1.4.0, December 2025

Section	Change Summary
All	• Added the IP version information on the cover page. • Updated instances of *I²C* to *I2C*. • Added a note on the IP version in the *Revision History* section. • Made editorial fixes.

Functional Description	• Removed the *FPGA Tool* attribute in the Packaged SMBus IP Block Diagrams section.
	• Updated Figure 2.5. Packaged I2C Filter Block Diagram.
	• Added the *Wherever applicable, default values are in bold* sentence to the Attributes section.
	• Updated Table 2.2. Attributes Description:
	  • Bolded the default values for all parameters.
	  • Removed the *FPGA Tool* parameter.

I2C Filter IP Generation	Updated Figure 4.1. Entering Component Name – Figure 4.5. Generated Instance.
Applicable Devices	Added the *MachXO4* device family.
References	Added *Lattice Sentry I2C Filter IP Release Notes (FPGA-RN-02070)* and *MachXO4*, *Lattice Sentry I2C Filter IP Core*, and *Lattice Solutions Reference Designs* web pages.

Revision 1.3, July 2024

Section	Change Summary

All	• Changed document title from *Lattice Sentry I2C Filter IP Core - Lattice Propel Builder* to *Lattice Sentry I²C Filter IP*.
	• Made changes to ensure that the document is consistent with Lattice Semiconductor's inclusive language policy:
	  • Updated *master* to *controller*.
	  • Updated *slave* to *target*.
	  • Updated *whitelist* to *allow list*.

Disclaimers	Updated boilerplate.
Inclusive Language	Added boilerplate.
Introduction	Added the Licensing Information section.

Functional Description	• Updated the following figures:
	  • Figure 2.1. I2C Filter Topology
	  • Figure 2.2. I2C Filter Read Transaction
	  • Figure 2.3. I2C Filter Non-blocked Write Transaction
	  • Figure 2.4. I2C Filter Blocked Write Transaction
	  • Figure 2.6. Functional Block Diagram
	• Updated the descriptions for *Enable Glitch Filter* attribute in the Packaged SMBus IP Block Diagrams section.
	• Replaced *Blocked Address* and *Blocked Command* registers with *Most Recent Address* and *Most Recent Command* registers respectively in Table 2.3. Register Address Map.

Program Flow	Updated Figure 3.1. SMBus IP Program Flow.
I²C Filter IP Generation	Updated all the figures in this section.
Applicable Devices	Added MachXO5™-NX device.
References	Updated this section.

Revision 1.2, April 2023

Section	Change Summary
Functional Description	In section Modules Description: • Removed instances of I2C Clock Frequency. • Updated Figure 2.5 Packaged I2C Filter Block Diagram to remove the I2C Clock Frequency parameter and add a new input signal to the i2c_filter block diagram. • Updated Figure 2.6 Functional block diagram with new input signal scl_speed_i. • Updated Table 2.1. Interface Signal Description to include input signal scl_speed_i. • Updated Table 2.2. Attributes Description to remove the I2C Clock Frequency parameter.
I^2C Filter IP Generation	In section I2C Filter IP Generation: • Updated Figure 4.2. Configuring Parameters to include input signal scl_speed_i and remove the I2C Clock Frequency parameter. • Updated Figure 4.3. Verifying Results. • Updated Figure 4.5. Generated Instance with new input signal scl_speed_i.

Revision 1.1, April 2022

Section	Change Summary
Functional Description	In section Modules Description: • Updated range of the System Clock Frequency • Added option to specify the SCL Clock Frequency • Added option to remove tri-state buffers • Modified table of signals to include changes based on tri-state buffer parameter

Revision 1.0, December 2021

Section	Change Summary
All	Initial release.

www.latticesemi.com