eSPI Target IP

IP Version: v2.2.0

User Guide

FPGA-IPUG-02260-1.1

November 2024

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 acronyms and abbreviations used in this document.

Abbreviation	Definition
AXI	Advanced Extensible Interface
AHB-Lite	Advanced High-Performance Bus – Lite
APB	Advanced Peripheral Bus
CRC	Cyclic Redundancy Check
CS	Chip Select
CSR	Configuration and Status Registers
EBR	Embedded Block RAM
eSPI	Enhanced Serial Peripheral Interface
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
IP	Intellectual Property
LSE	Lattice Synthesis Engine
LUT	Look Up Table
MC	Micro-Controller, RISC-V for Micro-Controller Applications
OOB	Out of Band
PDC	Physical Design Constraint
PLL	Phase-Locked Loop
RISC-V	Reduced Instruction Set Computer-V (Five)
Rx	Receiver
SDK	Software Development Kit
SPI	Serial Peripheral Interface
SoC	System on Chip
Tx	Transmitter
VW	Virtual Wire

1. Introduction

This document contains all the information about the Lattice Enhanced Serial Peripheral Interface (eSPI) Target IP. As this IP is compliant with Intel eSPI specifications, details about the specification are not discussed in this document.

For the Intel eSPI specifications, you can refer to Enhanced Serial Peripheral Interface (eSPI) Interface Base Specification (for Client and Server Platforms).

1.1. Overview of the IP

eSPI Target IP is compliant with the Intel eSPI specifications. It has its own virtual wire channel in the user interface while implementing peripheral channels, namely, Out of Band (OOB) Message Channel and Flash Access Channel in FIFO that are accessible by the Advanced Peripheral Bus (APB) or Advanced High-Performance Bus – Lite (AHB-Lite) interface.

1.2. Quick Facts

Table 1.1. Summary of eSPI Target IP

IP Requirements	Supported FPGA Family	MachXO5™-NX, Mach™-NX, MachXO3D™, MachXO3™ , and MachXO2™
	IP Changes	For a list of changes to the IP, refer to the eSPI Target IP Release Notes (FPGA-RN-02002).
Resource Utilization	Targeted Devices	LFMXO5, LFMNX, LAMXO3D, LCMXO3D, LAMXO3LF, LCMXO3LF, LCMXO3L, and LCMXO2
	Supported User Interface	APB, AHB-Lite, Virtual Wire Interface
	Resources	Refer to Table A.1, Table A.2, Table A.3, and Table A.4.
Design Tool Support	Lattice Implementation	IP v.2.2.0 – Lattice Radiant™ Software 2023.2 or later, Lattice Propel™ Builder Software 2023.2 or later
	Synthesis	Lattice Synthesis Engine (LSE) Synopsys® Synplify Pro for Lattice
	Simulation	For a list of supported simulators, see the Lattice Radiant Software User Guide.

1.3. IP Support Summary

Table 1.2. eSPI Target IP Support Readiness

Device Family	IP	Rank	eSPI Data Width	Data Rate (Mbps)	Radiant Timing Model	Hardware Validated
MachXO5-NX	eSPI Target	Single	X1	7 to 126	Preliminary	Yes
		Dual	X2	15 to 132	Preliminary	Yes
		Quad	X4	33 to 265	Preliminary	Yes

1.4. Features

Key features of the eSPI Target IP include:

- eSPI Base Specification Revision 1.0 features
  - Supports all eSPI commands except Short Read commands.
  - Supports all required error detection in eSPI specification.
  - Supports Single, Dual, and Quad SPI mode.
  - Cyclic Redundancy Check (CRC)
  - No response error detection in eSPI command
  - Fatal error detection in eSPI command

- Supports APB/AHB-Lite user interface.
- Supports peripheral channel transactions controlled by the host using the APB/AHB-Lite interface.
- Supports OOB message channel transactions controlled by the host using the APB/AHB-Lite interface.
- Supports flash access channel transactions controlled by the host using the APB/AHB-Lite interface.
- Simple implementation interface for virtual wire channel transactions
- General purpose input/output (GPIO) expander interface for virtual wire channel transactions
- Soft resets for configuration and status registers (CSR), Serial Peripheral Interface(SPI), and FIFO

Note: The following features are not supported in this IP:

- Optional non-fatal error detection of the eSPI target
- Soft reset by writing zero to the enable bit of the channels. Refer to the Channel Capabilities and Configurations section.
- Peripheral Short Read commands. This IP version responds with NON-FATAL ERROR response to peripheral channel Short Read commands.

1.5. Licensing and Ordering Information

The eSPI Target 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 board used.

1.7. Minimum Device Requirements

There is no limitation in device speed grade for the use of eSPI Target IP. See the maximum clock frequency in the Clocking section for more details.

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, asserted when value is logic 0.
- _i are input signals.
- _o are output signals.
- _io are bidirectional signals.

2. Functional Description

2.1. IP Architecture Overview

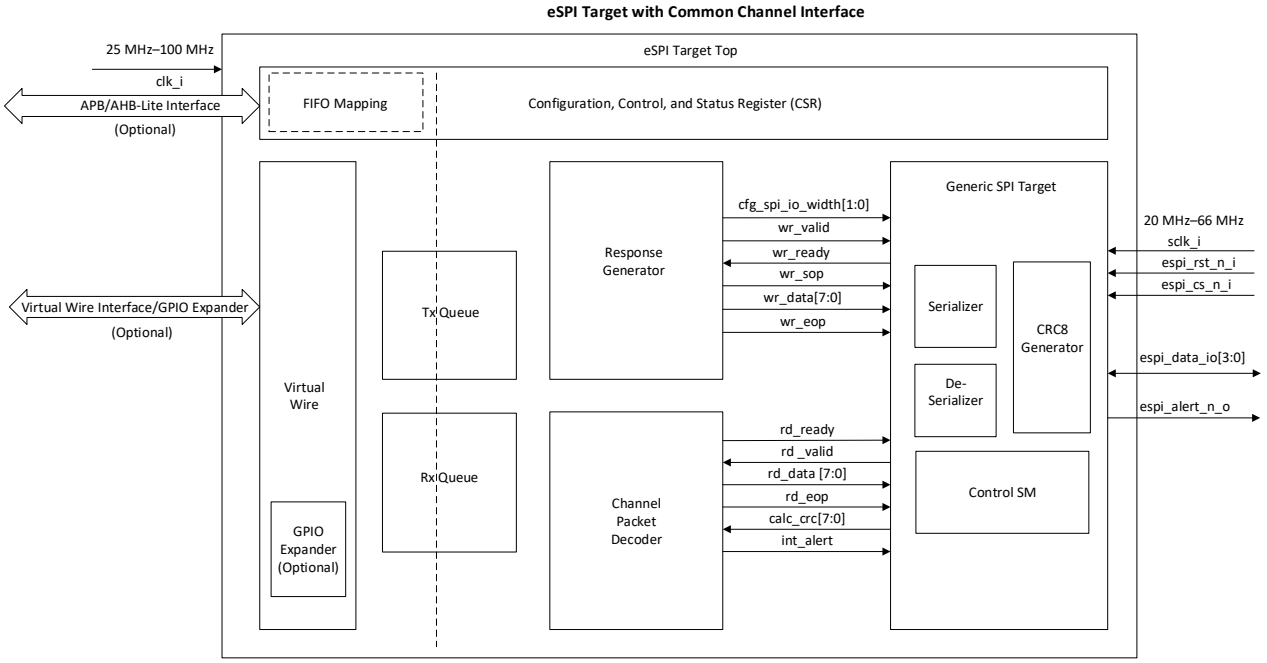

Figure 2.1. Lattice ESPI Target X4 Core Block Diagram

The eSPI Target IP includes the following components:

- APB/AHB-Lite Interface
- Virtual Wire Channel
- CSR
- Response Generator
- Channel Packet Decoder
- Generic SPI Target

2.2. Clocking

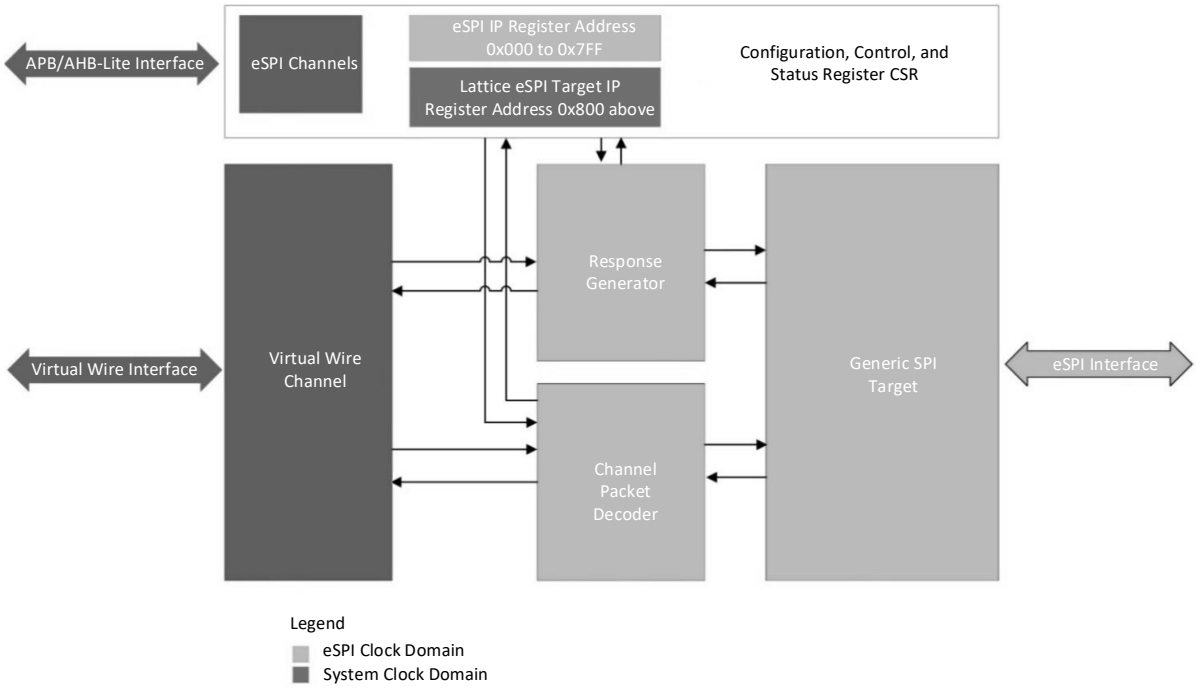

Figure 2.2. eSPI Target IP Clock Domain Block Diagram

2.2.1. Clocking Overview

- System Clock: clk_i
  - Supported frequency range: 25–100 MHz
  - System Clock must be faster than eSPI Clock.
- eSPI Clock: espi_clk_i
  - Supported frequency range: 20–66 MHz

2.3. Reset

There are two resets for the eSPI Target IP. One is eSPI active-low reset, espi_reset_n. The other is system active-low reset, reset_n.

2.3.1. Reset Overview

Both resets are asynchronous reset. The reset assertion can be asynchronous but reset deassertion is synchronized inside the eSPI Target IP. When deasserted, 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
CSR Interface	AHB-Lite	The AHB-Lite interface is used in this IP for register and memory access. See Signal Description and Register Description for more details. For AHB-Lite interface, refer to AMBA 3 AHB-Lite Protocol Specification for information and timing diagram of the AHB-Lite interface. • Write transaction has no wait state; • Read transaction has one to two wait states.
	APB	The APB interface is used in this IP for register and memory access. See Signal Description and Register Description for more details. For APB Interface, refer to AMBA 3 APB Protocol Specification for information and timing diagram of the APB interface.
Virtual Wire Interface	Simple Implementation	The simple implementation supports virtual wire feature with minimal usage of I/O ports in the eSPI Target IP. For more information, refer to Virtual Wire – Simple Implementation.
	GPIO Expander	The GPIO expander virtual wire interface implements the GPIO expander feature of the eSPI Target specification. Enabling this feature requires more resource than the simple implementation. For more details, refer to Virtual Wire – GPIO Expander.
Device Receiver/Transmitter Interface	eSPI	The eSPI interface can receive and send using all the eSPI commands stated in the Intel eSPI specifications.

2.5. Blocks

2.5.1. AHB-Lite/APB

2.5.1.1. APB

The APB target in this IP is compliant with the AMBA APB Protocol Specification.

2.5.1.2. AHB-Lite

The AHB-Lite in this IP only supports single burst type.

Table 2.2. AHB-Lite Signal

Signal Name	Direction	Description
HCLK	In	AHB-Lite signal timing
HRESETn	In	Active-low reset
HADDR[31:0]	In	Read/Write address
HBURST[2:0]	In	Indicates the burst type. Not supported in this IP. Not supported as the IP is always using single burst.
HMASTLOCK	In	Indicates that the current transaction is part of a locked sequence. Not supported in this IP.
HPROT[3:0]	In	The protection control signal. Not supported in this IP.
HSIZE[2:0]	In	Indicates the size of the transfer.

Signal Name	Direction	Description
HTRANS[1:0]	In	Indicates the transfer type of the current transfer.
HWDATA[31:0]	In	Write data
HWRITE	In	Indicates if the current transaction is write or read.
HRDATA[31:0]	Out	Read data
HREADYOUT	Out	Indicates that transfer is finished.
HRESP	Out	The transfer response. 0 indicates an OKAY response, which means the transfer is successful. Meanwhile, 1 indicates that an error occurs during the transfer.
HSELx	In	AHB target select bit sent by the controller

2.5.2. CSR

CSR contains the configuration and status registers, which are written either by the system controller through the AHB-Lite, APB interface, or by the eSPI controller through the eSPI bus. The access type for every register is different for the AHB-Lite, APB interface, and the eSPI controller. More details on access type are discussed in the Other IP Specific Blocks/Layers/Interfaces section.

2.6. Other IP Specific Blocks/Layers/Interfaces

2.6.1. eSPI Commands

All the eSPI commands in the Intel eSPI specification are supported in the Lattice eSPI Target IP, except the Short Read commands. For details of eSPI commands timing and specification, you can refer to the Intel eSPI specification document.

For Short Read commands, the eSPI Target IP responds with the NON-FATAL ERROR response.

2.6.2. Virtual Wire – Simple Implementation

The Virtual Wire Simple Implementation is an additional feature in the Lattice eSPI Target IP to allow access to the other Virtual Group Index without using additional resources for the additional I/O ports.

2.6.2.1. Virtual Wire IN

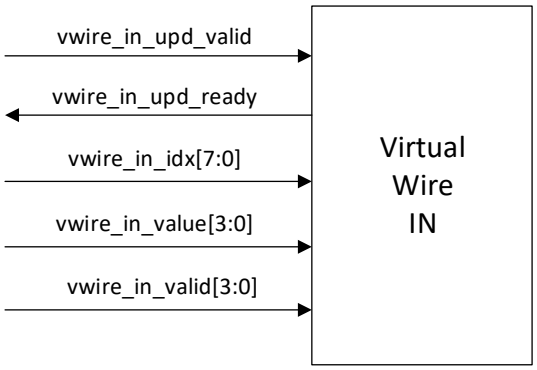

Figure 2.3. Virtual Wire IN Interface

The Virtual Wire IN interface (Figure 2.3) is in the system clock domain.

Below are the instructions on how to use the Virtual Wire IN interface and read in eSPI using the GET VW command.

In Virtual Wire In Interface

1. Drive vwire_in_upd_valid to 1. While this signal is high, the Virtual Wire IN interface samples all vwire inputs every system clock posedge.
2. Drive vwire_in_idx to the virtual wire index.
3. Drive vwire_in_value[3:0] and vwire_in_valid[3:0]. vwire_in_value bits hold the value of the virtual wire data while the corresponding bit in vwire_in_valid indicates if vwire_in_valid changes.

In eSPI Interface

1. Wait for the alert signal from the eSPI Target.
2. Use the eSPI SET STATUS command in address 0x020 bit [0] to enable the Virtual Wire Channel Enable bit. Skip this step when the specified register bit is already set to 1.
3. Use the eSPI GET CONFIG command in address 0x020 bit [1] to check if Virtual Wire Channel Ready is asserted.
4. Use the eSPI GET STATUS command and check if VWIRE Avail is high. If the response modifier is enabled, GET VWIRE appends to this GET STATUS transaction. See virtual wire packet format in Figure 41 of Enhanced Serial Peripheral Interface (eSPI) Interface Base Specification.
5. Use the eSPI GET VWIRE command. Repeat until VWIRE Avail in the eSPI status is low.

2.6.2.2. Virtual Wire OUT

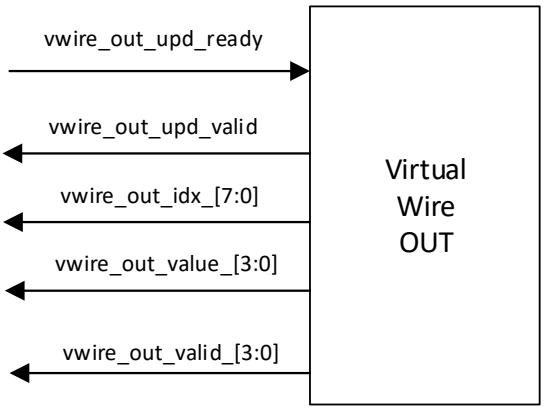

Figure 2.4. Virtual Wire OUT Interface

The Virtual Wire OUT interface (Figure 2.4) is in the system clock domain.

Below are the instructions on how to use the Virtual Wire IN interface and read in eSPI using the GET VW command.

In eSPI Interface

1. Use the eSPI GET STATUS command to check if VWIRE FREE is asserted.
2. Use the eSPI SET STATUS in address 0x020 bit[0] to enable the Virtual Wire Enable bit.
3. Use the eSPI GET CONFIG in address 0x020 bit [1] to check if Virtual Wire Channel Ready is asserted.
4. The eSPI PUT VWIRE command is 8'h04. See virtual wire packet format in Figure 41 of the Enhanced Serial Peripheral Interface (eSPI) Interface Base Specification.

In VW OUT Interface

1. Wait for the assertion of vwire_out_upd_valid.
2. Read vwire_out_idx[7:0], vwire_out_value[3:0], and vwire_out_valid[3:0].
3. Drive vwire_out_upd_ready after reading the Virtual Wire OUT data.
4. Repeat until vwire_out_upd_valid deasserts to zero. When vwire_out_upd_valid is zero, it means there is no valid data from the virtual wire.

2.6.3. Virtual Wire – GPIO Expander

The GPIO Expander is compliant with the Intel eSPI specification. In the Lattice eSPI target IP, general-purpose I/Os are assigned automatically to the Virtual Wire Index. d128 is the minimum index and d255 is the maximum index.

The size of GPIO input and output is configurable in the parameter GUI. The priority of assigning GPIO to virtual wire index is also configurable in the GUI.

An example of GPIO to virtual wire index assignment is shown below.

If GPIO Input First is selected in the GUI, general-purpose inputs are assigned first and general-purpose outputs assigning starts at the next index after assigning all inputs (Figure 2.5).

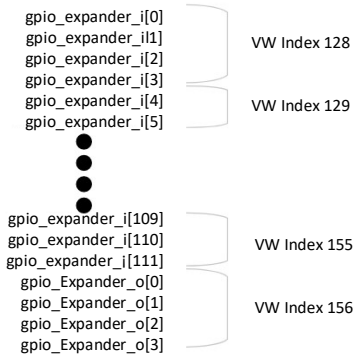

Figure 2.5. Select GPIO Input First

If GPIO Output First is selected in the GUI. General-purpose outputs are assigned first and general-purpose inputs assigning starts at the next index after assigning all outputs (Figure 2.6).

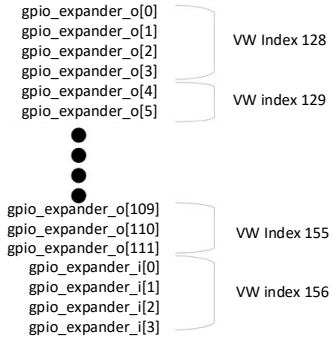

Figure 2.6. Select GPIO Output First

If the input or output count is not divisible by four, it still utilizes one virtual wire index in the most significant bits.

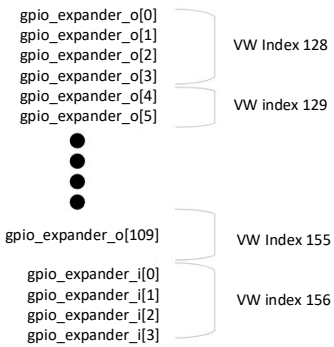

Figure 2.7. Input or Output Not Divisible by Four

2.6.3.1. GPIO Expander Input

In this section, the term GPIO Group is used to describe GPIOs that are assigned to the same virtual wire index.

When using the GPIO Expander input, you must change the value of the GPIO Expander input bit to trigger the GPIO IN to send virtual wire data to the eSPI controller. Changing one bit in a GPIO Group is enough to send virtual wire data of the GPIO Group to the Virtual Wire channel. The difference is unchanged GPIO inputs have zero value in their assigned virtual wire valid bit to indicate that the virtual wire data bit is not changed.

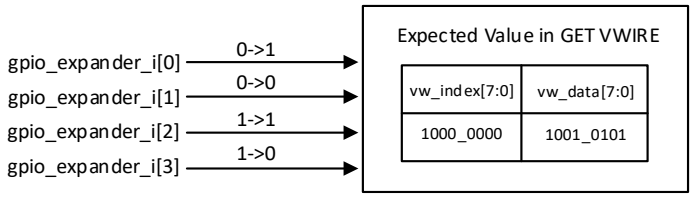

Figure 2.8. GPIO Expander Input

In Figure 2.8, the vw_data[7:4] is the value after driving the GPIO Group while the vw_data_level[3:0] indicates if it changes. In the example above, the value of the vw_index 8'h80 is used to show that the virtual wire index is changed.

2.6.3.2. GPIO Expander Output

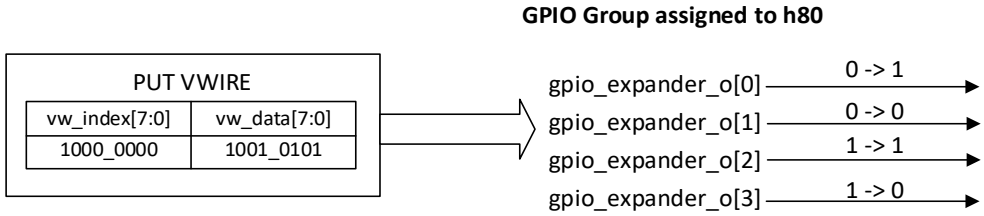

Figure 2.9. GPIO Expander Output

In the GPIO Expander, Virtual Wire data bits with zero virtual wire valid bits are not reflected in the GPIO Expander output bits, in compliance with the eSPI Specification (Figure 2.9). In the example above, the virtual wire index 8'h80 is the index assigned to gpio_expander_o[3:0].

Changes in the GPIO Expander outputs are only reflected after the eSPI controller stops driving low-active eSPI Chip Select# (Figure 2.10).

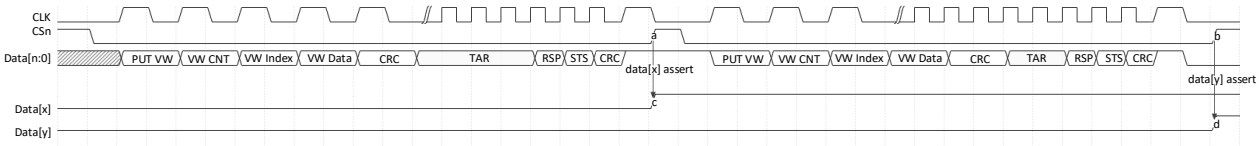

Figure 2.10. PUT VWIRE Interrupt Assertion Timing

2.6.4. eSPI Avail Status Valid Register Bits

The eSPI Target IP has Avail Status valid register bits, tx_avail_valid (0x808 [29:24]) and tx_avail_order (0x810). tx_avail_order (0x810) shows the order of the status assertion of PC AVAIL (3'd0), NP AVAIL (3'd1), OOB AVAIL (3'd2), FLASH NP AVAIL (3'd3), and FLASH C AVAIL(3'd4).

Example: If you intend to transmit VWIRE, OOB_AVAIL, and NP sequentially, the tx_free_valid bits [5:0] should be set to 6'b000111 to indicate the three available packets. Then, tx_free_order should be set to {9'd0, 3'd1, 3'd2, 3'd4}.

The IP first asserts the VWIRE AVAIL status since it is the first in the order. Once GET VWIRE is received and transmission is completed, the VWIRE AVAIL status is cleared and OOB AVAIL is asserted. The tx_avail_valid and tx_avail_order should shift and show 6'b0011, and {12'd0, 3'd1, 3'd2} respectively. The process continues until all packets are transmitted and tx_avail_valid shows 6'b000000.

2.6.5. eSPI Free Status Valid Register Bits

The eSPI Target IP has Free Status valid register bits, rx_free_valid (0x808 [19:16]) and rx_free_order (0x810). rx_free_order (0x810) shows which of the status is asserted: PC_FREE(2'd0), NP_FREE(2'd1), OOB_FREE(2'd2), and FLASH_NP_FREE(2'd3).

Example: If you are able to receive PC, FLASH_NP, and NP, the rx_free_valid[3:0] should be set to 4'b0111 to indicate the three free allocation. Then, rx_free_order should be set to {don't care, 2'd3, 2'd1, 2'd0}.

The IP asserts PC_FREE, NP_FREE, and FLASH_NP_FREE in the status during response. Once a packet is received, for example, PUT_PC, the corresponding status PC_FREE is cleared. The corresponding rx_free_valid bit should also be cleared. The process continues until all packets are received and rx_free_valid shows 4'b0000. If the Rx FIFO cannot receive a new packet, then the status sent during response is cleared and is only asserted once the Rx FIFO can accept a new packet.

2.6.6. Program Flow for Writing Data to FIFO

Writing data to FIFO is used when the microcontroller is sending data to the eSPI controller and the controller receives the data using GET <CHANNEL> eSPI commands. While writing to the FIFO, the payload header must be sent first, then, the payload data is sent. After writing data to the FIFO, the microcontroller needs to write to control the eSPI target status to assert the AVAIL register bit for their respective channel depending on the transaction.

In the APB/AHB-Lite Interface

1. Write the header and payload to the Tx FIFO Queue Write register. The sequence of the data write is payload header first, then, the payload data.
2. Write to the Tx Avail Valid register (0x808) and Tx Avail Order register (0x810) to assert the avail signal in the eSPI Status register. See the eSPI Avail Status Valid Register Bits section for more information on Tx Avail Valid and Tx Avail Order registers. See the eSPI Target IP Registers section for more information on specific registers of the eSPI Target IP.

Since the data width of APB/AHB-Lite is limited to 32 bits. The data is sent word by word. For example, the eSPI target is sending a 64-bit request peripheral memory write to the controller. The first word includes the cycle type, tag, length, and address bits 31 to 24. The second word contains the remainder of the address and the first byte of the payload data. The data write to the FIFO is shown in Figure 2.11.

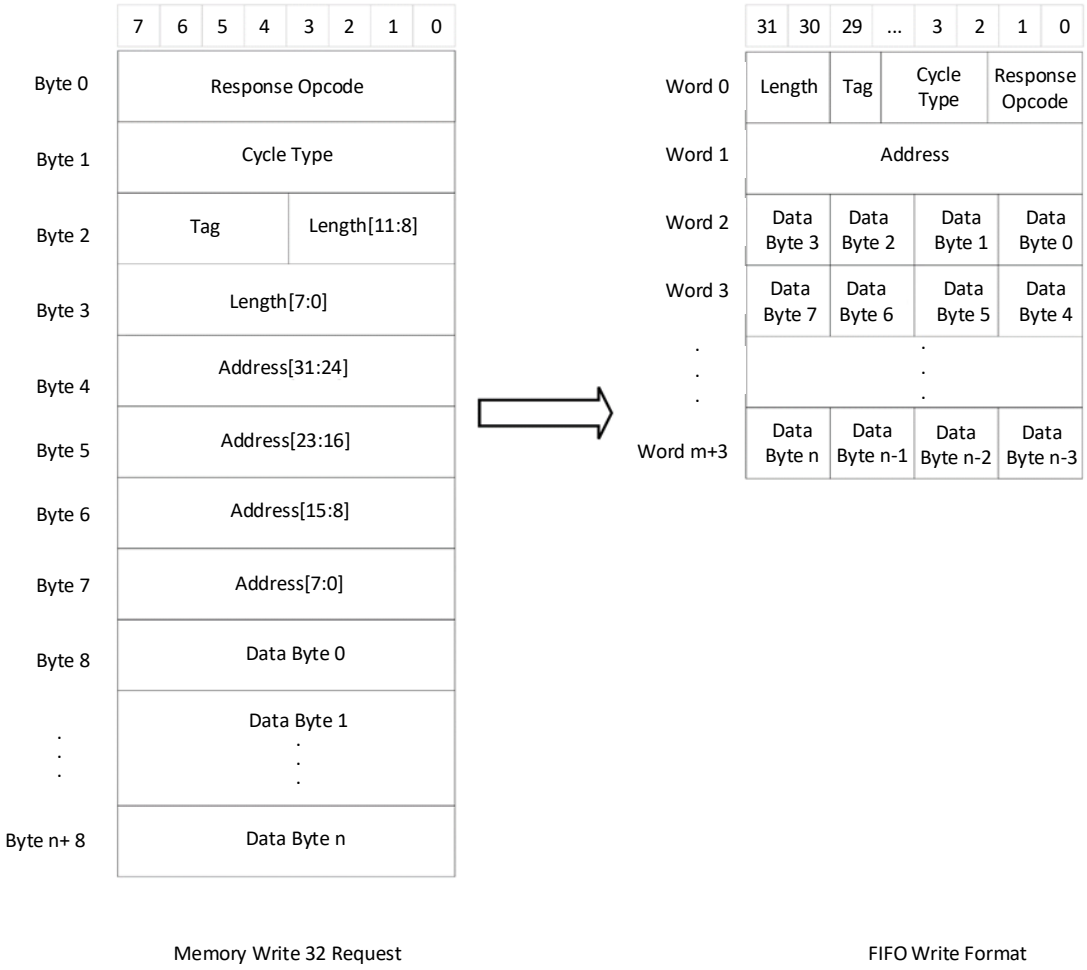

Figure 2.11. FIFO Write Format

2.6.7. Program Flow for Reading Data from FIFO

Reading data from the FIFO is performed when the interrupt status indicates that the FIFO is not empty. The data to be read from the FIFO includes the command opcode, payload header, and optional payload data.

In the APB/AHB-Lite Interface

1. Assert the Free eSPI Status register bits.
2. Write to the Rx Avail Valid register (0x808) and Rx Avail Order register (0x810) to assert the free signal in the eSPI Status register. See the eSPI Free Status Valid Register Bits section for more information on Rx Avail Valid and Rx Avail Order registers. See the eSPI Target IP Registers section for more information on specific registers of the eSPI Target IP.
3. After using GET Channel Command, read the FIFO.
4. Read the header and payload from the RX FIFO Queue Write register. The sequence of the data write is header first, then the payload data.

Since the data width of APB/AHB-Lite is limited to 32 bits. The data is sent word by word. For example, the eSPI controller is sending a 32-bit request to write to the peripheral memory of the target. The first word includes the command opcode, cycle type, tag, and length. The second word contains the address. After that, the payload data, which is optional because there are transactions without message, is sent. The data read to the FIFO is shown in Figure 2.12.

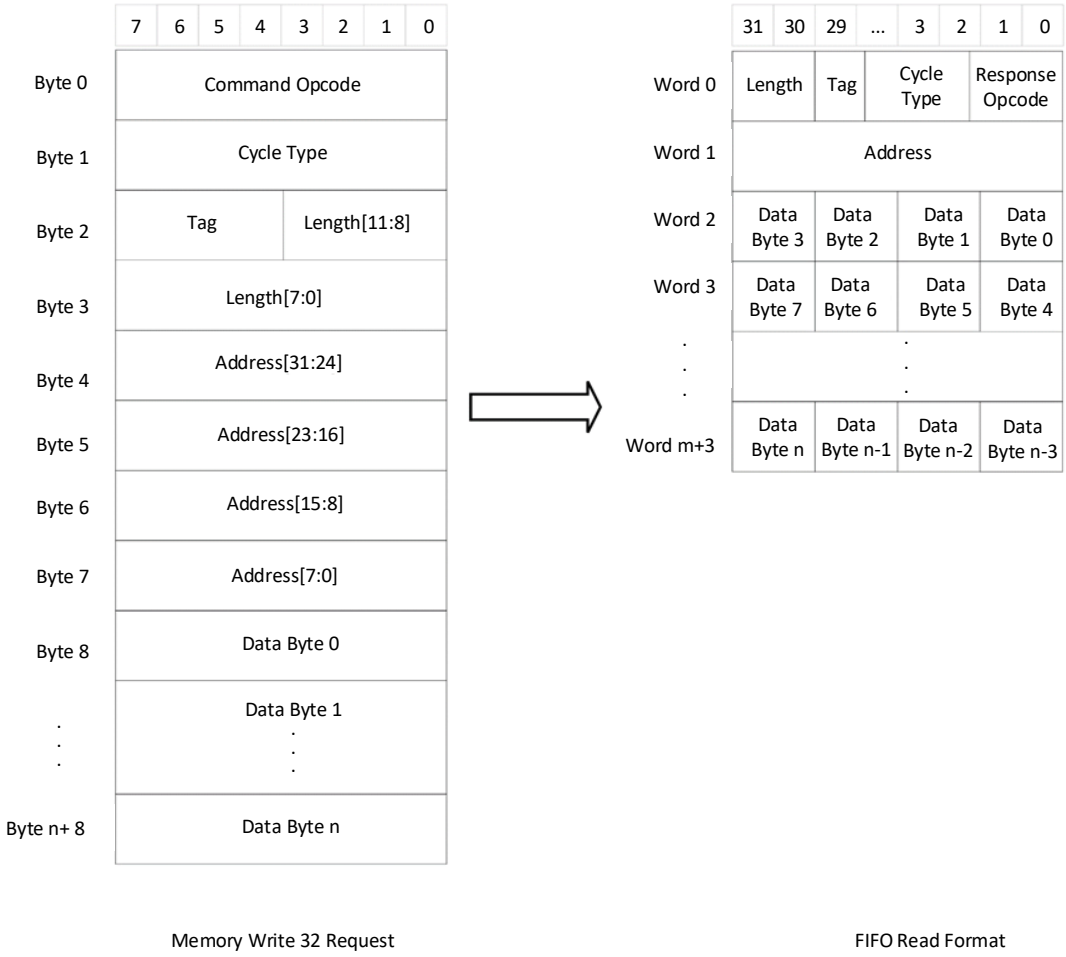

Figure 2.12. FIFO Read Format

2.6.8. Program Flow for Using GET Channel Commands

In this section, the GET Channel commands refer to all eSPI channel commands where the eSPI Controller receives data from the four available channels, Peripheral Channel, Virtual Wire Channel, OOB Message Channel, and Flash Access Channel. Although there may be a large number of FIFO and GET Channel commands, they all follow a general flow: what goes in FIFO IN goes out through the GET Channel commands. Below are examples of executing GET Channel commands in the eSPI Target IP.

For command opcode values of GET Channel commands, see Table 3 Command Opcode Encodings of Enhanced Serial Peripheral Interface (eSPI) Interface Base Specification. For the packet format, see Section 5.2 Channels of the base specification.

2.6.8.1. GET Channel Commands – Peripheral Channel

In this section, the example used is a scenario where the eSPI controller receives a GET PC command of the Completion with Data cycle type. If you use other command opcode, data values in the payload header and payload data change depending on the specified values in the Intel eSPI base specification.

In APB/AHB-Lite Interface

1. Write the payload address and the payload data in the FIFO.
   The format is discussed in the Program Flow for Writing Data to FIFO section.
2. Write to the eSPI Avail Status Valid register to set the PC AVAIL bit. Refer to eSPI Avail Status Valid Register Bits for more details. Additionally, set bit 0 to 1 in the eSPI Target Channel Control 0 register. The bit field is named Peripheral Channel Ready.

In eSPI Interface

1. Wait for the deassertion of the port espi_alert_n_o. If the open-drain alert support is enabled, the name of the port is espi_alert_n_io.
2. Execute the eSPI GET STATUS command.
3. Use the status data from the eSPI GET STATUS command to check if PC AVAIL is asserted.
4. Use the eSPI GET CONFIG command in address 0x010 to save the current value of the Peripheral Channel Configuration register.
5. For the Peripheral Channel, use the eSPI SET CONFIG command in bit 0 of address 0x010 to enable the Peripheral Channel Enable bit. All the other values fetched in the earlier transaction do not change.
6. Use the eSPI GET CONFIG command in the respective channel address you are using. The address is 0x010 for the Peripheral Channel. Check the value of bit 1 to check if the Channel Ready bit is asserted.
7. Initiate the GET PC command.

Figure 2.13 shows the timing diagram of the eSPI GET PC transaction.

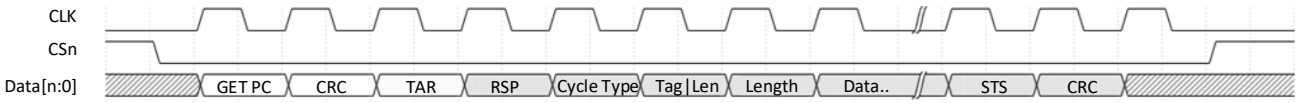

Figure 2.13. GET PC – Completion with Data

2.6.8.2. GET Channel Commands – Virtual Wire Channel

For the Virtual Wire Channel, there are no additional steps required. There is a dedicated interface for this channel, which means the virtual wire channel does not use the FIFO to read the virtual wire data. For more information, refer to the Virtual Wire Channel section.

In Virtual Wire Interface

- If Simple Interface is used:
  a. Write value in vwire_in_value[3:0], vwire_in_valid[3:0], and vwire_in_idx[7:0].
  b. Check if vwire_in_upd_ready is HIGH.
  c. Assert the vwire_in_upd_valid input port.
- If GPIO Expander is used:
  Change the value of gpio_expander_i[GPIO_IN_COUNT-1:0].

In the APB/AHB-Lite Interface

Write the value 1 to bit 1 of the eSPI Target Channel Control 0 register. The bit field is named Virtual Wire Channel Ready.

In eSPI Interface

1. Wait for the deassertion of the port espi_alert_n_o. If the open-drain alert support is enabled, the name of the port is espi_alert_n_io.
2. Execute the eSPI GET STATUS command.
3. Use the status data from the eSPI GET STATUS command to check if VW AVAIL is asserted.
4. Use the eSPI GET CONFIG command in address 0x020 to save the current value of the Virtual Wire Channel configuration register.
5. For the Virtual Wire Channel, use the eSPI SET STATUS command in bit 0 of address 0x020 to enable the Virtual Wire Channel Enable bit. All the other bits in the register retain the value fetched from the transaction earlier.
6. Use the eSPI GET CONFIG command in the respective channel address you are using. The address is 0x020 for Virtual Wire Channel. Read the value of bit 1 to check if the Channel Ready bit is asserted.
7. Initiate the GET VWIRE command.

Figure 2.14 shows the timing diagram of the eSPI GET VWIRE transaction.

Figure 2.14. GET VWIRE

2.6.8.3. GET Channel Commands – OOB Message Channel

The only valid cycle type for GET OOB is the OOB Tunneled SMBus Message. This cycle type is used for the programming flow discussed in this section.

In APB/AHB-Lite Interface

1. Write the payload address and the payload data in the FIFO. The format is discussed in the Program Flow for Writing Data to FIFO section.
2. Write to the eSPI Avail Status Valid register to set the PC AVAIL bit. Refer to eSPI Avail Status Valid Register Bits for more details. Additionally, set bit 2 to 1 in the eSPI Target Channel Control 0 register. The bit field is named OOB Message Channel Ready.

In eSPI Interface

1. Wait for the deassertion of the port espi_alert_n_o. If the open-drain alert support is enabled, the name of the port is espi_alert_n_io.
2. Execute the eSPI GET STATUS command.
3. Use the status data from the eSPI GET STATUS command to check if OOB AVAIL is asserted.
4. Use the eSPI GET CONFIG command in address 0x030 to save the current value of the OOB Message Channel Configuration register.
5. For the OOB Message Channel, use eSPI SET STATUS command in bit 0 of address 0x030 to enable the OOB Message Channel Enable bit. Other bits retain the value fetched from the transaction earlier.
6. Use the eSPI GET CONFIG command in the respective channel address you are using. The address is 0x030 for the OOB Message Channel. Read the value of bit 1 to check if the Channel Ready bit is asserted.
7. Initiate the GET OOB command.

Figure 2.15 shows the timing diagram of the eSPI GET OOB transaction.

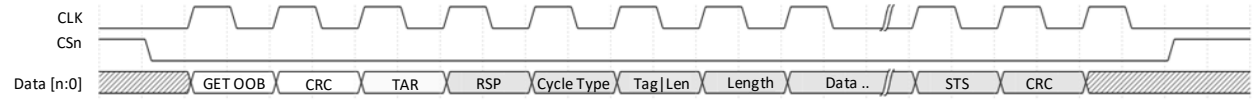

Figure 2.15. GET OOB

2.6.8.4. GET Channel Commands – Flash Access Channel

In this section, the program flow shows a transaction with the Flash Write cycle type for the GET FLASH NP command opcode.

In APB/AHB-Lite Interface

1. Write the payload address and the payload data in the FIFO.
   The format is discussed in the Program Flow for Writing Data to FIFO section.
2. Write to the eSPI Avail Status Valid register bit to set the PC AVAIL bit.
   Refer to the eSPI Avail Status Valid Register Bits section for more details. Additionally, set bit 3 to 1 in the eSPI Target Channel Control 0 register. The bit field is named Flash Access Channel Ready.

In eSPI Interface

1. Wait for the deassertion of the port espi_alert_n_o. If the open-drain alert support is enabled, the name of the port is espi_alert_n_io.
2. Execute the eSPI GET STATUS command.
3. Use the status data from the eSPI GET STATUS command to check if FLASH NP AVAIL is asserted.

4. Use the eSPI GET CONFIG command in address 0x040 to save the current value of the Flash Access Channel Configuration register.
5. For Flash Access Channel, use the eSPI SET STATUS command in bit 0 of address 0x040 to enable the Flash Access Channel Enable bit. Other bits retain the value fetched from the transaction earlier.
6. Use the eSPI GET CONFIG command in the respective channel address you are using. The address is 0x040 for the Flash Access Channel. Read the value of bit 1 to check if the Channel Ready bit is asserted.
7. Initiate the GET FLASH NP command.

Figure 2.16 shows the timing diagram of the GET FLASH NP transaction.

Figure 2.16. GET FLASH NP – Flash Write

2.6.9. Program Flow for Using PUT Channel Commands

In this document, PUT Channel commands refer to all eSPI channel commands where the eSPI controller sends data to the four available channels, Peripheral Channel, Virtual Wire Channel, OOB Message Channel, and Flash Access Channel. Although there may be a large number of FIFO and PUT Channel commands, they all follow a general flow: what goes in through the GET Channel commands goes out in the FIFO OUT. Below are examples of executing PUT Channel commands in the eSPI Target IP.

2.6.9.1. PUT Channel Commands – Peripheral Channel

In APB Interface

Write to the eSPI Avail Status Valid register bit to set the PC FREE bit. See the eSPI Free Status Valid Register Bits section for more details. Additionally, set bit 0 to 1 in the eSPI Target Channel Control 0 register. The bit field is named Peripheral Channel Ready.

In the eSPI Interface

1. Wait for the deassertion of the port espi_alert_n_o. If the open-drain alert support is enabled, the name of the port is espi_alert_n_io.
2. Execute the eSPI GET STATUS command.
3. Use the status data from the eSPI GET STATUS command to check if PC Free is asserted.
4. Use the eSPI GET CONFIG command in the address 0x010 to save the current value of the Peripheral Channel Configuration register.
5. For the Peripheral Channel, use the eSPI SET CONFIG command in address 0x010 bit[0] to enable Peripheral Channel Enable bit. All the other values fetched in the earlier transaction do not change.
6. Use the eSPI GET CONFIG command in the respective channel address you are using. The address is 0x010 for the Peripheral Channel. Check the value of bit 1 to check if the Channel Ready bit is asserted.
7. Initiate the PUT PC command.

Figure 2.17 below shows the timing diagram of PUT PC transaction.

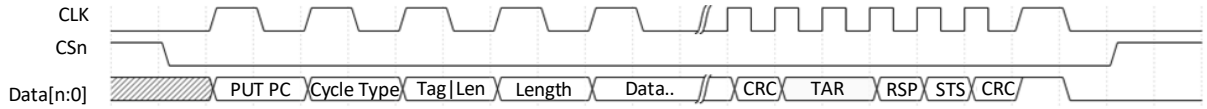

Figure 2.17. PUT PC Command

In APB Interface

Read the data in the FIFO. The format is discussed in the Program Flow for Writing Data to FIFO section.

2.6.9.2. PUT Channel Commands – Virtual Wire Channel

In the eSPI Status register, VW FREE is always 1. There is no need to assert the FREE bit for the virtual wire.

In APB/AHB-Lite Interface

Write the value 1 to bit 1 of the eSPI Target Channel Control 0 register. The bit field is named Virtual Wire Channel Ready.

In eSPI Interface

1. Wait for the deassertion of the port espi_alert_n_o. If the open-drain alert support is enabled, the name of the port is espi_alert_n_io.
2. Execute the eSPI GET STATUS command.
3. Use the status data from the eSPI GET STATUS command to check if VW AVAIL is asserted.
4. Use the eSPI GET CONFIG command in address 0x020 to save the current value of the Virtual Wire Channel Configuration register.
5. For the Virtual Wire Channel, use the eSPI SET STATUS command in bit 0 of address 0x020 to enable the Virtual Wire Channel Enable bit. All the other bits in the register retain the value fetched from the transaction earlier.
6. Use the eSPI GET CONFIG command in the respective channel address you are using. The address is 0x020 for the Virtual Wire Channel. Read the value of bit 1 to check if the Channel Ready bit is asserted.
7. Initiate the PUT VWIRE command.

   Figure 2.18 shows the timing diagram of the eSPI PUT VWIRE transaction.

Figure 2.18. PUT VWIRE

In Virtual Wire Interface

- If Simple Interface is used:
  a. Check if vwire_out_upd_valid is high.
  b. Read value in vwire_out_value[3:0], vwire_out_valid[3:0], and vwire_out_idx[7:0].
  c. Assert the vwire_out_upd_ready input port. This triggers the signals mentioned in the second step to display the next virtual wire value.
- If GPIO Expander is used:

  The values are displayed in gpio_expander_o[GPIO_OUT_COUNT-1:0].

2.6.9.3. PUT Channel Commands – OOB Message Channel

In APB Interface

Write to eSPI Avail Status Valid register bit to set the OOB FREE bit. Refer to eSPI Free Status Valid Register Bits for more details. Additionally, set bit 2 to 1 in the eSPI Target Channel Control 0 register. The bit field is named OOB Message Channel Ready.

In eSPI Interface

1. Wait for the deassertion of the port espi_alert_n_o. If the open-drain alert support is enabled, the name of the port is espi_alert_n_io.
2. Execute the eSPI GET STATUS command.
3. Use the status data from the eSPI GET STATUS command to check if OOB FREE is asserted.
4. Use the eSPI GET CONFIG command in address 0x030 to save the current value of the OOB Message Channel configuration register.
5. For the OOB Message Channel, use the eSPI SET STATUS command in bit 0 of the address 0x030 to enable the OOB Message Channel Enable bit. Other bits retain the value fetched from the transaction earlier.
6. Use the eSPI GET CONFIG command in the respective channel address you are using. The address is 0x030 for the OOB Message Channel. Read the value of bit 1 to check if the Channel Ready bit is asserted.
7. Initiate the PUT OOB command.

Figure 2.19 shows the timing diagram of the eSPI PUT OOB transaction.

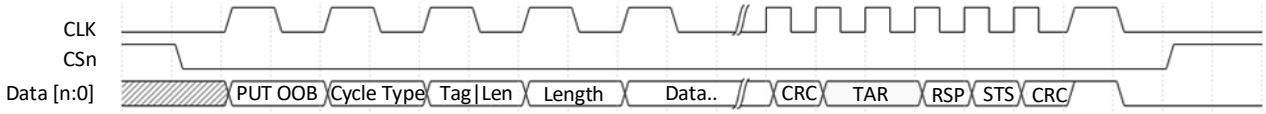

Figure 2.19. PUT OOB – Completion with data

In APB Interface

Read the data in the FIFO. The format is discussed in the Program Flow for Writing Data to FIFO section.

2.6.9.4. PUT Channel Commands – Flash Access Channel

In the APB Interface

Write to the eSPI Avail Status Valid register to set the FLASH C FREE bit. Refer to eSPI Free Status Valid Register Bits for more details. Additionally, set bit 3 to 1 in the eSPI Target Channel Control 0 register. The bit field is named Flash Access Channel Ready.

In eSPI Interface

1. Wait for the deassertion of the port espi_alert_n_o. If the open-drain alert support is enabled, the name of the port is espi_alert_n_io.
2. Execute the eSPI GET STATUS command.
3. Use the status data from eSPI GET STATUS command to check if FLASH C Free is asserted.
4. Use the eSPI GET CONFIG command in address 0x040 to save the current value of the Flash Access Channel configuration register.
5. For Flash Access Channel, use the eSPI SET STATUS command in address 0x040 bit 0 to enable the Flash Access Channel Enable bit. Other bits retain the value fetched from the transaction earlier.
6. Use the eSPI GET CONFIG command in the respective channel address you are using. The address is 0x040 for the Flash Access Channel. Read the value of bit 1 to check if the Channel Ready bit is asserted.
7. Initiate the PUT FLASH C command.

Figure 2.20 shows the timing diagram of the PUT FLASH C transaction.

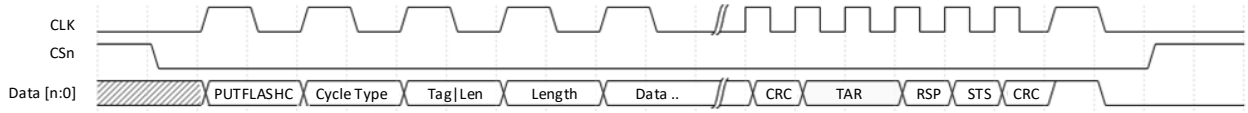

Figure 2.20. PUT FLASH C – Completion with Data

In the APB Interface

Read the data in the FIFO. The format is discussed in the Program Flow for Writing Data to FIFO section.

2.7. Error Handling

Error	Description	Supported	Required
Invalid Command Opcode	NO_RESPONSE Response Code. The command is discarded.	Yes	Yes
Invalid Cycle Type with respect to specific command	NO_RESPONSE Response Code. The command is discarded.	Yes	Yes
Command phase CRC Error	NO_RESPONSE Response Code. The command is discarded.	Yes	Yes
Unexpected deassertion of Chip Select#	Target tri-state the bus tSHQZ after Chip Select# is deasserted. **Note:** The controller is expected to detect CRC errors during the response phase if CRC checking is enabled.	Yes	Yes
Protocol Error • PUT without FREE • GET without AVAIL	FATAL_ERROR Response Code. The command is discarded.	Yes	Yes
Malformed Packet during Command Phase Peripheral Channel: • Payload length > Max Payload Size aligned • Read request size > Max Read Request Size aligned • (Address + Length) crosses 4 KB aligned boundary Virtual Wire Channel: Virtual Wire Count > Max Virtual Wire Count OOB Channel: SMBus Byte Count > Max Payload Size Flash Access Channel: • Payload length > Max Payload Size • Read request size > Max Read Request Size	FATAL_ERROR Response Code. The command is discarded. or FATAL_ERROR Virtual Wire. Before signalling the FATAL ERROR Virtual Wire, the transaction is completed on the eSPI bus with the ACCEPT RESPONSE and the following outcomes: Posted: The command is discarded. Completion: The command is discarded. Non-posted: Unsuccessful completion without data is returned and the command is discarded. Virtual Wire: The command is discarded.	Yes	Yes
PUT_MEMRD32_SHORTS PUT_IORD_SHORT	NON_FATAL_ERROR Response Code. It is not supported in this eSPI Target IP.	Yes	Optional

3. IP Parameter Description

The configurable attributes of the eSPI Target IP are shown in Table 3.1 and Table 3.2. You can configure the IP by setting the attributes accordingly in the IP Catalog's Module/IP wizard of the Lattice Radiant software.

Wherever applicable, default values are in bold.

3.1. General

Table 3.1. General Attributes

Attribute	Selectable Values	Description	Parameter
Remove Tristate	**0**,1	Parameter for removing tristate input/output of eSPI data.	REMOVE_TRISTATE 0 – Tristate eSPI Data I/O. 1 – Remove tristate eSPI Data I/O.
Enable Chip Select Glitch Filter	**0,**1	Parameter for enabling glitch filter for chip select.	EN_CS_FILTER 0 – Disabled. 1 – Enabled.
Select User Interface	**APB**, AHBL, none	Parameter for selecting the SoC Interface. If None is selected, Virtual Wire must be enabled.	USER_INTERFACE_SELECT APB – APB User Interface. AHBL – AHB-Lite User Interface. None – no available User Interface.
Peripheral Channel Enable	**0**,1	Parameter for peripheral channel enable/disable.	PERIPHERAL_CHANNEL_ENABLE 0 – Disable the eSPI Peripheral Channel. 1 – Enable the eSPI Peripheral Channel.
Virtual Wire Channel Enable	**0**,1	Parameter for VW channel enable/disable. It is automatically enabled when the USER_INTERFACE_SELECT value is None.	VIRTUAL_WIRE_INTERFACE_SELECT 0 – Disable eSPI Virtual Wire Channel. 1 – Enable eSPI Virtual Wire Channel.
OOB Message Channel Enable	**0**, 1	Parameter for OOB Message channel enable/disable.	OOB_MESSAGE_CHANNEL_ENABLE 0 – Disable eSPI OOB Message Channel. 1 – Enable eSPI OOB Message Channel.
Flash Access Channel Enable	**0**, 1	Parameter for Flash access channel enable/disable.	FLASH_CHANNEL_ENABLE 0 – Disable eSPI Flash Access Channel. 1 – Enable eSPI Flash Access Channel.
eSPI Target Configuration Programmable	**0**,1	Parameter to configure if address 0x804 is programmable by User Interface.	ESPI_TARGET_CONFIG_PROG 0 – Not Programmable. 1 – Programmable.
PERIPHERAL_CHANNEL_ENABLE = 0 and OOB_MESSAGE_CHANNEL_ENABLE = 0 and FLASH_CHANNEL_ENABLE = 0			
Enable eSPI Channel Configurations Registers	**0**,1	Parameter of instantiating channel configuration registers when the eSPI channel is disabled.	EN_ALL_REGISTERS 0 – Disabled. 1 – Enabled.
PERIPHERAL_CHANNEL_ENABLE = 0 or VIRTUAL_WIRE_INTERFACE_SELECT = 0 or OOB_MESSAGE_CHANNEL_ENABLE = 0 or FLASH_CHANNEL_ENABLE = 0			
Enable eSPI Configuration Registers	**0,**1	The parameter to enable all the channel capabilities and configuration registers when their respective channels are disabled in the IP Parameter GUI, General Attributes. This allows the eSPI controller to read and write in the eSPI Channel Configuration	EN_ALL_REGISTERS 0 – Disabled. 1 – Enabled.

Attribute	Selectable Values	Description	Parameter
		registers. However, if a channel is disabled in the IP instantiation parameters, the channel ready bit does not assert.	
EN_CS_FILTER = 1			
Filtered Chip Select Pulse Width	**50**–200	Parameter for the filter width of chip select glitch filter. The value of this parameter is in nanosecond.	ECS_SPIKE_WIDTH = {50,200}
VIRTUAL WIRE CHANNEL ENABLE = 1			
Virtual Wire Interface Select	**GPIO Expander**, Simple Implementation	Parameter for the virtual wire GPIO extender interface enable/disable. If enabled, the virtual wire GPIO extender I/O is in the input/output port. If disabled, 8-bit data I/O per virtual wire is implemented in the IP.	VIRTUAL_WIRE_INTERFACE_SELECT GPIO Extender – 0 Simple Implementation – 1
VIRTUAL WIRE CHANNEL ENABLE = 1 , VIRTUAL_WIRE_INTERFACE_SELECT = 0			
GPIO Input Count	0–512	Parameter for the count of GPIO input. One Virtual Wire Index is assigned per four GPIO. If the count is not divisible by four, the Most Significant bit which is less than four takes a VW Index. Current configuration exceeds the maximum of 128 indexes.	GPIO_IN_COUNT = {0-512}
GPIO Output Count	0–(512–GPIO Input Count)	Parameter for the count of GPIO output.	GPIO_OUT_COUNT = {0-(512-GPIO Input Count)}
GPIO Index Order Start	**GPIO Input First**, GPIO Output First	Parameter to configure the order of the GPIO Input and Output in Virtual Wire Indexing.	GPIO_INDEX_ORDER_START = {0,1}

3.2. Capabilities and Configuration Register Default Values

Table 3.2. Capabilities and Configuration Register Attributes

Attribute	Selectable Values	Description	Parameter
I/O Mode Support	**Single**, Single and Dual, Single and Quad, Single, Dual, and Quad	Parameter for the SPI mode supported by the target.	IO_MODE_SUPPORT = {0,1,2,3}
Open Drain Alert Supported	**0**,1	Parameter that indicates the support of the Alert# pin as an open-drain output by the target.	OPEN_DRAIN_ALERT_SUPPORT = {0,1}
Maximum Frequency Supported	**20**, 25, 30, 50, 66 MHz	Maximum frequency of operation supported by the target.	MAX_FREQ_SUPPORT = {0,1,2,3,4}
Peripheral Channel Maximum Payload Size Supported	**64**, 128, 256 Bytes	The Maximum Payload Size supported by the target for the Peripheral channel.	PERIPHERAL_CHANNEL_MAX_SIZE = {64,128,256}
Maximum Virtual Wire Count Supported	0–63 Bytes	The maximum Virtual Wire count supported by the target.	MAX_VW_COUNT_SUPPORT = {0-63}
OOB Message Channel Maximum Payload Size Supported	**64**, 128, 256 Bytes	The maximum payload size supported by the target for the OOB Message channel.	OOB_MSG_CH_MAX_PAYLOAD_SIZE = {64,128,256}
Flash Access Channel Maximum Payload Size Supported	**64**, 128, 256 Bytes	The maximum payload size supported by the target for the Flash Access channel.	FLASH_ACCESS_CH_MAX_PAYLOAD_SIZE = {64,128,256}

4. Signal Description

Port	Width	Direction	Description
System			
clk_i	1	Input	System clock
resetn_i	1	Input	Active-low system reset
int_o	1	Output	Active-high interrupt output
espi_rst_n_i	1	Input	Active-low eSPI reset
espi_clk_i	1	Input	eSPI clock input driven by eSPI controller
espi_cs_n_i	1	Input	eSPI chip select input driven by eSPI controller
OPEN_DRAIN_ALERT_SUPPORT = 0			
espi_alert_n_o	1	Output	eSPI target alert output
OPEN_DRAIN_ALERT_SUPPORT = 1			
espi_alert_n_io	1	Inout	eSPI target alert output with Pull-up
eSPI Target I/O Interface REMOVE_TRISTATE = True			
espi_data_i	4	Input	eSPI serial data input
espi_data_o	4	Output	eSPI target data output
espi_data_en_o	4	Output	Output enable signal for eSPI target data output, o_espi_data
eSPI Target I/O Interface REMOVE_TRISTATE = False			
espi_data_io	4	Inout	eSPI serial data I/O
AHB-(Lite) Target Bus Interface			
ahbl_tar_sel_i	1	Input	AHB-Lite target select
ahbl_tar_write_i	1	Input	AHB-Lite write • Write = 1 • Read = 0
ahbl_tar_trans_i	2	Input	AHB-Lite target transaction type
ahbl_tar_addr_i	32	Input	AHB-Lite target address
ahbl_tar_size_i	3	Input	AHB-Lite target size indication
ahbl_tar_wdata_i	32	Input	AHB-Lite target write data
ahbl_tar_ready_i	1	Input	AHB-Lite target ready input
ahbl_tar_hburst_i	3	Input	AHB-Lite target burst input
ahbl_tar_rdata_o	32	Output	AHB-Lite target read data
ahbl_tar_ready_o	1	Output	AHB-Lite target ready output
ahbl_tar_resp_o	1	Output	AHB-Lite target response
ahbl_tar_mast_lock_i	1	Input	AHB-Lite controller lock. This is not used in the IP.
ahbl_tar_prot_i	4	Input	AHB-Lite protection control. This is not used in the IP.
APB Target Bus Interface			
apb_tar_addr_i	32	Input	APB address input
apb_tar_sel_i	1	Input	APB transfer type input
apb_tar_enable_i	1	Input	APB transfer type signal using input APB enable control purpose
apb_tar_write_i	1	Input	APB read/write control input
apb_tar_wdata_i	32	Input	APB write data input
apb_tar_rdata_o	32	Output	APB read data output
apb_tar_ready_o	1	Output	APB ready output
apb_tar_err_o	1	Output	APB error indicating output
apb_tar_prot_i	3	Input	APB protection type. This is not used in the IP.

Port	Width	Direction	Description
apb_tar_nse_i	1	Input	APB protection type extension.
apb_tar_strb_i	DATA_WIDTH/8	Input	APB write strobe. This is not used in the IP.
apb_tar_wakeup_i	1	Input	APB wake up. This is not used in the IP.
apb_tar_auser_i	USER_REQ_WIDTH	Input	APB user request attribute. This is not used in the IP.
apb_tar_wuser_i	USER_DATA_WIDTH	Input	APB user write data attribute. This is not used in the IP.
apb_tar_ruser_i	USER_DATA_WIDTH	Input	APB user read data attribute. This is not used in the IP.
apb_tar_buser_i	USER_RESP_WIDTH	Input	APB user response attribute. This is not used in the IP.
Virtual Wire – GPIO Extender			
gpio_expander_i	GPIO Input Count (GUI Parameter)	Input	GPIO expander input.
gpio_expander_o	GPIO Output Count (GUI Parameter)	Output	GPIO expander output.
Virtual Wire – Simple Implementation			
vwire_in_upd_valid	1	Input	Virtual Wire Interface input data is valid.
vwire_in_upd_ready	1	Output	Virtual Wire is ready to receive data from Virtual Wire Interface.
vwire_in_idx	8	Input	Virtual Wire Interface data input index.
vwire_in_value	4	Input	Virtual Wire Interface data input.
vwire_in_valid	4	Input	Virtual Wire Interface data input is valid.
vwire_out_upd_ready	1	Input	Virtual Wire Interface output update is ready.
vwire_out_upd_valid	1	Output	Virtual Wire Interface data output update is valid.
vwire_out_idx	8	Output	Virtual Wire Interface output index.
vwire_out_value	4	Output	Virtual Wire Interface output data.
vwire_out_valid	4	Output	Virtual Wire Interface output data is valid.

4.1. Clock Interface

Table 4.1 Clock Ports

Port	Type	Description
clk_i	Input	System clock. The frequency range is 25 MHz to 100 MHz. It must be faster than espi_clk_i.
espi_clk_i	Input	eSPI clock. The frequency range is 20 MHz to 66 MHz.

5. Register Description

Table 5.1. Summary of eSPI Target IP Core Registers

Start Offset	End Offset	Register Name	eSPI Access	APB/AHB-Lite Access
0x000	0x003	Reserved	RSVD	RSVD
0x004	0x007	Device Identification	RW	RO
0x008	0x00B	General Capabilities and Configurations	RW	RO
0x00C	0x00F	Reserved	RSVD	RSVD
0x010	0x013	Channel 0 Capabilities and Configurations	RW	RO
0x014	0x01F	Reserved	RSVD	RSVD
0x020	0x023	Channel 1 Capabilities and Configurations	RW	RO
0x024	0x02F	Reserved	RSVD	RSVD
0x030	0x033	Channel 2 Capabilities and Configurations	RW	RO
0x034	0x03F	Reserved	RSVD	RSVD
0x040	0x043	Channel 3 Capabilities and Configurations	RW	RO
0x044	0x7FF	Reserved	RO	RO
0x800	0x8FF	IP target core registers	RO	RW

Table 5.2. Access Type Definition

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

5.1. Device Identification

Offset	Bit Field Name	Bit Index	Description	Reset Value	eSPI Access Type
0x004	Reserved	31:8	Reserved	24'h0	RSVD
	Version ID	7:0	Version ID: Indicates compliance to specific eSPI specification revision. Target compliant to this revision of the specification must advertise a value of 01h in this field.	8'h01	RO

5.2. General Capabilities and Configurations

Offset	Bit Field Name	Bit Index	Description	Reset Value	eSPI Access Type
0x008	CRC Checking Enable	31	This bit is set to 1 by the eSPI controller to enable the CRC checking on the eSPI bus. By default, CRC checking is disabled. 0b: CRC checking is disabled. 1b: CRC checking is enabled.	0	RW
	Response Modifier Enable	30	This bit is set to 1 to enable the use of the response modifier by the eSPI target to append either a peripheral completion of channel 0, a virtual wire packet of channel 1, or a flash access completion of channel 3 to the GET STATUS response phase.	0	RW

<table>
<tr><th>Offset</th><th>Bit Field Name</th><th>Bit Index</th><th>Description</th><th>Reset Value</th><th>eSPI Access Type</th></tr>
<tr><td></td><td></td><td></td><td>When this bit is a 0, the eSPI target must only use the Response Modifier of 00, for example, no append. By default, the Response Modifier is disabled.</td><td></td><td></td></tr>
<tr><td></td><td>Reserved</td><td>29</td><td>Reserved</td><td>0</td><td>RO</td></tr>
<tr><td></td><td>Alert Mode</td><td>28</td><td>Alert Mode: This bit serves to configure the Alert mechanism used by the target to initiate a transaction on the eSPI interface.
• 0b: I/O[1] pin is used to signal the Alert event.
• 1b: Alert# pin is used to signal the Alert event.
Note: This bit can only be 0 or 1 in a single controller single target topology. For single controller multiple target topology, this bit must be programmed to 1. Alert Mode is allowed to change from default 0 to 1 during runtime in both single or multiple targets topologies provided. When this happens, only a single target is enabled for generating the Alert# event.</td><td>0</td><td>RW</td></tr>
<tr><td></td><td>IO Mode Select</td><td>27:26</td><td>eSPI controller programs this field to enable the appropriate mode of operation, which takes effect at the deassertion edge of the Chip Select#.
The I/O Mode configured in this field must be supported by both the controller and the target.
The single I/O mode is supported by default.
<table>
<tr><th>Encoding</th><th>Operating Mode</th></tr>
<tr><td>00</td><td>Single I/O</td></tr>
<tr><td>01</td><td>Dual I/O</td></tr>
<tr><td>10</td><td>Quad I/O</td></tr>
<tr><td>11</td><td>Reserved</td></tr>
</table></td><td>00</td><td>RW</td></tr>
<tr><td></td><td>IO Mode Support</td><td>25:24</td><td>This field indicates the I/O modes supported by the target.
<table>
<tr><th>Encoding</th><th>Support I/O Mode</th></tr>
<tr><td>00</td><td>Single</td></tr>
<tr><td>01</td><td>Single and Dual I/O</td></tr>
<tr><td>10</td><td>Single and Quad I/O</td></tr>
<tr><td>11</td><td>Single, Dual, and Quad I/O</td></tr>
</table></td><td>HwInit[1]</td><td>RO</td></tr>
<tr><td></td><td>Open Drain Alert# Select</td><td>23</td><td>This bit is set to 1 by the eSPI controller to configure the Alert# pin as an open-drain output.
By default, Alert# pin operates as a driven output.
This bit must only be programmed to 1 if open-drain Alert# pin is supported by the target.
The bit must be valid when the Alert Mode bit is a 1, indicating Alert# pin is used for signalling the Alert event.
0b: The Alert# pin is a driven output.
1b: The Alert# pin is an open-drain output.</td><td>0</td><td>RW</td></tr>
<tr><td></td><td>Operation Frequency</td><td>22:20</td><td>This field identifies the frequency of operation.
<table>
<tr><th>Bits</th><th>Frequency</th></tr>
<tr><td>000</td><td>20 MHz</td></tr>
<tr><td>001</td><td>25 MHz</td></tr>
<tr><td>010</td><td>33 MHz</td></tr>
</table></td><td>000</td><td>RW</td></tr>
</table>

Offset	Bit Field Name	Bit Index	Description	Reset Value	eSPI Access Type
			011 — 50 MHz 100 — 66 MHz Others are Reserved.		
	Open Drain Alert Supported	19	This bit indicates the support of the Alert# pin as an open-drain output by the target. 0b: The open-drain Alert# pin is not supported. 1b: The open-drain Alert# pin is supported.	HwInit	RO
	Max Frequency Supported	18:16	This field identifies the maximum frequency of operation supported by the target. **Bits** — **Frequency** 000b — 20 MHz 001b — 25 MHz 010b — 33 MHz 011b — 50 MHz 100b — 66 MHz Others are Reserved. The target that indicates support for the maximum frequency of operation through this field also supports all the lower frequencies on the list.	HwInit	RO
	Max wait state allowed	15:12	The eSPI controller sets the maximum wait state allowed to be responded by the target, before the target must respond with an ACCEPT, DEFER, NON-FATAL ERROR, or FATAL ERROR response code. This is a 1-based field in the granularity of byte time. When it is 0, it indicates a value of 16 byte time. A byte time corresponds to eight serial clocks in the Single I/O mode, four serial clocks in the Dual I/O mode or two serial clocks in the Quad I/O mode.	0	RW
	Reserved	11:8	Reserved	0	RO
	Channel Supported	7:0	When set, each of the bits indicates that the corresponding channel is supported by the target. **Bits** — **Channel** 0 — Peripheral 1 — Virtual Wire 2 — OOB Message 3 — Flash Access 4:7 — Reserved for platform specific channels	HwInit	RO

Note:

1. HwInit reset values of a register field means it is tied to the value written in their corresponding register bit field in the eSPI Target IP register.

5.2.1. Channel 0 Capabilities and Configurations

To use less resource, this register is read-only when the Peripheral Channel is disabled in the parameter GUI during IP instantiation. If you intend to instantiate this register in the IP instantiation configuration mentioned earlier, the eSPI Target IP has the parameter, Enable eSPI Channel Configurations Registers.

Offset	Bit Field Name	Bit Index	Description	Reset Value	eSPI Access Type
0x010	Reserved	31:15	Reserved	0	RO
	Peripheral Channel Maximum Read Request Size	14:12	The eSPI controller sets the maximum read request size for the Peripheral channel. The length of the read request must not cross the naturally aligned address boundary of the corresponding Maximum Read Request Size. 000b: Reserved. 001b: 64 bytes address aligned max read request size. 010b: 128 bytes address aligned max read request size. 011b: 256 bytes address aligned max read request size. 100b: 512 bytes address aligned max read request size. 101b: 1024 bytes address aligned max read request size. 110b: 2048 bytes address aligned max read request size. 111b: 4096 bytes address aligned max read request size.	001b	RW
	Reserved	11	Reserved	0	RO
	Peripheral Channel Maximum Payload Size Selected	10:8	The eSPI controller sets the maximum payload size for the Peripheral channel. The value set by the eSPI controller must never be more than the value advertised in the Max Payload Size Supported field. The payload of the transaction must not cross the naturally aligned address boundary of the corresponding Maximum Payload Size. 000b: Reserved. 001b: 64 bytes address aligned max payload size. 010b: 128 bytes address aligned max payload size. 011b: 256 bytes address aligned max payload size. 100b–111b: Reserved.	001b	RW
	Reserved	7	Reserved	0	RO
	Peripheral Channel Maximum Payload Size Supported	6:4	This field advertises the Maximum Payload Size supported by the target. 000b: Reserved. 001b: 64 bytes address aligned max payload size. 010b: 128 bytes address aligned max payload size. 011b: 256 bytes address aligned max payload size. 100b – 111b: Reserved.	HwInit	RO
	Reserved	3	Reserved	0	RO
	Bus Controller Enable	2	When this bit is a 0, it disables the target from generating bus controlling cycles on the Peripheral channel. When this bit is a 1, it allows the target to generate bus controlling cycles on the Peripheral channel. Prior to clearing the Bus Controller Enable bit from 1 to 0, there must be no outstanding non-posted cycle pending completion from the target.	0	RW
	Peripheral Channel Ready	1	When this bit is a 1, it indicates that the target is ready to accept transactions on the Peripheral channel. The eSPI controller should poll this bit after the channel is enabled before running any	0	RO

Offset	Bit Field Name	Bit Index	Description	Reset Value	eSPI Access Type
			transaction on this channel to the target. 0b: Channel is not ready. 1b: Channel is ready.		
	Peripheral Channel Enable	0	Peripheral Channel Enable: The channel is by default enabled after the eSPI Reset#. This bit is cleared to 0 by the eSPI controller to disable the Peripheral channel.	1b	RW

5.2.2. Channel 1 Capabilities and Configurations

To use less resource, this register is read-only when the Peripheral Channel is disabled in the parameter GUI during IP instantiation. If you intend to instantiate this register in the IP instantiation configuration mentioned earlier, the eSPI Target IP has the parameter, Enable eSPI Channel Configurations Registers.

Offset	Bit Field Name	Bit Index	Description	Reset Value	eSPI Access Type
0x020	Reserved	31:22	Reserved	0	RO
	Operating Maximum Virtual Wire Count	21:16	The maximum number of Virtual Wire groups that can be sent in a single Virtual Wire packet. This is a 0-based count. The default value of 0 indicates count of 1. The value configured in this field must never be more than the value advertised in the Maximum Virtual Wire Count Supported field.	0	RW
	Reserved	15:14	Reserved	0	RO
	Maximum Virtual Wire Count Supported	13:8	This field advertises the Maximum Virtual Wire Count supported by the target. If the target supports different count values as initiator and as receiver of the Virtual Wire, this field indicates the lower of the two. The Virtual Wire count specifies the maximum number of Virtual Wire groups communicated in a single Virtual Wire packet. eSPI target must advertise a value of 000111b or more in this field to indicate the support of at least eight Virtual Wire groups communicated in a single Virtual Wire packet. This is a 0-based count.	HwInit	RO
	Reserved	7:2	Reserved	0	RO
	Virtual Wire Channel Ready	1	When this bit is a 1, it indicates that the target is ready to accept transactions on the Virtual Wire channel. The eSPI controller should poll this bit after the channel is enabled before running any transaction on this channel to the target. 0b: The channel is not ready. 1b: The channel is ready.	0	RO
	Virtual Wire Channel Enable	0	This bit is set to 1 by the eSPI controller to enable the Virtual Wire channel. When this bit is 0, no transaction shall occur on the Virtual Wire channel. The channel is by default disabled after the eSPI Reset#.	0	RW

5.2.3. Channel 2 Capabilities and Configurations

To use less resource, this register is read-only when the Peripheral Channel is disabled in the Parameter GUI during IP instantiation. If you intend to instantiate this register in the IP instantiation configuration mentioned earlier, the eSPI Target IP has the parameter, Enable eSPI Channel Configurations Registers.

Offset	Bit Field Name	Bit Index	Description	Reset Value	eSPI Access Type
0x030	Reserved	31:11	Reserved	0	RO
	OOB Message Channel Maximum Payload Size Selected	10:8	The eSPI controller sets the maximum payload size for the OOB Message channel. The value set by the eSPI controller must never be more than the value advertised in the Max Payload Size Supported field. The Maximum Payload Size applies to the actual payload of the protocol embedded in the OOB packet. 000b: Reserved. 001b: 64 bytes max payload size. 010b: 128 bytes max payload size. 011b: 256 bytes max payload size. 100b–111b: Reserved.	001b	RW
	Reserved	7	Reserved	0	RO
	OOB Message Channel Maximum Payload Size Supported	6:4	This field advertises the Maximum Payload Size supported by the target. The Maximum Payload Size applies to the actual payload of the protocol embedded in the OOB packet. 000b: Reserved. 001b: 64 bytes max payload size. 010b: 128 bytes max payload size. 011b: 256 bytes max payload size. 100b–111b: Reserved.	HwInit	RO
	Reserved	3:2	Reserved	0	RO
	OOB Message Channel Ready	1	When this bit is a 1, it indicates that the target is ready to accept transactions on the OOB Message channel. The eSPI controller should poll this bit after the channel is enabled before running any transaction on this channel to the target. 0b: Channel is not ready. 1b: Channel is ready.	0b	RO
	OOB Message Channel Enable	0	This bit is set to 1 by the eSPI controller to enable the OOB Message channel. No OOB transaction is supported while this bit is 0. The channel is by default disabled after the eSPI Reset#.	0b	RW

5.2.4. Channel 3 Capabilities and Configurations

To use less resource, this register is read-only when the Peripheral Channel is disabled in the Parameter GUI during IP instantiation. If you intend to instantiate this register in the IP instantiation configuration mentioned earlier, the eSPI Target IP has the parameter, Enable eSPI Channel Configurations Registers.

Offset	Bit Field Name	Bit Index	Description	Reset Value	eSPI Access Type
0x040	Reserved	31:15	Reserved	0	RO
	Flash Access Channel Maximum Read Request Size	14:12	The eSPI controller sets the maximum read request size for the Flash Access channel. The length of the read request must not exceed the corresponding Maximum Read Request Size with no address alignment requirement. 000b: Reserved. 001b: 64 bytes max read request size. 010b: 128 bytes max read request size. 011b: 256 bytes max read request size. 100b: 512 bytes max read request size. 101b: 1024 bytes max read request size. 110b: 2048 bytes max read request size. 111b: 4096 bytes max read request size.	001b	RW
	Flash Sharing Mode	11	When Flash Access channel is supported, this bit advertises the flash sharing scheme intended by the target. 0b: The controller attaches flash sharing. 1b: The target attaches flash sharing. This bit is a read-only 0 in the base specification, as only the controller-attached flash sharing is defined.	0b	RO
	Flash Access Channel Maximum Payload Size Selected	10:8	The eSPI controller sets the maximum payload size for the Flash Access channel. The value set by the eSPI controller must never be more than the value advertised in the Max Payload Size Supported field. 000b: Reserved. 001b: 64 bytes max payload size. 010b: 128 bytes max payload size. 011b: 256 bytes max payload size. 100b–111b: Reserved.	001b	RW
	Flash Access Channel Maximum Payload Size Supported	7:5	This field advertises the Maximum Payload Size supported by the target. 000b: Reserved. 001b: 64 bytes max payload size. 010b: 128 bytes max payload size. 011b: 256 bytes max payload size. 100b–111b: Reserved.	HwInit	RO
	Flash Block Erase Size	4:2	The eSPI controller sets this field to communicate the block erase size to the target. This field is applicable only to controller-attached flash-sharing scheme. 000b: Reserved 001b: 4 KB 010b: 64 KB 011b: Both 4 KB and 64 KB are supported 100b: 128 KB 101b: 256 KB 110b–111b: Reserved	001b	RW

Offset	Bit Field Name	Bit Index	Description	Reset Value	eSPI Access Type
	Flash Access Channel Ready	1	When this bit is a 1, it indicates that the target is ready to accept transactions on the Flash Access channel. The eSPI controller should poll this bit after the channel is enabled before running any transaction on this channel to the target. 0b: The channel is not ready. 1b: The channel is ready.	0b	RO
	Flash Access Channel Enable	0	This bit is set to 1 by the eSPI controller to enable the Flash Access channel. The channel remains disabled and is not supported until this bit is set to 1. The channel is by default disabled after the eSPI Reset#.	0b	RW

5.2.5. eSPI Target IP Registers

5.2.5.1. eSPI Target ID

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
0x800	Vendor ID	31:24	—	8'h76	RO
		23:16	—	8'h83	RO
		15:8	—	8'h67	RO
	Version ID	7:0	Version ID	8'h01	RO

5.2.5.2. eSPI Target Configuration 0

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
0x804	Reserved	31	Reserved	0	RO
	Flash Access Channel Max Payload Size Supported	30:28	This field advertises the Maximum Payload Size supported by the target. 000b: Reserved. 001b: 64 bytes max payload size. 010b: 128 bytes max payload size. 011b: 256 bytes max payload size. 100b – 111b: Reserved.	GUI IP Param[1]	RW
	Reserved	27	Reserved	0	RO
	OOB Message Channel Max Payload Size Supported	26:24	This field advertises the Maximum Payload Size supported by the target. The Maximum Payload Size applies to the actual payload of the protocol embedded in the OOB packet. 000b: Reserved. 001b: 64 bytes max payload size. 010b: 128 bytes max payload size. 011b: 256 bytes max payload size. 100b – 111b: Reserved.	GUI IP Param[1]	RW
	Reserved	23:22	Reserved	0	RO
	Max Virtual Wire Count Supported	21:16	This field advertises the Maximum Virtual Wire Count supported by the target.	GUI IP Param[1]	RW

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
			If the target supports different count value as initiator and as receiver of the Virtual Wire, this field indicates the lower of the two. The Virtual Wire Count specifies the maximum number of Virtual Wire groups communicated in a single Virtual Wire packet. The eSPI target must advertise a value of 000111b or more in this field to indicate the support of at least eight Virtual Wire groups communicated in a single Virtual Wire packet. This is a 0-based count.		
	Reserved	15	Reserved	0	RO
	Peripheral Channel Max Payload Size Supported	14:12	This field advertises the Maximum Payload Size supported by the target. 000b: Reserved. 001b: 64 bytes address aligned max payload size. 010b: 128 bytes address aligned max payload size. 011b: 256 bytes address aligned max payload size. 100b – 111b: Reserved.	GUI IP Param[1]	RW
	Channel Supported	11:8	Each of the bits when set indicates that the corresponding channel is supported by the target. **Bits** / **Channel** 0 – Peripheral 1 – Virtual Wire 2 – OOB Message 3 – Flash Access	GUI IP Param[1]	RW
	RESERVED	7	Reserved	0	RO
	Maximum Frequency Supported	6:4	This field identifies the maximum frequency of operation supported by the target. **Bits** / **Channel** 000b – 20 MHz 001b – 25 MHz 010b – 33 MHz 011b – 50 MHz 100b – 66 MHz Others are Reserved. The target that indicates support for the maximum frequency of operation through this field also supports all the lower frequencies on the list.	GUI IP Param[1]	RW
	Reserved	3	Reserved	0	RO
	Open Drain Alert Supported	2	This bit indicates the support of the Alert# pin as an open-drain output by the target.	GUI IP Param[1]	RW

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
			0b: Open-drain Alert# pin is not supported. 1b: Open-drain Alert# pin is supported.		
	IO Mode Supported	1:0	This field indicates the I/O modes supported by the target. Encoding / Support I/O Mode: 00 – Single 01 – Single and Dual I/O 10 – Single and Quad I/O 11 – Single, Dual, and Quad I/O	GUI IP Param[1]	RW

Note:

1. GUI IP Param reset values of a register field means the default value is set in the IP Parameter GUI during IP instantiation.

5.2.5.3. eSPI Target Channel Control 0

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
0x808	Reserved	31:30	Reserved	0	RO
	Tx Avail Valid	29:24	This shows that the order in Tx Avail Order in Address 0x810 is valid. If the value is zero, it is ignored. Bit 24 is the first while bit 29 is the last in the sequence.	0	RW
	Reserved	23:20	Reserved	0	RO
	Rx Free valid	19:16	This shows that the order in Rx Avail Order in Address 0x810 is valid. If the value is zero, it is ignored. Bit 16 is the first while bit 19 is the last in the sequence.	0	RW
	Reserved	15:4	—	—	—
	Flash Access Channel Ready	3	When this bit is a 1, it indicates that the target is ready to accept transactions on the Flash Access channel. 0b: Channel is not ready. 1b: Channel is ready.	0	RW
	OOB Message Channel Ready	2	When this bit is a 1, it indicates that the target is ready to accept transactions on the OOB Message channel. 0b: The channel is not ready. 1b: The channel is ready.	0	RW
	Virtual Wire Channel Ready	1	When this bit is a 1, it indicates that the target is ready to accept transactions on the Virtual Wire channel. 0b: The channel is not ready. 1b: The channel is ready.	0	RW
	Peripheral Channel Ready	0	When this bit is a 1, it indicates that the target is ready to accept transactions on the Peripheral channel. 0b: The channel is not ready. 1b: The channel is ready.	0	RW

5.2.5.4. eSPI Target Channel Status 0

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
0x80C	Reserved	31:14	Reserved	0	RO
	Flash NP Avail	13	When the value is 1, it indicates the target has a channel 3 Flash Access non-posted header and data up to maximum payload size available to send. This bit is only applicable when the controller-attached flash sharing is supported and in operation. Otherwise, this bit value is irrelevant.	0	RO
	Flash C Avail	12	When the value is 1, it indicates the target has a channel 3 Flash Access completion header and data up to maximum payload size available to send. This bit is only applicable when target-attached flash sharing is supported and in operation. Otherwise, this bit value is irrelevant.	0	RO
	Reserved	11:10	Reserved	0	RO
	Flash NP Free	9	When the value is 1, it indicates the target is free to accept at least one channel 3 Flash Access non-posted header and data up to maximum payload size. This bit is only applicable when target-attached flash sharing is supported and in operation. Otherwise, this bit value is irrelevant.	0	RO
	Flash C Free	8	When the value is 1, it indicates the target is free to accept at least one channel 3 Flash Access completion header and data up to maximum payload size. This bit must be always a 1. The target must be able to accept the completion for the non-posted request it sends. This bit is only applicable when the controller-attached flash sharing is supported and in operation. Otherwise, the bit is a don't care.	1	RO
	OOB Avail	7	When the value is 1, indicates the target has a channel 2 OOB tunnelled SMBus message with data up to maximum payload size available to send.	0	RO
	VWire Avail	6	When the value is 1, it indicates the target has a channel 1 tunnelled virtual wire available to send.	0	RO
	NP Avail	5	When the value is 1, it indicates the target has a channel 0 peripheral non-posted header available to send.	0	RO
	PC Avail	4	When the value is 1, it indicates the target has a channel 0 peripheral posted or completion header and optional data up to maximum payload size are available to send.	0	RO

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
	OOB Free	3	When the value is 1, it indicates the target is free to accept at least one channel 2 OOB tunnelled SMBus message with data up to the maximum payload size.	0	RO
	VWIRE Free	2	This bit must be always a 1. Tunnelling of channel 1 virtual wire does not have flow control.	1	RO
	NP Free	1	When the value is 1, it indicates the target is free to accept at least one channel 0 peripheral non-posted header and one word of data, if applicable.	0	RO
	PC Free	0	When the value is 1, it indicates the target is free to accept at least one channel 0 peripheral posted or completion header and data up to maximum payload size.	0	RO

5.2.5.5. eSPI Target Tx Rx Order 0

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
0x810	Reserved	31:26	Reserved	0	RO
	Tx Avail Order	25:8	The order of available Tx in GET Channel. Bits 9:8 is the first while 25:24 is the last. Below is the value equivalent for every channel. b000 – PC Avail b001 – NP Avail b010 – OOB Avail b011 – Flash NP Avail b0100 – Flash C Avail	18'h0	RW
	Rx Free Order	7:0	The order of available Rx in PUT Channel. Bits 1:0 is the first while 7:6 is the last. Below is the value equivalent for every channel. 00 – PC Free 01 – NP Free 10 – OOB Free 11 – Flash NP Free	8'h00	RW

5.2.5.6. eSPI Target Interrupt Status 0

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
0x814	Reserved	31:12	Reserved	0	RO
	Rx FIFO Underflow	11	This bit asserts when user interface do a read transaction in RX FIFO while RX FIFO is empty.	0	RW1C
	Invalid Command	10	This bit asserts when the target receives an invalid command.	0	RW1C
	CS Deassertion Error	9	This bit asserts when the target receives an unexpected chip select assertion.	0	RW1C
	CRC Error	8	This bit asserts when the target receives the wrong CRC.	0	RW1C

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
	Tx FIFO Full	7	This bit asserts when the target Tx FIFO is full.	0	RW1C
	VW IN FIFO Full	6	This bit asserts when Virtual Wire Interface Input FIFO is full.	0	RW1C
	Tx FIFO Overflow	5	This bit asserts when the target Tx FIFO reaches overflow.	0	RW1C
	VW IN FIFO Overflow	4	This bit asserts when Virtual Wire Interface Input FIFO reaches overflow.	0	RW1C
	Rx FIFO Overflow	3	This bit asserts when the target Rx Data FIFO reaches overflow.	0	RW1C
	VW OUT FIFO Overflow	2	This bit asserts when the Virtual Wire Interface Output FIFO reaches overflow.	0	RW1C
	Rx FIFO Not Empty	1	This bit asserts when the Rx Data FIFO is not empty.	0	RW1C
	VW OUT FIFO Not Empty	0	This bit asserts when Virtual Wire Interface Output FIFO is not empty.	0	RW1C

5.2.5.7. eSPI Target Interrupt Enable 0

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
0x818	Reserved	31:12	Reserved	0	RO
	Rx FIFO Underflow	11	Enables RX FIFO Underflow error detection.	0	RW
	Invalid Command	10	Enables invalid command error detection.	0	RW
	CS Deassertion Error	9	Enables CS Deassertion error detection.	0	RW
	CRC Error	8	Enables CRC error detection.	0	RW
	Tx FIFO Full	7	Enables Tx Data Full error detection.	0	RW
	VW IN FIFO Full	6	Enables VW IN FIFO Full error detection.	0	RW
	Tx FIFO Overflow	5	Enables Tx Data Overflow error detection.	0	RW
	VW IN FIFO Overflow	4	Enables VW IN FIFO Overflow error detection.	0	RW
	Rx FIFO Overflow	3	Enables Rx Data Overflow error detection.	0	RW
	VW OUT FIFO Overflow	2	Enables VW OUT FIFO Overflow error detection.	0	RW
	Rx FIFO Not Empty	1	Enables Rx Data Not Empty error detection.	0	RW
	VW OUT FIFO Not Empty	0	Enables VW OUT FIFO Not Empty error detection.	0	RW

5.2.5.8. eSPI Target Interrupt Set 0

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
0x81C	Reserved	31:12	Reserved	0	RO
	Rx FIFO Underflow	11	Sets the Rx FIFO Underflow error detection bit.	0	WO
	Invalid Command	10	Sets the invalid command error detection.	0	WO

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
	CS Deassertion Error	9	Sets the CS deassertion error detection.	0	WO
	CRC Error	8	Sets the CRC error detection.	0	WO
	Tx FIFO Full	7	Sets the Tx Data Full error detection.	0	WO
	VW IN FIFO Full	6	Sets the VW IN FIFO Full error.	0	WO
	Tx FIFO Overflow	5	Sets the Tx Data Overflow error detection.	0	WO
	VW IN FIFO Overflow	4	Sets the VW IN FIFO Overflow error.	0	WO
	Rx FIFO Overflow	3	Sets the Rx Data Overflow error.	0	WO
	VW OUT FIFO Overflow	2	Sets the VW OUT FIFO Overflow error.	0	WO
	Rx FIFO Not Empty	1	Sets the Rx Data Not Empty error.	0	WO
	VW OUT FIFO Not Empty	0	Sets the VW OUT FIFO Not Empty error.	0	WO

5.2.5.9. eSPI Target Tx FIFO Queue

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
0x820	Write Register	31:0	Write data to the Tx FIFO	32'h0	WO

5.2.5.10. eSPI Target Rx FIFO Queue

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
0x824	Read Register	31:0	Read data from the Rx FIFO	32'h0	RO

5.2.5.11. eSPI Target Soft Reset

Offset	Bit Field Name	Bit Index	Description	Reset Value	APB/AHB-Lite Access Type
0x828	Reserved	31:7	Reserved	24'h0	RO
	CSR Reset	6	Configuration and Status Register block active-high soft reset	0	RW
	Auto Clear Reset	5:3	If the corresponding bit of the register below is set to 1, it clears their respective reset signal after one clock cycle. Bit 5 – Rx FIFO reset Bit 4 – Tx FIFO reset Bit 3 – IP Core reset	3'h0	RW
	RX FIFO Reset	2	RX FIFO active-high soft reset	0	RW
	TX FIFO Reset	1	TX FIFO active-high soft reset	0	RW
	IP Core Reset	0	eSPI Target IP active-high soft reset. Resets every register in the IP except the CSR block.	0	RW

6. Example Design

The eSPI Target IP example design allows you to compile, simulate, and test the eSPI Target IP on MachXO5-NX Development Board Rev B. Refer to MachXO5-NX Programming and Configuration User Guide (FPGA-TN-02271) for more details.

6.1. Example Design Supported Configuration

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

eSPI Target IP GUI Parameter	eSPI IP Configuration Supported in Example Demo Design
Remove Tristate	0
Enable Chip Select Glitch Filter	0
Enable eSPI Channel Configurations Registers	0
User Interface Select	None
Peripheral Channel Enable	0
Virtual Wire Channel Enable	1
OOB Message Channel Enable	0
Flash Access Channel Enable	0
eSPI Target Configuration Programmable	1
Virtual Wire Interface Select	GPIO Expander
GPIO Input Count	64
GPIO Output Count	64
GPIO Index Order Start	0
I/O Mode Support	Single, Dual, and Quad
Open Drain Alert Supported	1
Maximum Frequency Supported	20
Peripheral Channel Maximum Payload Size Supported	64
Maximum Virtual Wire Count Supported	7
OOB Message Channel Maximum Payload Size Supported	64
Flash Access Channel Maximum Payload Size Supported	64

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
- AXI Interconnect
- AXI to APB
- Oscillator
- PLL
- Glue Logic

The eSPI Controller is instantiated and connected in the project shown in Figure 6.1.

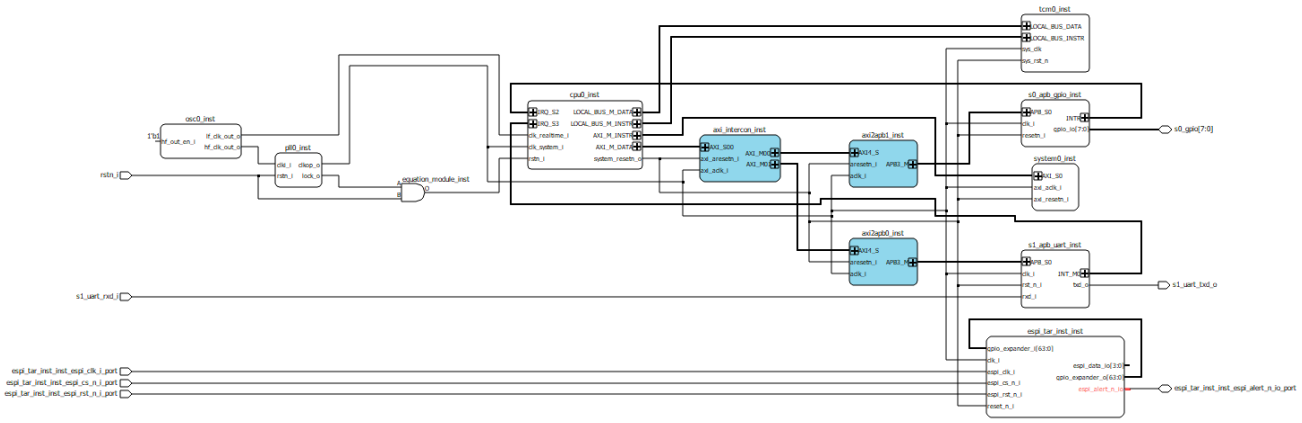

Figure 6.1. eSPI Target IP in Propel SoC Project

6.3. Example Design Components

The eSPI IP Target example design includes the following blocks:

- RISC-V CPU – Passes the C Code Test Routine from system memory to system bus. It handles interrupts.
- Memory – Contains commands to be done for testing.
- System Bus – APB systems bus for transfers between the memory and IP
- eSPI Target IP

6.4. Simulating the Example Design

Refer to the Lattice Propel 2024.1 SDK User Guide (FPGA-UG-02211) for more details on the Lattice Propel design environment.

1. Launch Lattice Propel SDK and set your workspace directory.
2. In Lattice Propel SDK, create a new Lattice SoC Design Project by clicking **File > New > Lattice SoC Design Project**.
3. The Create SoC Project window opens.
   - In the **Device Select** section, specify correct details of the device or board that you are using. In Figure 6.2, the device is set to LFMXO5-25-8BBG400C since MachXO5-NX Evaluation Board is used in the hardware testing.
   - In the **Template Design** section, choose **RISC-V MC SoC Project**. Click **Finish**.

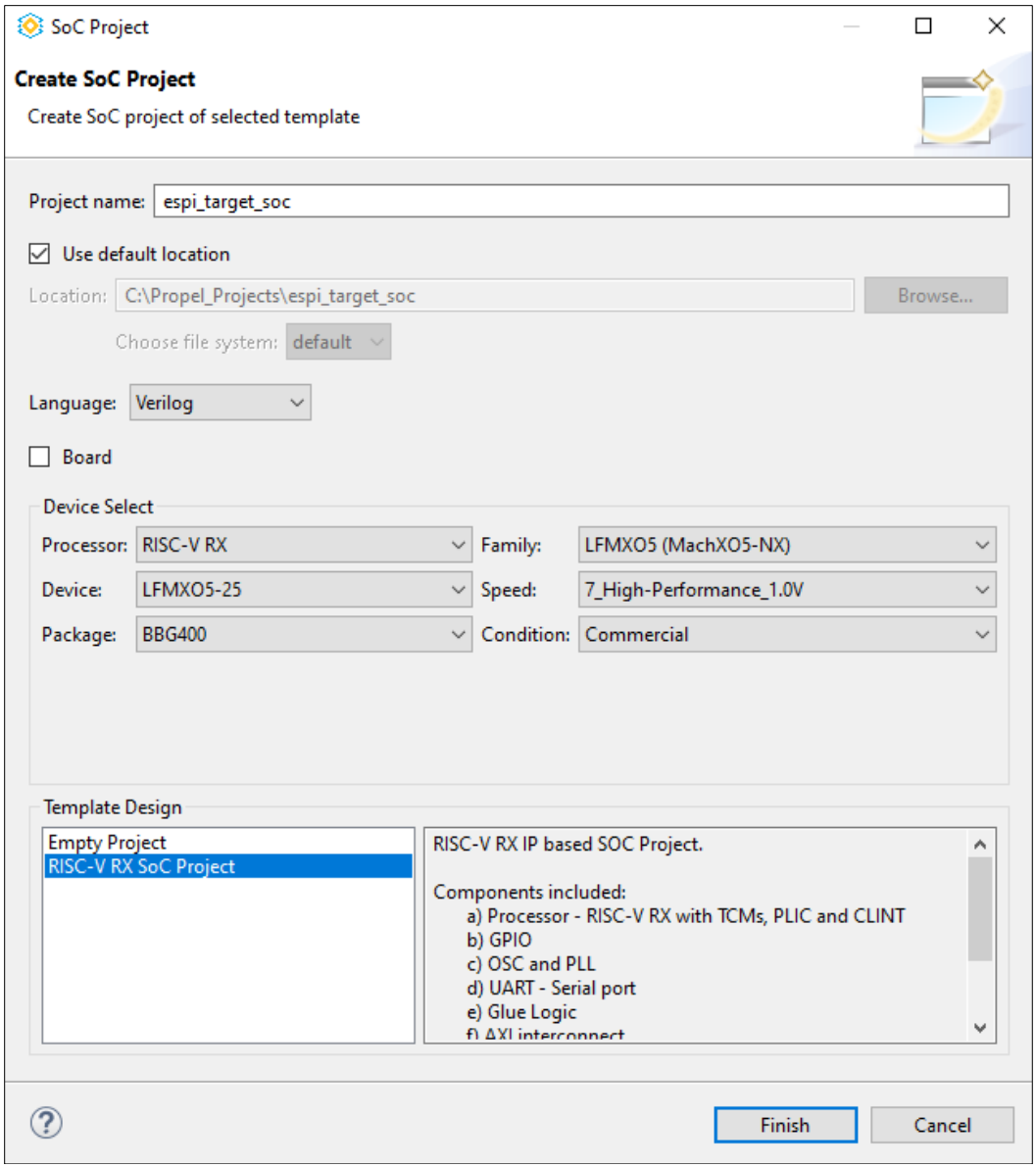

Figure 6.2. Create SoC Project

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

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

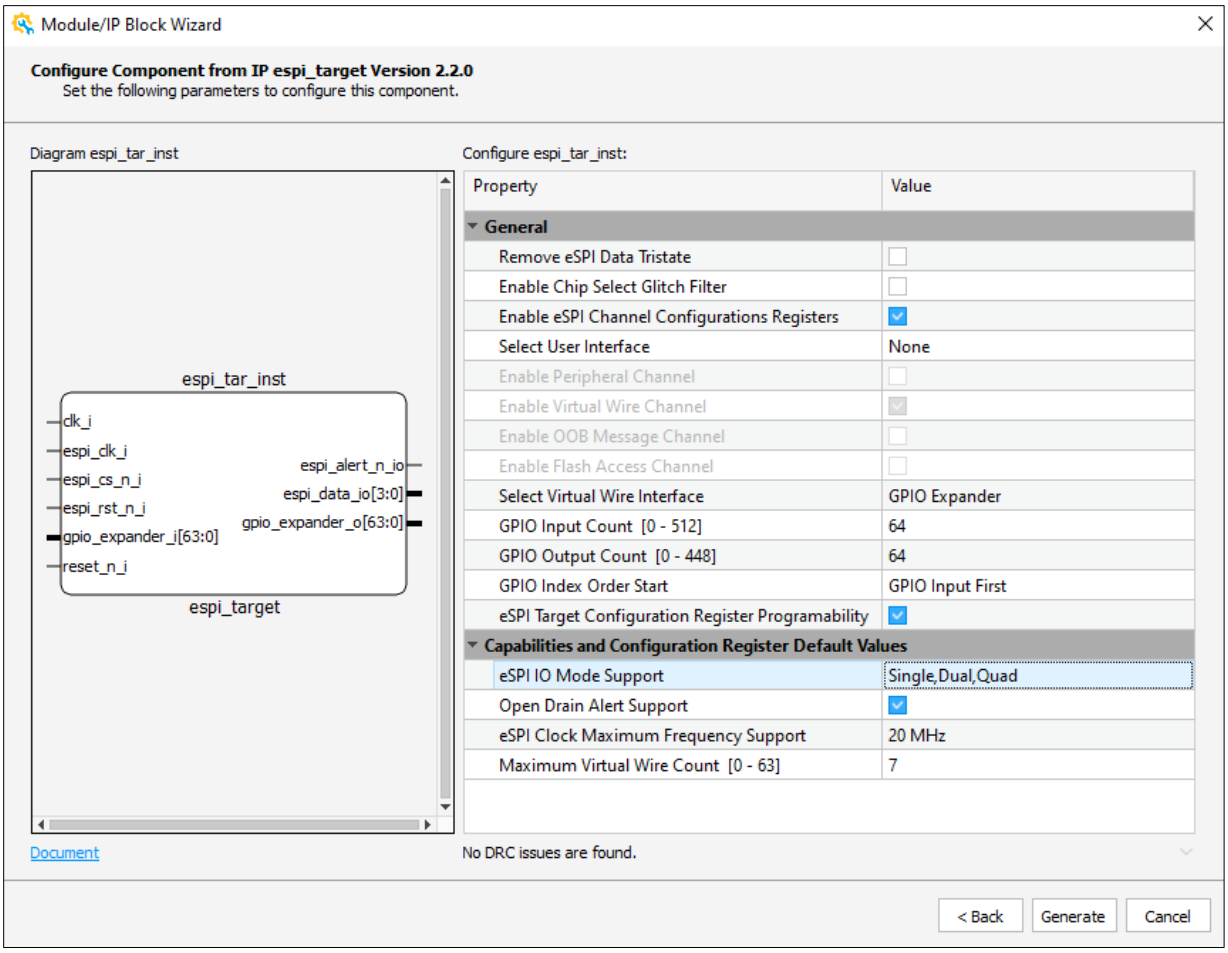

Figure 6.3. Instantiating eSPI Target IP Module

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

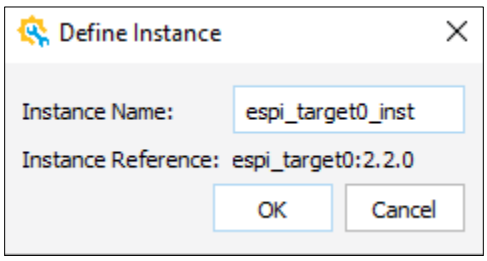

Figure 6.4. Defining Instances

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

13. Select your C/C++ project, then, select **Project** > **Build**.
14. Check the build result from the **Console** view.
15. This environment is now ready for running your tests on the device. Refer to the MachXO5-NX Development Board User Guide (FPGA-EB-02052) for a step-by-step guide.

7. Designing with the IP

This section 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.

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 eSPI Target IP in the Lattice Radiant software.

To generate the eSPI Target IP:

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

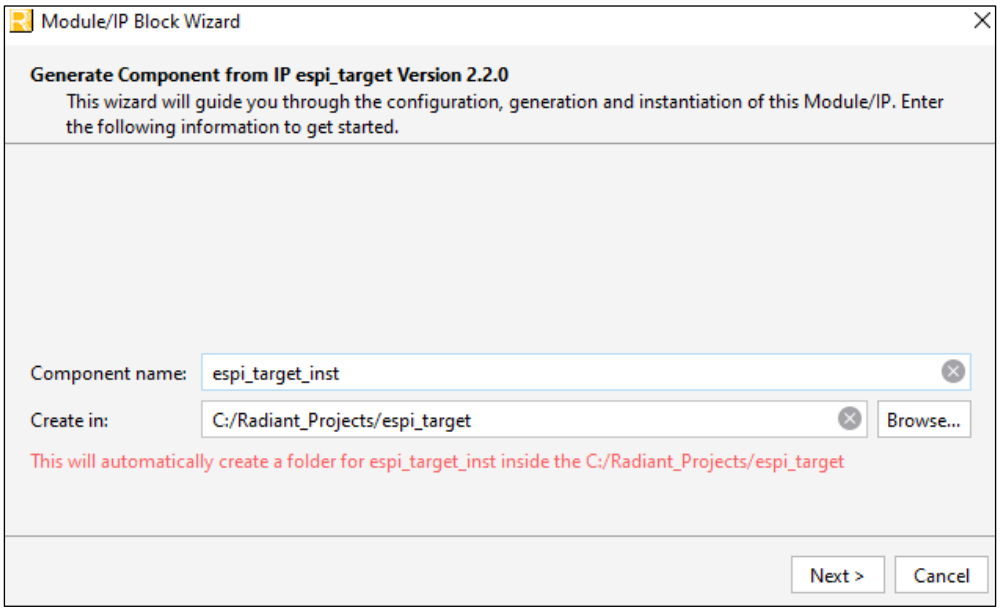

Figure 7.1. Module/IP Block Wizard

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

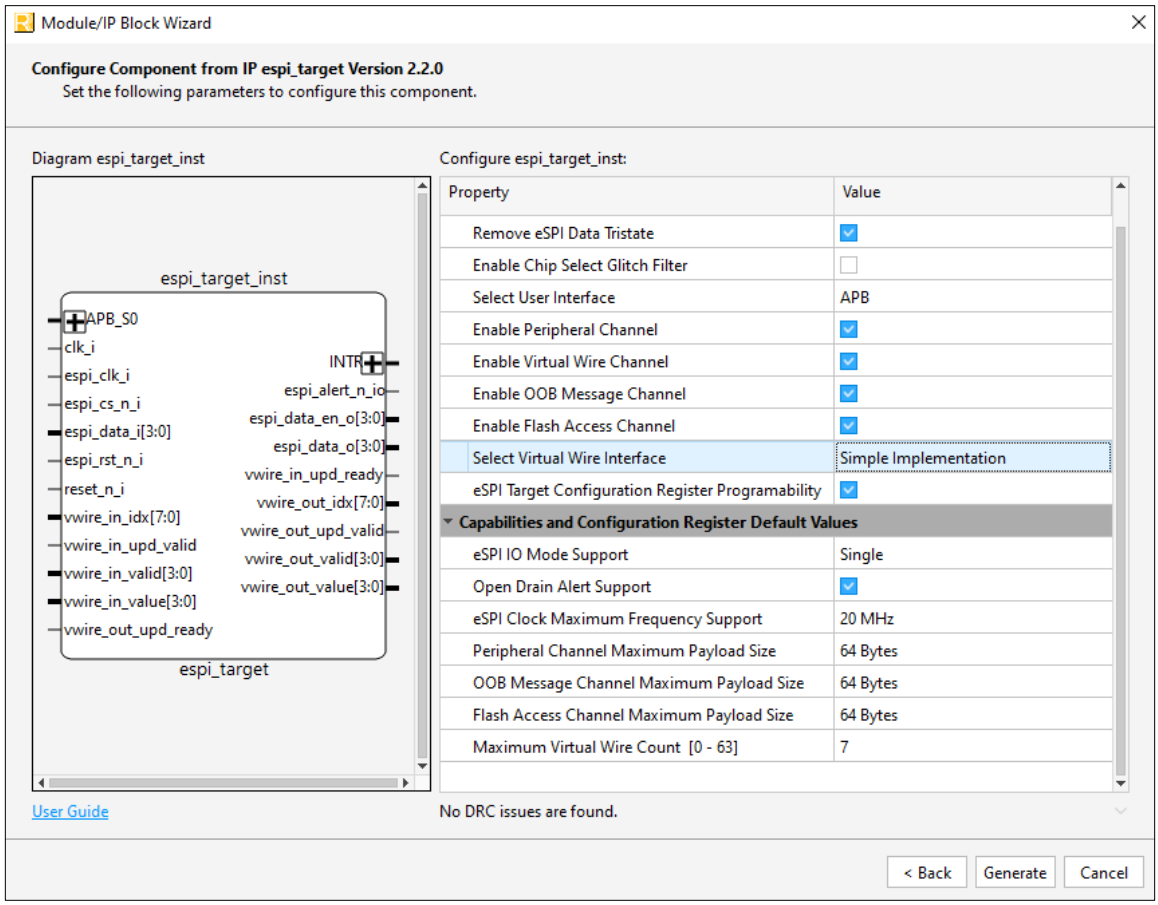

Figure 7.2. IP Configuration

4. Click **Generate**. The **Check Generated Result** dialog box opens, showing design block messages and results (Figure 7.3).

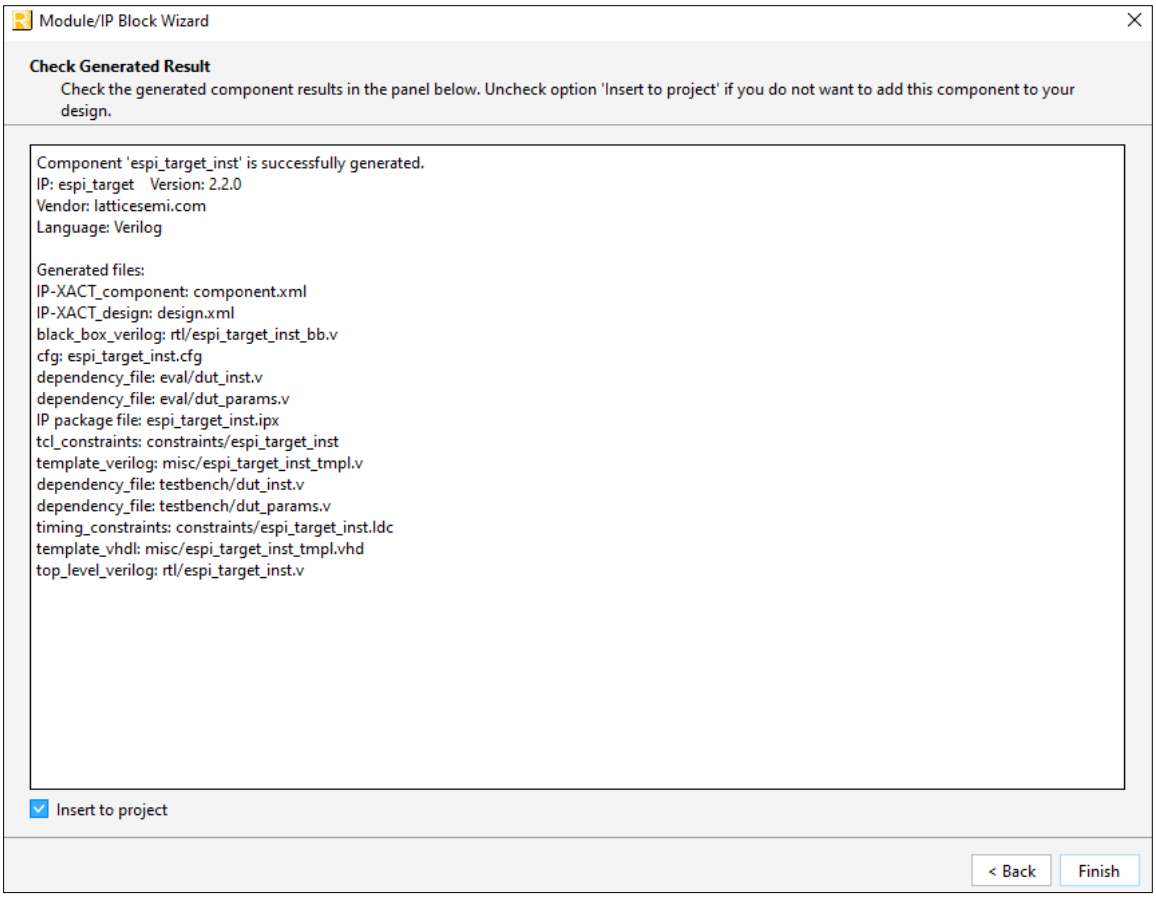

Figure 7.3. Check Generated Result

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 eSPI Target 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 with 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.

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 Constraints

The timing constraints are based on the clock frequency used. The timing constraints for the IP are defined in relevant constraint files. The example below shows the IP timing constraints generated for the eSPI Target IP.

```
if {$radiant(stage) == "premap"} {
set CLKI_PERIOD  10.0
set ESPI_PERIOD  16.303

set CLKI_PERIOD_BY_2 5.000
set ESPI_PERIOD_BY_2 8.152

create_clock -name {clk_i} -period $CLKI_PERIOD -waveform "0 $CLKI_PERIOD_BY_2" [get_ports clk_i]
create_clock -name {espi_clk_i} -period $ESPI_PERIOD -waveform "0 $ESPI_PERIOD_BY_2" [get_ports espi_clk_i]

set_false_path -from clk_i -to espi_clk_i
set_false_path -from espi_clk_i -to clk_i
}
```

Figure 7.4. Timing Constraint File (.pdc) for the eSPI Target IP

For timing closure, it should be automatically added in the project hierarchy. In case it is not added, put the following timing constraints in a .sdc file. Define proper clock constraints as follows:

- set CLKI_PERIOD 10.0
- set ESPI_PERIOD 16.303
- set CLKI_PERIOD_BY_2 5.000
- set ESPI_PERIOD_BY_2 8.152
- create_clock -name {clk_i} -period $CLKI_PERIOD -waveform "0 $CLKI_PERIOD_BY_2" [get_ports clk_i]
- create_clock -name {espi_clk_i} -period $ESPI_PERIOD -waveform "0 $ESPI_PERIOD_BY_2" [get_ports espi_clk_i]
- set_false_path -from clk_i -to espi_clk_i
- set_false_path -from espi_clk_i -to clk_i

7.4. Specifying the Strategy

The Radiant software provides two predefined strategies: Area and Timing. It also enables you to create customized strategies. For details on how to create a new strategy, refer to the Strategies section of the Lattice Radiant Software User Guide.

7.5. Running Functional Simulation

You can run functional simulation after the IP is generated.

To run functional simulation:

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

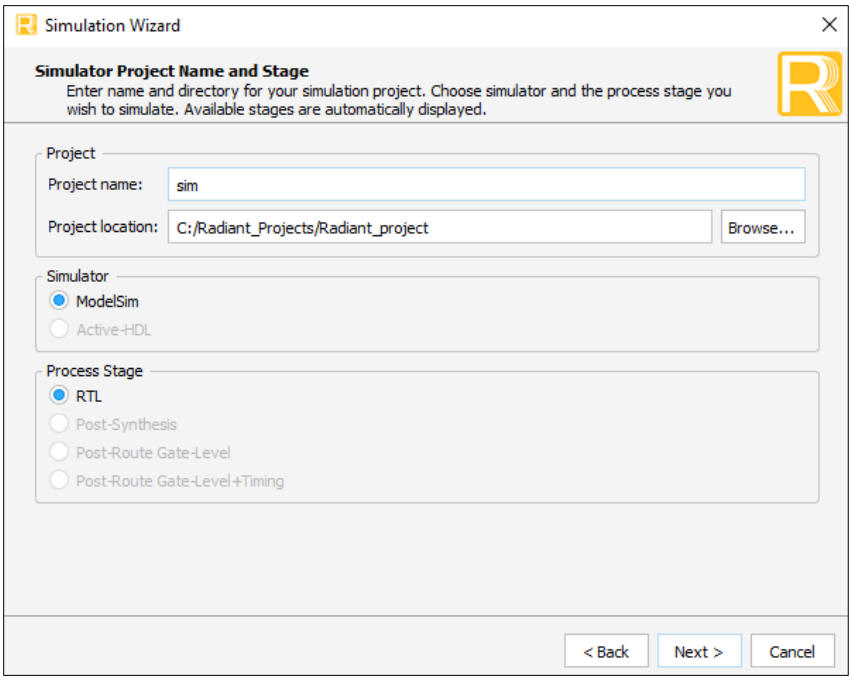

Figure 7.5. Simulation Wizard

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

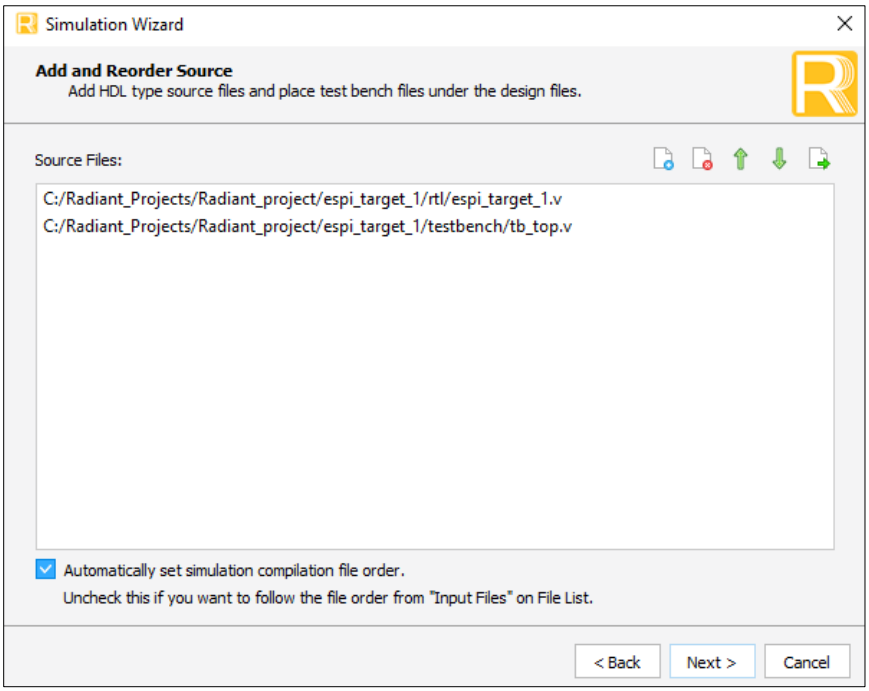

Figure 7.6. Add and Reorder Source

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

The waveform in Figure 7.7 shows an example simulation result.

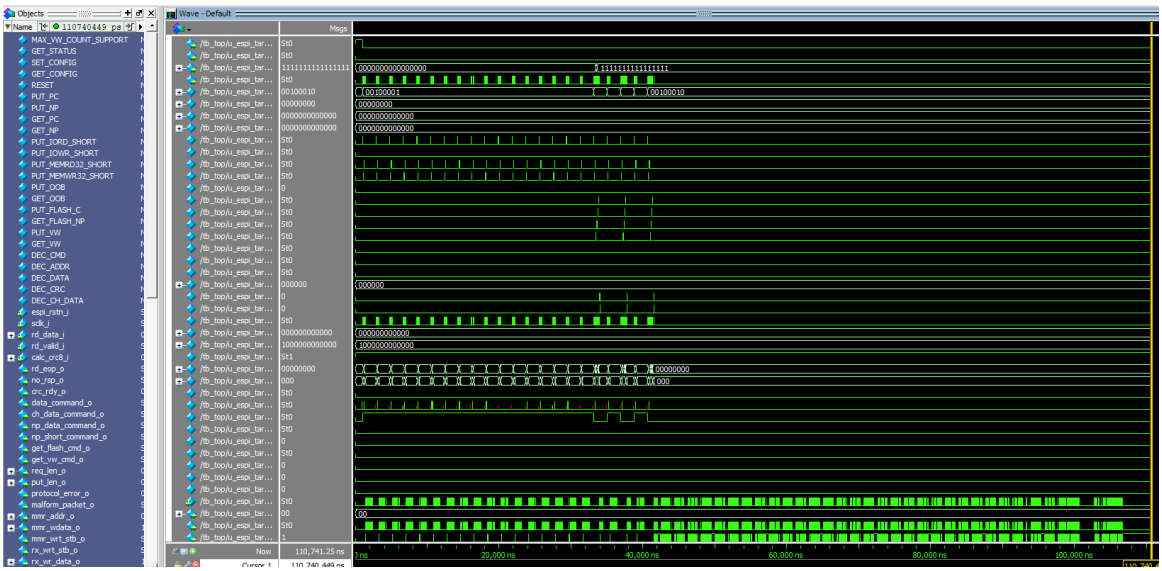

Figure 7.7. Simulation Waveform

7.5.1. Simulation Results

Observe all the eSPI interface related signals. It should be displaying the transactions done by the customer testbench.

Check the transcript page of the simulations to read all the transactions. Errors and warnings are also reported in this page.

Appendix A. Resource Utilization

Table A.1, Table A.2, Table A.3, and Table A.4 show the resource utilization of the eSPI Target IP on the LFMXO5-100T-9BBG400C device.

Table A.1. Virtual Wire Only

IP Configuration	Slices	LUTs	Registers	EBR	eSPI Target
No User Interface, Peripheral Channel Disabled, Virtual Wire Channel Enabled, OOB Message Channel Disabled, Flash Access Channel Disabled, Virtual Wire Interface – Simple Implementation	464/23040	758/23040	485/23637	8/80	1

Table A.2. Channel 0,2,3 Enabled with APB Interface

IP Configuration	Slices	LUTs	Registers	eSPI Target
APB Register Interface, Peripheral Channel Enabled, Virtual Wire Channel Disabled, OOB Message Channel Enabled, Flash Access Channel Enabled,	966/23040	1630/23040	1006/23637	1

Table A.3. All Channel Enabled with APB Interface – GPIO Expander

IP Configuration	Slices	LUTs	Registers	eSPI Target
APB Register Interface, Peripheral Channel Enabled, Virtual Wire Channel Enabled, OOB Message Channel Enabled, Flash Access Channel Enabled, Virtual Wire Interface – GPIO Expander	1330/23040	2142/23040	1434/23637	1

Table A.4. All Channel Enabled with APB Interface – Simple Implementation

IP Configuration	Slices	LUTs	Registers	EBR	eSPI Target
APB Register Interface, Peripheral Channel Enabled, Virtual Wire Channel Enabled, OOB Message Channel Enabled, Flash Access Channel Enabled, Virtual Wire Interface – Simple Implementation	1168/23040	1907/23040	1224/23637	8/80	1

References

- eSPI Target IP Release Notes (FPGA-RN-02002)
- eSPI Target IP web page
- Lattice Radiant software
- Lattice Propel Design Environment web page
- MachXO5-NX Family Devices web page
- Mach-NX Family Devices web page
- MachXO3D Family Devices web page
- MachXO3 Family Devices web page
- MachXO2 Family Devices web page
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, please refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.1, IP v2.2.0, December 2024

Section	Change Summary

Introduction	• In Table 1.1. Summary of eSPI Target IP:
	  • added the IP Changes row;
	  • removed the original IP Version row;
	  • added *MachXO2* to Supported FPGA Family;
	  • updated IP Version from *2.2.0* to *2.2.2*;
	  • added *LCMXO2*, *LAMXO3D*, *LAMXO3LF*, and *LCMXO3L* to Targeted Devices;
	  • added *Virtual Wire Interface* to the Supported User Interface.
	• In the Features section:
	  • added *eSPI Base Specification Revision 1.0 features*;
	  • changed *supports all eSPI Commands* to *supports all eSPI commands except Short Read commands*;
	  • changed *CRC check* to *Cyclic Redundancy Check*;
	  • changed *non-fatal error detection in eSPI command* to *optional non-fatal error detection of the eSPI target*;
	  • added *refer to the Channel Capabilities and Configurations section*;
	  • added *peripheral Short Read commands. This IP version responds with NON-FATAL ERROR response to peripheral channel Short Read commands*.
	• Added the IP Support Summary and Hardware Support sections.
	• Removed the original IP Validation Summary section.

Functional Description	• Updated Figure 2.1. Lattice ESPI Target X4 Core Block Diagram.
	• In the Reset Overview section, changed *the reset assertion can be asynchronous but reset negation should be synchronous. When asserted, output ports and registers are forced to their reset values* to *The reset assertion can be asynchronous but reset deassertion is synchronized inside the eSPI Target IP. When deasserted, output ports and registers are forced to their reset values.*
	• In Table 2.1. User Interfaces and Supported Protocols, added *Virtual Wire Interface* and related contents.
	• Added the Blocks section.
	• In the eSPI Commands section:
	  • changed a*ll the eSPI Commands in the Intel eSPI Specification are supported in the Lattice eSPI Target IP* to *all the eSPI commands in the Intel eSPI specification are supported in the Lattice eSPI Target IP, except the Short Read commands*;
	  • added *For Short Read commands, the eSPI Target IP responds with the NON-FATAL ERROR response*.
	• In the Virtual Wire IN section, changed *SET_CONFIG* to *SET STATUS* in step 5.
	• In the Virtual Wire OUT section, changed *SET_CONFIG* to *SET STATUS* in step 2.
	• In the GPIO Expander Input section:
	  • updated Figure 2.8. GPIO Expander Input;
	  • added i*n the example above, the value of the vw_index 8'h80 is used to show that the virtual wire index is changed.*
	• In the GPIO Expander Output section:
	  • updated Figure 2.9. GPIO Expander Output;
	  • added *in the example above, the virtual wire index 8'h80 is the index assigned to gpio_expander_o[3:0]*;
	  • updated Figure 2.10. PUT VWIRE Interrupt Assertion Timing.
	• In the Program Flow for Writing Data to FIFO section:
	  • added w*riting data to FIFO is used when the microcontroller is sending data to the eSPI controller and the controller receives the data using GET <CHANNEL> eSPI*

Section	Change Summary
	commands. While writing to the FIFO, the payload header must be sent first, then, the payload data is sent. After writing data to the FIFO, the microcontroller needs to write to control the eSPI target status to assert the AVAIL register bit for their respective channel depending on the transaction; • added *Since the data width of APB/AHB-Lite is limited to 32 bits. The data is sent word by word. For example, the eSPI target is sending a 64-bit request peripheral memory write to the controller. The first word includes the cycle type, tag, length, and address bits 31 to 24. The second word contains the remainder of the address and the first byte of the payload data. The data write to the FIFO is shown in Figure 2.11*; • added Figure 2.11. FIFO Write Format. • In the Program Flow for Reading Data from FIFO section: • added *reading data from the FIFO is performed when the interrupt status indicates that the FIFO is not empty. The data to be read from the FIFO includes the command opcode, payload header, and optional payload data*; • added *since the data width of APB/AHB-Lite is limited to 32 bits. The data is sent word by word. For example, the eSPI controller is sending a 32-bit request to write to the peripheral memory of the target. The first word includes the command opcode, cycle type, tag, and length. The second word contains the address. After that, the payload data, which is optional because there are transactions without message, is sent. The data read to the FIFO is shown in Figure 2.12.*; • added Figure 2.12. FIFO Read Format. • Divided the original Program Flow for Using GET Channel Commands section into four sub-sections and provided corresponding information: GET Channel Commands – Peripheral Channel, GET Channel Commands – Virtual Wire Channel, GET Channel Commands – OOB Message Channel, and Flash Access Channel. • Divided the original Program Flow for Using PUT Channel Commands section into four sub-sections and provided corresponding information: PUT Channel Commands – Peripheral Channel, PUT Channel Commands – Virtual Wire Channel, PUT Channel Commands – OOB Message Channel, and PUT Channel Commands – Flash Access Channel. • Added the Error Handling section.
IP Parameter Description	• In Table 3.1. General Attributes, added the *Enable eSPI Channel Configurations Registers* and *Enable eSPI Configuration Registers* attributes.
Signal Description	• In the Signal Description table: • added the following signals under System, *espi_rst_n_i*, *espi_clk_i*, and *espi_cs_n_i*; • added *espi_alert_n_o* under OPEN_DRAIN_ALERT_SUPPORT = 0; • added *espi_alert_n_io* under OPEN_DRAIN_ALERT_SUPPORT = 1; • removed the following signals from eSPI Target I/O Interface REMOVE_TRISTATE = True: *espi_rst_n_i*, *espi_clk_i*, *espi_cs_n_i*, *espi_alert_n_o*, and *espi_alert_en_o*; • removed the following signals from eSPI Target I/O Interface REMOVE_TRISTATE = False: *espi_rst_n_i*, *espi_clk_i*, *espi_cs_n_i*, *espi_alert_n_o*, *espi_alert_n_io*. • Changed the Width of ahbl_tar_resp_o from *2* to *1*. • In Table 4.1 Clock Ports, updated the description of clk_i from *System Clock. Frequency Range is 50 MHz to 100 MHz* to *System clock. The frequency range is 25 MHz to 100 MHz. It must be faster than espi_clk_i*.

Section	Change Summary

Register Description	• In the Device Identification section, changed the reserved Bit Index of 0x004 from *31:28* to *31:8*.
	• In the Channel 0 Capabilities and Configurations, Channel 1 Capabilities and Configurations, Channel 2 Capabilities and Configurations, Channel 3 Capabilities and Configurations sections, added the following description:
	to use less resource, this register is read-only when the Peripheral Channel is disabled in the Parameter GUI during IP instantiation. If you intend to instantiate this register in the IP instantiation configuration mentioned earlier, the eSPI Target IP has the parameter, Enable eSPI Channel Configurations Registers.
	• In the Channel 0 Capabilities and Configurations section, removed the following description for Peripheral Channel Enable:
	besides, clearing this bit from 1 to 0 triggers a reset to the Peripheral channel. The channel remains disabled until this bit is set to 1 again. Prior to disabling the Peripheral channel, the Bus Controller Enable bit should be cleared to 0 to disable the bus controlling cycles.
	• In the Channel 1 Capabilities and Configurations section, removed the following description for Virtual Wire Channel Enable:
	clearing this bit from 1 to 0 does not reset the Virtual Wire channel whereby the state of all the Virtual Wires must continue to be maintained internally.
	• In the Channel 2 Capabilities and Configurations section, removed the following description for OOB Message Channel Enable:
	clearing this bit from 1 to 0 triggers a reset to the OOB Message channel such as during error handling. The channel remains disabled until this bit is set to 1 again.
	• In the Channel 2 Capabilities and Configurations section, added the following description for OOB Message Channel Enable:
	no OOB transaction is supported while this bit is 0.
	• In the Channel 3 Capabilities and Configurations section, removed the following description for the Flash Access Channel Enable:
	clearing this bit from 1 to 0 triggers a reset to the Flash Access channel such as during error handling.
	• In the Channel 3 Capabilities and Configurations section, updated the following description for the Flash Access Channel Enable from *the channel remains disabled until this bit is set to 1 again* to
	the channel remains disabled and not supported until this bit is set to 1.
	• In the eSPI Target IP Registers section:
	  • removed the following description for Flash Access Channel Ready, OOB Message Channel Ready, Virtual Wire Channel Ready, and Peripheral Channel Ready in eSPI Target Channel Control 0:
	  eSPI controller should poll this bit after the channel is enabled before running any transaction on this channel to the target;
	  • removed the following description for Write Register in eSPI Target Tx FIFO Queue, *write sequence in eSPI GET commands is least significant byte to most significant byte*;
	  • removed the following description for Read Register in eSPI Target Rx FIFO Queue, *read sequence is least significant byte to most significant byte*.

Example Design	• In Table 6.1. eSPI Target IP Configuration Supported by the Example Design:
	  • updated the value of Remove Tristate from *1* to *0*;
	  • added the *Enable eSPI Channel Configurations Registers* parameter;
	  • updated the value of User Interface Select from *APB* to *None*;
	  • updated the value of Peripheral Channel Enable from *1* to *0*;
	  • updated the value of OOB Message Channel Enable from *1* to *0*;
	  • updated the value of Flash Access Channel Enable from *1* to *0*;
	  • updated the value of GPIO Input Count from *8* to *64*;
	  • updated the value of GPIO Output Count from *8* to *64*;
	  • updated the value of I/O Mode Support from *Single* to *Single*, *Dual*, and *Quad*;

Section	Change Summary

	• updated the value of Maximum Virtual Wire Count Supported from *15* to *7*.
	• In the Overview of the Example Design and Features section:
	  • changed the components of the generated project from *Processor – RISC-V MC with PIC Timer, AHB-Lite Target, APB Target, Oscillator, PLL, Glue Logic* to *Processor – RISC-V MC, AXI Interconnect, AXI to APB, Oscillator, PLL, Glue Logic*
	  • Updated Figure 6.1. eSPI Target IP in Propel SoC Project and related description.
	• Removed the original Figure 6.2 eSPI Target Example Design Block Diagram.
	• Updated Figure 6.2. Create SoC Project and Figure 6.4. Defining Instances.
	• Added Figure 6.3. Instantiating eSPI Target IP Module.

Designing with the IP	• Updated Figure 7.1. Module/IP Block Wizard, Figure 7.2. IP Configuration, Figure 7.3. Check Generated Result, Figure 7.4. Timing Constraint File (.pdc) for the eSPI Target IP, Figure 7.5. Simulation Wizard, Figure 7.6. Add and Reorder Source, and Figure 7.7. Simulation Waveform. • Updated the clock timing constraints in the Timing Constraints section.
Debugging	Removed this section.
Resource Utilization	• Changed the device for testing from *LFMXO5-100T-7BBG400C* to *LFMXO5-100T-9BBG400C*. • Added the resource utilization data for the following conditions: Table A.1. Virtual Wire Only; Table A.2. Channel 0,2,3 Enabled with APB Interface; Table A.3. All Channel Enabled with APB Interface – GPIO Expander; Table A.4. All Channel Enabled with APB Interface – Simple Implementation.
References	Added the link to MachXO2 Family Devices web page.

Revision 1.0, June 2024

Section	Change Summary
All	Production release.

www.latticesemi.com