QSPI Flash Controller IP

IP Version: v1.5.0

User Guide

FPGA-IPUG-02248-1.3

June 2025

Disclaimers

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

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AHB	Advanced High-performance Bus
AHB-L, AHB-Lite	Advanced High-performance Bus Lite
AMBA	Advanced Microcontroller Bus Architecture
AXI4	Advanced eXtensible Interface 4
AXI4-L, AXI4-Lite	Advanced eXtensible Interface 4 Lite
CLINT	Core Local Interrupter
CPHA	SPI Clock Phase
CPOL	SPI Clock Polarity
CPU	Central Processing Unit
CSR	Configuration Status Register
DRC	Design Rule Checking
DSPI	Dual Serial Peripheral Interface
FIFO	First In, First Out
FPGA	Field-Programmable Gate Array
HDL	Hardware Description Language
INCR	Incrementing
LSB	Least Significant Bit
MSB	Most Significant Bit
PFR	Platform Firmware Resiliency
PLIC	Platform Level Interrupt Controller
QSPI	Quad Serial Peripheral Interface
SPI	Serial Peripheral Interface
TCM	Tightly-Coupled Memory
XiP	eXecute-in-Place

1. Introduction

1.1. Overview of the IP

The Lattice QSPI Flash Controller IP core supports the SPI, DSPI, and QSPI protocols to perform operations on the target flash device.

A Quad Serial Peripheral Interface (QSPI) uses four tri-state data lines and is commonly used to program, erase, and read SPI flash memories. QSPI enhances the throughput of a standard SPI by four times since four bits are transferred with every clock cycle.

A Dual Serial Peripheral Interface (DSPI) uses two tri-state data lines and is used to program, erase, and read SPI flash memories. DSPI transfers two bits with every clock cycle.

A Standard Serial Peripheral Interface (SPI) uses separate data lines for input and output to perform flash operations.

1.2. Quick Facts

Table 1.1. Summary of the QSPI Flash Controller IP

IP Requirements	Supported Devices	Lattice CrossLink™-NX, Certus™-NX (LFD2NX-17, LFD2NX-40), CertusPro™-NX, MachXO5™-NX (LFMXO5-25, LFMXO5-55T, LFMXO5-100T), Avant™ (LAV-AT-E70, LAV-AT-G70, LAV-AT-X70), Certus-N2 (LN2-CT-20)
	IP Changes	Refer to the QSPI Flash Controller IP Release Notes (FPGA-RN-02016).
Resource Utilization	Supported User Interface	Advanced High-performance Bus Lite (AHB-Lite), Advanced eXtensible Interface 4 Lite (AXI4-Lite), Advanced eXtensible Interface 4 (AXI4)
	Resources	Refer to Appendix A. Resource Utilization.
Design Tool Support	Lattice Implementation[1]	IP Core v1.4.0 – Lattice Radiant™ Software 2024.2 IP Core v1.5.0 – Lattice Radiant Software 2025.1
	Synthesis	Synopsys® Synplify Pro®, Lattice Synthesis Engine
	Simulation	Refer to the Lattice Radiant Software User Guide for the list of supported simulators.

Note:

1. Lattice Implementation indicates the IP version release coinciding with the software version release. Check the software for IP version compatibility with earlier or later software versions.

1.3. IP Support Summary

Table 1.2. QSPI Flash Controller IP Support Readiness

Device Family	IP Configuration and Settings	On-board Flash	Clock Frequency	Flash Settings	Flash Commands	Radiant Timing Model	Hardware Validated
CertusPro-NX	AHB-L and AXI4L interface, Disabled Transmit and Receive FIFO, Supported Commands	Micron 128 Mbit SPI Flash (MT25QU128ABA)	System Clock Frequency: 125 MHz SPI Clock Frequency (max): 31.25 MHz (System Clock Frequency divide by 4)	Quad SPI, Clock mode 3, 3-byte address mode	Read Configuration Register, Read ID, Multiple I/O Read ID, Fast Read, Dual Input/Output Fast Read, Quad Input/Output Fast Read, Block Erase Type 1, Block Erase Type 3, Write Configuration Register, Page Program, Dual Input Fast Program, Quad Input Fast Program	Final	Yes
	AHB-L and AXI4L interface, Disabled Transmit and Receive FIFO, Direct Write and Read Access, Generic Command Packet (To enable/disable XiP Mode of Flash)		System Clock Frequency: 125 MHz SPI Clock Frequency (max): 31.25 MHz (System Clock Frequency divide by 4)	Extended SPI Mode, XiP and non-XiP modes, Clock mode 0 and 3, 3-byte address mode	Page Program, Dual Input Fast Program, Extended Dual Input Fast Program, Quad Input Fast Program, Extended Quad Input Fast Program, Read Data, Fast Read, Dual Output Fast Read, Dual Input/Output Fast Read, Quad Output Fast Read, Quad Input/Output Fast Read	Final	Yes

Device Family	IP Configuration and Settings	On-board Flash	Clock Frequency	Flash Settings	Flash Commands	Radiant Timing Model	Hardware Validated
CertusPro-NX	AXI4 and AXI4L interface, Disabled Transmit and Receive FIFO, Direct Write and Read Access, Generic Command Packet (To enable/disable XiP Mode of Flash)	Winbond 512 Mbit SPI Flash (W25Q512JV)	System Clock Frequency: 125 MHz SPI Clock Frequency (max): 31.25 MHz (System Clock Frequency divide by 4)	Quad SPI, XiP and non-XiP modes, Clock mode 0 and 3, 3-byte and 4-byte address modes	Page Program, Quad Input Page Program, Read Data, Fast Read, Dual Output Fast Read, Dual Input/Output Fast Read, Quad Output Fast Read, Quad Input/Output Fast Read	Final	Yes
	AXI4 and AXI4L interface, Disabled Transmit and Receive FIFO, Supported Commands			Quad SPI, Clock mode 0 and 3, 3-byte and 4-byte address modes	Read ID (Read JEDEC ID), Read Manufacturer and Device ID (Read Device ID), Read Manufacturer and Device ID - Dual SPI, Read Manufacturer and Device ID - Quad SPI, Enter 4-byte address mode Exit 4-byte address mode	Final	Yes
	AHB-L and AXI4L interface, Disabled Transmit and Receive FIFO, Supported Commands	Macronix 512 Mbit SPI Flash (MX25L51245G)	System Clock Frequency: 125 MHz SPI Clock Frequency (max): 31.25 MHz (System Clock Frequency divide by 4)	Standard SPI, Clock mode 0 and 3,	Read ID, Read Electronic ID, Read Manufacturer and Device ID, Write Status Register, Read Status Register, Enter Quad Mode, Exit Quad Mode	Final	Yes
	AHB-L and AXI4L interface, Disabled Transmit and Receive FIFO, Direct Write and Read Access, Generic Command Packet (To enable/disable XiP Mode of Flash)		System Clock Frequency: 125 MHz SPI Clock Frequency (max): 31.25 MHz (System Clock Frequency divide by 4)	Quad SPI, XiP and non-XiP modes, Clock mode 0 and 3, 3-byte and 4-byte address modes	Page Program (QPI Mode), Quad Input/Output Fast Read (QPI Mode)	Final	Yes

Device Family	IP Configuration and Settings	On-board Flash	Clock Frequency	Flash Settings	Flash Commands	Radiant Timing Model	Hardware Validated
Avant	AXI4 interface, Disabled Transmit and Receive FIFO, Supported Commands	Winbond 512Mbit SPI Flash (W25Q512JV)	System Clock Frequency: 125 MHz SPI Clock Frequency (max): 31.25 MHz (System Clock Frequency divide by 4)	Quad SPI, Clock mode 3, 3-byte address mode	Read ID, Quad Input/Output Fast Read, Block Erase Type 1, Block Erase Type 3, Write Status Register, Quad Input Fast Program	Advance	Yes
	AXI4 and AXI4L interface, Disabled Transmit and Receive FIFO, Supported Commands		System Clock Frequency: 100 MHz SPI Clock Frequency (max): 25 MHz (System Clock Frequency divide by 4)	Quad SPI, Clock mode 3, 3-byte and 4-byte address mode	Read Manufacturer and Device ID Dual I/O, Read Manufacturer and Device ID Quad I/O, Read Status Register-1, Dual Input/Output Fast Read, Quad Input/Output Fast Read, Block Erase Type 1, Block Erase Type 3, Write Status Register, Page Program, Quad Input Fast Program	Advance	Yes

1.4. Features

Key features of the QSPI Flash Controller IP include:

- Different bus interfaces:
  - AMBA 3 AHB-Lite Protocol v1.0
    - 32-bit data and 32-bit address
    - Single and burst transfers. Burst transfer is only supported on direct access – non-XiP and XiP.
    - For non-XiP and XiP accesses, only a 32-bit data size is supported.
  - AMBA AXI4-Lite Protocol (for CSR access only)
    - 32-bit data and 32-bit address
    - Single transfer only
  - AMBA AXI4 Protocol
    - 32-bit data and 32-bit address
    - Single and multiple burst transfers. Burst transfer is only supported on direct access – non-XiP and XiP.
    - For non-XiP and XiP accesses, only a 32-bit data size is supported.
- Scalable performance: 1X, 2X, 4X I/O widths
- Programmable SPI clock mode of 0 or 3
- Programmable serial clock frequency
- Optional use of transmit and receive FIFOs with configurable FIFO depth for register-based access only
- RISC V QSPI code little endian and code execution
- Big- and little-endian support
- Supports flash commands that allow access and control to configuration, function, and other write registers
- Internal decoding of SPI flash instructions for supported flash commands
- Generic command packet to perform user-defined SPI transactions
- eXecute In Place (XiP) for read access
- Configurable SPI mode/protocols with the following interfaces:
  - Standard SPI – SCK, CS, SI, SO
  - Dual SPI – SCK, CS, IO0, IO1
  - Quad SPI – SCK, CS, IO0, IO1, IO2, IO3
- Standard SPI mode:
  - Controller mode only
  - MSB or LSB first transaction
  - Multiple-target environment
- Dual and Quad SPI modes:
  - Controller mode only
  - MSB first transaction
  - SPI transfer length of 8-bits only
  - Multiple-target environment
- Interrupt capability
- Supported system clock frequency up to 200 MHz. The maximum frequencies for supported Avant and Nexus devices are 200 MHz and 125 MHz, respectively. Refer to Appendix A for sample f_{MAX} values.

1.5. Licensing and Ordering Information

The QSPI Flash Controller IP is provided at no additional cost with the Lattice Radiant software.

1.5.1. Ordering Part Number

The QSPI Flash Controller IP does not require an ordering part number.

1.6. Hardware Support

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

1.7. Minimum Device Requirements

The minimum device requirements for the QSPI Flash Controller IP are as follows:

Table 1.3. Minimum Device Requirements for QSPI Flash Controller IP

Device	Speed Grades
LAV-AT-E70, LAV-AT-G70, LAV-AT-X70	1, 2, 3
LFD2NX-17, LFD2NX-40	7, 8, 9
LFCPNX-50, LFCPNX-100	7, 8, 9
LIFCL-17, LIFCL-33, LIFCL-40	7, 8, 9
LFMXO5-25, LFMXO5-55T, LFMXO5-100T	7, 8, 9
LN2-CT-20	1, 2, 3

1.8. Naming Conventions

1.8.1. Nomenclature

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

1.8.2. Signal Names

Signal names that end with:

- *_n* are active low (asserted when value is logic 0)
- *_i* are input signals
- *_o* are output signals
- *_io* are bi-directional input/output signals

1.8.3. Attribute Names

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

2. Functional Description

2.1. IP Architecture Overview

The QSPI Flash Controller IP core allows the host inside the FPGA to communicate with multiple external SPI flash devices using either standard, extended dual/quad, dual, or quad SPI protocols.

Figure 2.1 shows the block diagram of the QSPI Flash Controller IP.

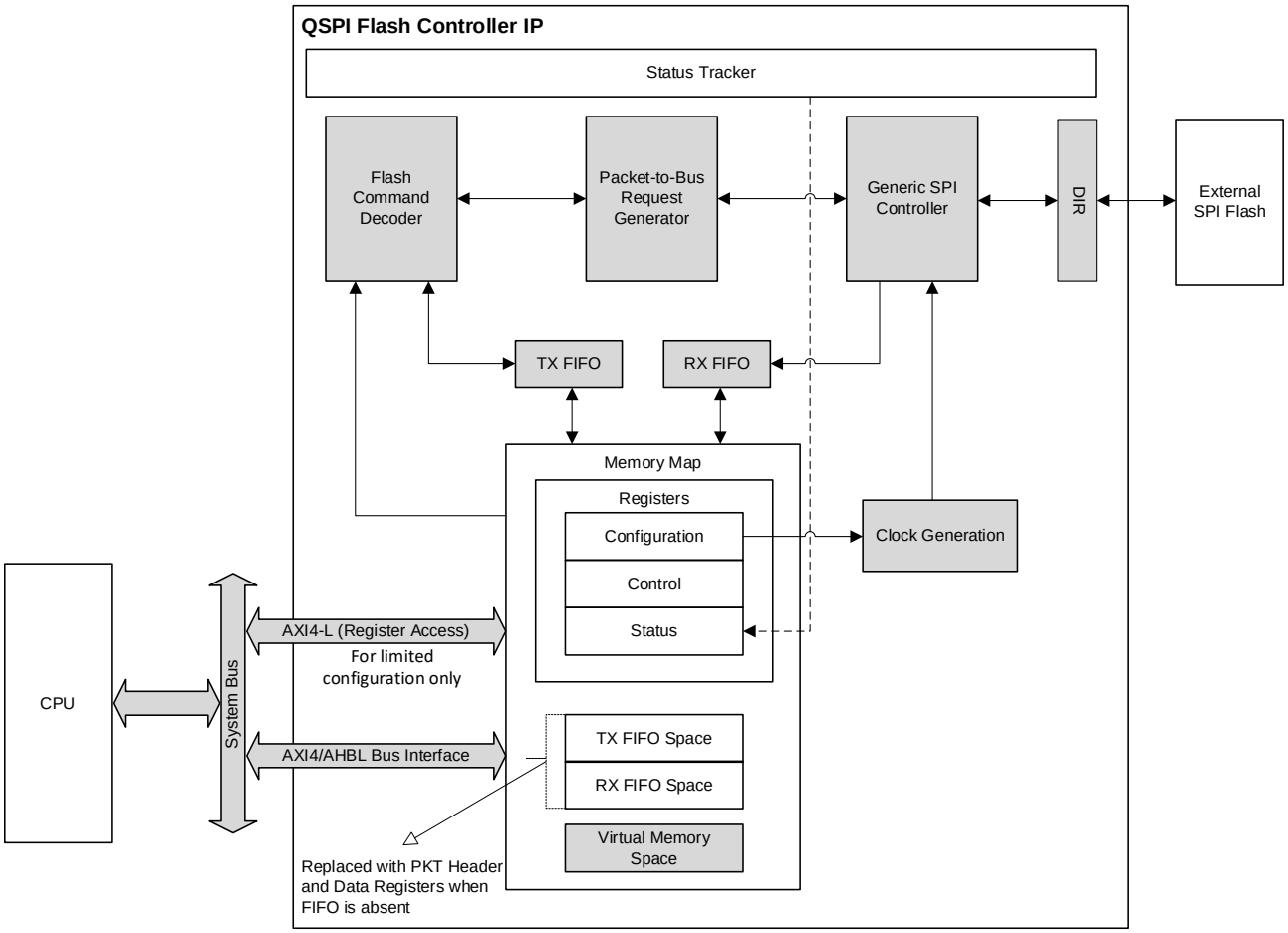

Figure 2.1. QSPI Flash Controller IP Block Diagram

2.2. Clocking

There are two clocks for this IP.

2.2.1. Clocking Overview

a_clk_i is the system clock that is also used as the AHB-Lite, AXI4-Lite, or AXI4 input clock. sclk_o is the serial clock to SPI flash.

2.3. Reset

There is only one reset for this IP.

2.3.1. Reset Overview

a_reset_n_i is an asynchronous active low reset. When low, 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
Advanced eXtensible Interface 4	AXI4	The AXI4 bus interface can be used as the main interface for all IP operations. When flash address mapping is enabled, this interface can only be used for non-XiP direct write/read access and XiP direct read access.
Advanced eXtensible Interface 4	AXI4-Lite	This interface is used when flash address mapping is enabled. Configuration, control and status register, and supported command transactions can use this interface.
Advanced High-performance Bus Interface	AHB-Lite	The AHB-L bus interface can be used as the main interface for all IP operations. When flash address mapping is enabled, this interface can only be used for non-XiP direct write/read access and XiP direct read access.
Serial Peripheral Interface	Standard, dual, and quad SPI	The Serial Peripheral Interface enables communication between the QSPI Flash Controller IP and target SPI flash device.

2.5. Other IP Specific Blocks/Layers/Interfaces

2.5.1. IP Core Operation

This IP supports standard, dual, and quad SPI transactions. During quad SPI-enabled protocol communication, standard and dual SPI transactions can still be processed by the controller. Likewise, during dual SPI-enabled protocol communication, standard SPI transactions can still be processed by the controller.

Controller sequences are mainly categorized for supported commands and generic SPI transactions.

This IP also provides an option to map the target flash device to an address space to mimic local memory access. When this is enabled, XiP access and non-XiP access can be performed by the controller using the selected IP's main interface.

2.5.1.1. Supported Commands Flash Operation

IP supported flash commands are listed in Table 5.29 including commonly used flash commands such as erase, page program, and read. For a flash operation, the flash_command_code bits of the packet header 0 register value are used by the controller to generate the complete sequence needed to execute the operation. The opcode used is based on the flash command opcode setting on the IP GUI or through the Flash Command Code 0 to 7 Registers with addresses 0x0000_000C to 0x0000_0028.

Below are the sample sequences automatically created by the controller based on flash_command_code register bits:

- flash_command_code == 5'h14, Write Enable flash command: The controller sends the equivalent opcode to the SPI interface.

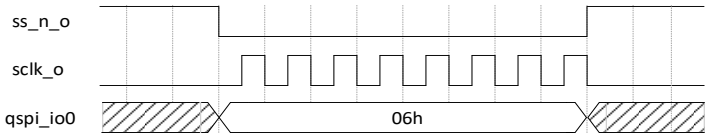

Figure 2.2. Write Enable Command 06h Supported Command Sequence on SPI Mode

- flash_command_code == 5'h0A, Read Data flash command: The controller sends the equivalent opcode and specified flash address to the SPI interface. The SPI transaction remains active until xfer_len_bytes of data are read from the flash.

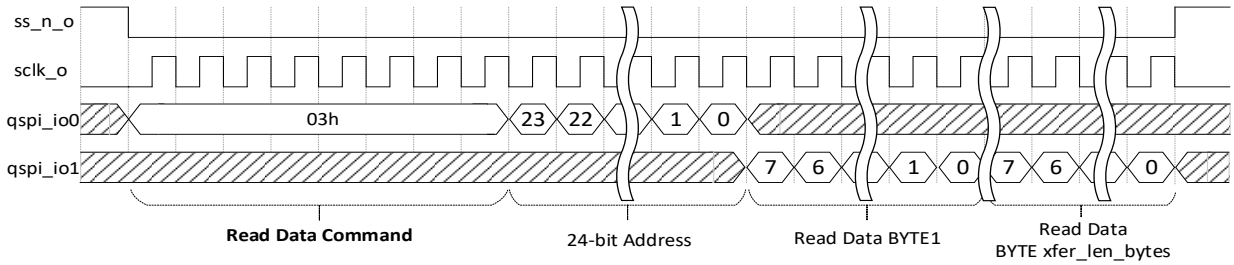

Figure 2.3. Read Data Command 03h Supported Command Sequence on SPI Mode

- flash_command_code == 5'h10, Block Erase Type 1 flash command: The controller sends the write enable opcode, equivalent erase opcode, specified flash address, and read status register opcode to the SPI interface. For erase flash transaction, the controller automatically sends the write enable and read status register flash command opcodes before and after sending the main flash command and address to the interface, respectively. The controller continues to send the read status register command until it reads that the flash is no longer busy.

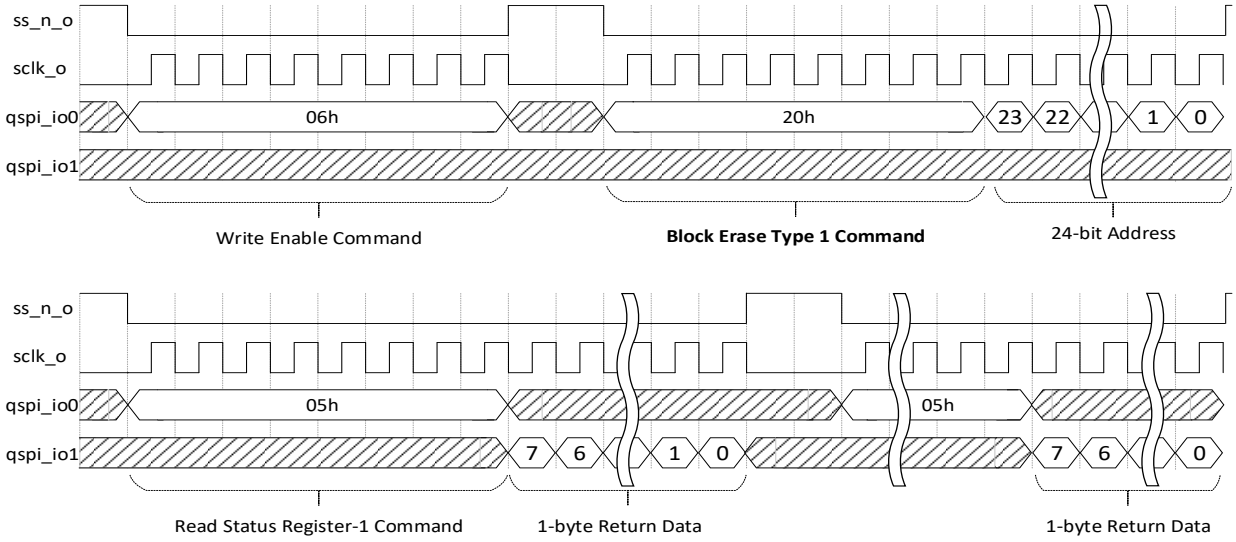

Figure 2.4. Block Erase Type 1 Command 20h Supported Command Sequence on SPI Mode

- flash_command_code == 5'h17, Page Program flash command: The controller sends the write enable opcode, equivalent page program opcode, and specified flash address. The SPI transaction remains active until all xfer_len_bytes of data from the packet data register or FIFO are sent to the flash after which the read status register opcode is sent to the SPI interface to end the flash operation. For write transaction, the controller automatically sends the write enable and read status register flash command opcodes before and after sending the main flash command, address, and data to the interface, respectively. The controller continues to send the read status register command until it reads that the flash is no longer busy.

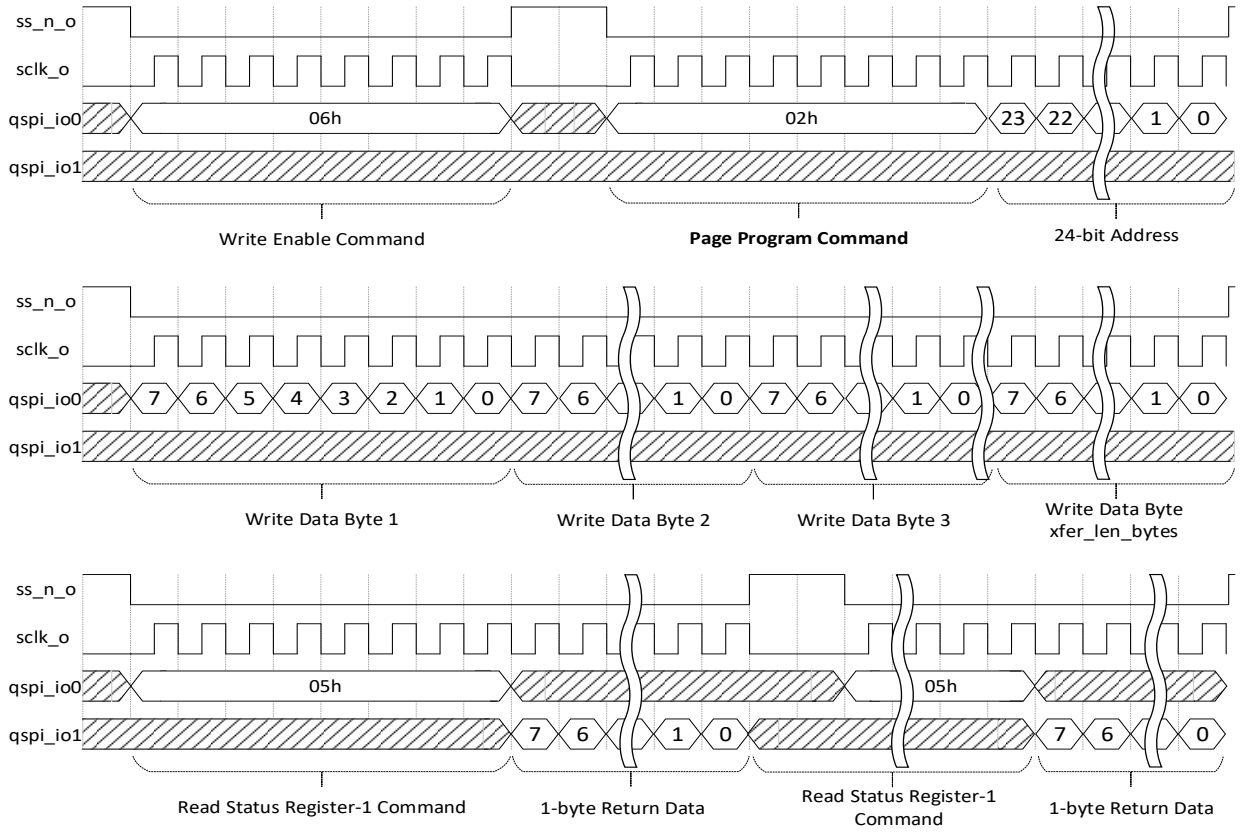

Figure 2.5. Page Program Command 02h Supported Command Sequence on SPI Mode

2.5.1.2. Generic Commands Operation

The IP can also process generic SPI transactions. The controller just sends and reads data to and from the SPI interface using packet data registers or FIFO based on the transaction described through the packet header 0 register for generic SPI transaction as listed in Table 5.30. Refer to the Generic Flash Write Operation and Generic Flash Read Operation sections for sample sequences on how use these features of the IP.

2.5.1.3. Non-XiP Access

This is a direct write or read access using AXI4 or AHB-L interface with INCR type burst operation to program to or read from flash. Single and burst transfers are supported for AHB-L and AXI4. When using this feature, both interfaces support a 32-bit data size only.

The program command can be set through the *Write Access Command* IP attribute or through the configuration register 0 flash_map_en_wr_access_cmd bits. The read command can be set through the *Read Access Command* IP attribute or through the configuration register 0 flash_map_en_rd_access_cmd bits. There are other IP attributes which should also be set to define the access to the flash such as *Flash Address Width*, *Command Lane Width*, *Address Lane Width*, *Data Lane Width*, and *Dummy clock cycles* equivalent to QSPI configuration register 0 bits [31:16] as described in the QSPI Configuration Register 0 section.

The write or read access address on the interface and the *Flash Address Mapping* attributes are used to determine the flash address targeted for the access. All *Flash Address Mapping* attributes are listed in Table 3.1. An example of non-XiP access is described in the Non-XiP Access (Direct Write/Read Access) through Flash Memory Map section.

2.5.1.4. eXecute-in-Place (XiP) Access

This is a direct read access using the AXI4 or AHB-L interface with INCR type burst operation read from flash using its XiP mode. Single and burst transfers are supported for AHB-L and AXI4. When using this feature, both interfaces support a 32-bit data size only.

Accessing flash devices via XiP mode allows for faster read execution because only an address is required during operation.

To use this IP feature, the flash device XiP mode must first be activated using generic command packet. Refer to the flash device data sheet on how to activate and deactivate its XiP mode.

Once the flash device XiP mode is activated, set the xip_enable bit on the start transaction register to perform XiP access. Through direct read access using the AXI4 or AHB-L interface, the controller performs direct read using the read command specified on the IP GUI during generation or through QSPI Configuration Register 0 bits with XiP-enabled flash. An example of XiP access is described in the XiP Access section.

2.5.2. User Packet Frame Structure

This section describes the packet structures used by the IP to generate SPI transactions. The packet header and data structure can also be written to the TX FIFO whenever the TX FIFO is enabled. Read data from the flash device can be stored in the RX FIFO when enabled or in the packet data register before the data is read through the interface.

For flash operations of supported commands, the number of dwords on the header depends on the flash address width.

Figure 2.6 shows the packet structure for supported commands, such as write and read status register, that do not require a flash address but has write or read data.

	7	6	5	4	3	2	1	0
Byte 0	multiple_flash_target	flash_cmd_code [4:0]					with_payload	sup_flash_cmd = 1
Byte 1	num_wait_state [7:0]							
Byte 2	xfer_len_bytes [7:0]							
Byte 3	xfer_len_bytes [15:8]							
Byte 4	*reserved [1:0]*		dat_lane_width [1:0]		adr_lane_width [1:0]		cmd_lane_width [1:0]	
Byte 5	flash_addr_width [2:0]				tgt_cs [4:0]			
Byte 6	*reserved [7:0]*							
Byte 7	*reserved [7:0]*							
Byte 8	payload [7:0]							
Byte 9	payload [15:8]							
...	...							
Byte 7+xfer_len_bytes	payload [(xfer_len_bytes*8) -1 : (xfer_len_bytes-1)*8)]							

Figure 2.6. Supported Flash Command Packet with 2DW Header

Figure 2.7 shows the packet structure for supported commands, such as page program and read data, with 32-bit flash address and write or read data.

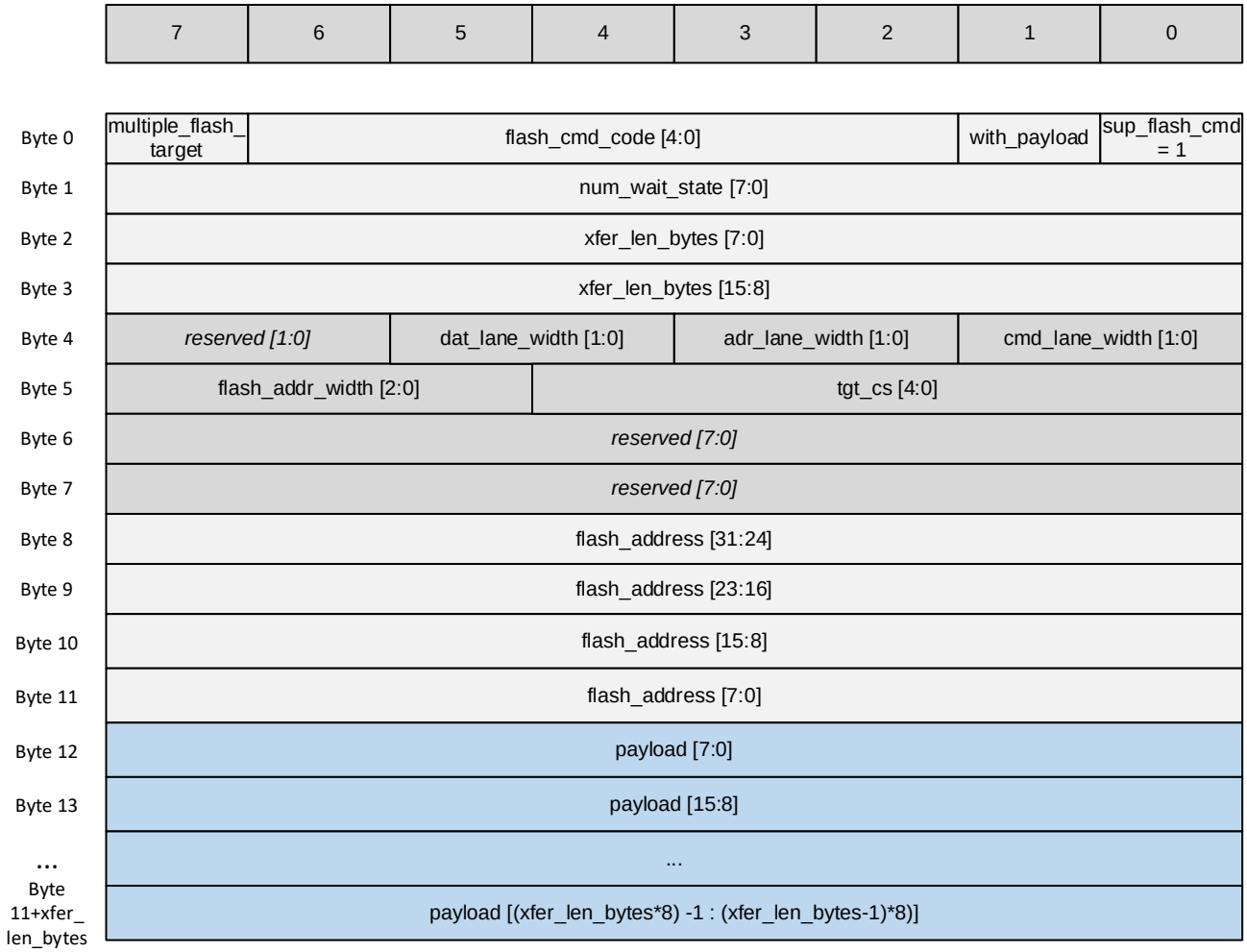

Figure 2.7. Supported Flash Command Packet with 3DW Header

Figure 2.8 shows the packet structure for supported commands, such as page program and read data, with more than 32-bit flash address and with write or read data.

	7	6	5	4	3	2	1	0
Byte 0	multiple_flash_target	flash_cmd_code [4:0]					with_payload	sup_flash_cmd = 1
Byte 1	num_wait_state [7:0]							
Byte 2	xfer_len_bytes [7:0]							
Byte 3	xfer_len_bytes [15:8]							
Byte 4	*reserved [1:0]*		dat_lane_width [1:0]		adr_lane_width [1:0]		cmd_lane_width [1:0]	
Byte 5	flash_addr_width [2:0]			tgt_cs [4:0]				
Byte 6	*reserved [7:0]*							
Byte 7	*reserved [7:0]*							
Byte 8	flash_address [63:56]							
Byte 9	flash_address [55:48]							
Byte 10	flash_address [47:40]							
Byte 11	flash_address [39:32]							
Byte 12	flash_address [31:24]							
Byte 13	flash_address [23:16]							
Byte 14	flash_address [15:8]							
Byte 15	flash_address [7:0]							
Byte 16	payload [7:0]							
Byte 17	payload [15:8]							
...	...							
Byte 15+xfer_len_bytes	payload [(xfer_len_bytes*8) -1 : (xfer_len_bytes-1)*8)]							

Figure 2.8. Supported Flash Command Packet with 4DW Header

Figure 2.9 shows the packet structure for generic SPI transaction where the controller only serializes the payload to the SPI interface or reads the data from the SPI interface.

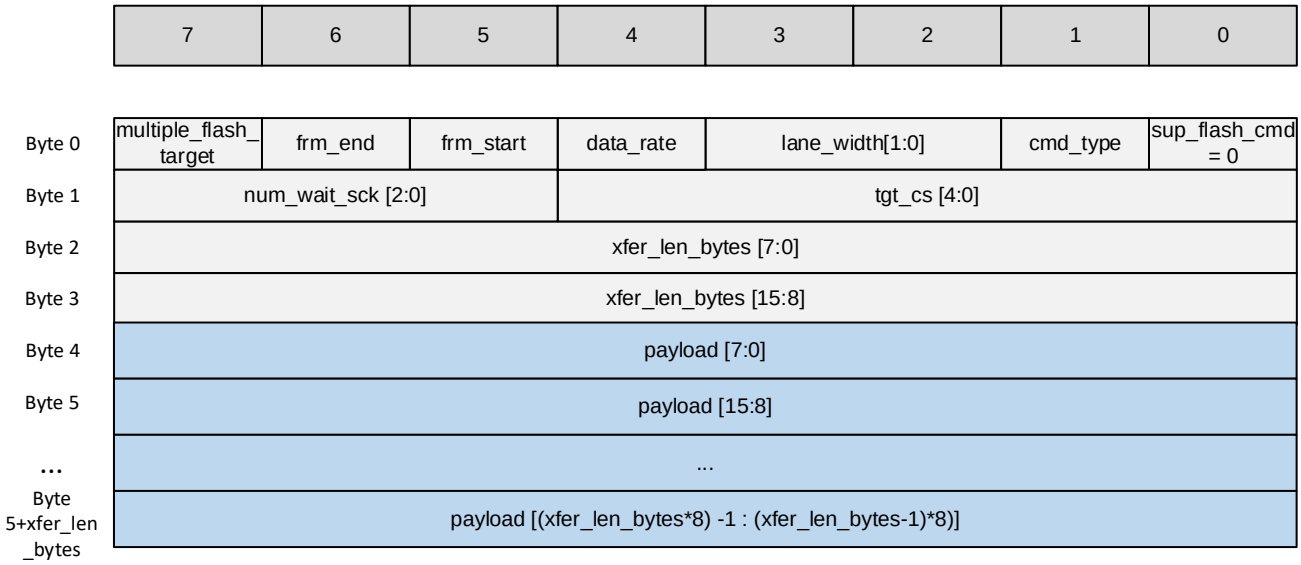

Figure 2.9. Generic Command Packet with 1DW Header

Table 5.28, Table 5.31, Table 5.32, Table 5.33, Table 5.34, and Table 5.35 show the packet header and data field definitions for flash operations of supported commands. Table 5.30, Table 5.34, and Table 5.35 show the packet header and data field definitions for generic SPI operations.

3. IP Parameter Description

The configurable attributes of the QSPI Flash Controller IP are shown in the following table. 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
Interface Settings		
Main Interface	**AHB-L**, AXI4	Selects memory-mapped interface to be used by the host. The unselected interfaces are not available in the generated IP.
Enable AXI4-L Interface (For register access only)	Checked	Use the AXI4-L for register access when flash address mapping is enabled. Informational only.
Interface Data Width	**32**	All interfaces use a 32-bit data width. Informational only.
AXI4 ID Width	0–32, **4**	Editable when *Main Interface == AXI4.*
Data Endianness	**Little-endian**, Big-endian	Endianness of the interface data port.
IO Primitive		
Enable IO Buffer	Checked, **Unchecked**	Enables bidirectional ports.
SPI Configuration and Transfer Setting		
Supported Protocol	**Standard,** Dual, Quad	Standard, dual, and quad SPI are the supported protocols. Extended dual and quad SPI are also supported. When quad SPI is selected, the IP can also operate on 1X and 2X I/O widths. Dual SPI can also operate on 1X I/O width.
Number of SPI Target	**1**–32	Specifies the number of target-select signal bits. Each bit is used for selecting an SPI target device.
Clock Mode	**0**, 3	SPI polarity and phase combination Specifies the polarity of the SPI serial clock during idle state. Also specifies whether the rising or falling edge of the SPI serial clock will be used to shift in and out data.
First Transmitted Bit	**MSB**, LSB	Specifies the direction of shifting the data bits into and out of the SPI interface. Editable if *Supported Protocol = Standard SPI.* Otherwise, value is fixed to MSB first transaction.
Chip Select High Time (ns)	**30**, …, 100	Specifies the minimum chip select high time.
Clock Frequency Settings		
System Clock Frequency (MHz)	1–**200**	Specifies the clock frequency of the interface.
Internal Clock Frequency (MHz)	1–**200**	Clock frequency to be used for core operation. Informational only. Value follows the *System Clock Frequency* attribute.
SPI Clock Frequency Divider	**2**, 4, 6, 8 … 62	Specifies the divider value to generate the SPI clock from the system clock.
SPI Clock Frequency (MHz)	*System Clock Frequency*/62–***System Clock Frequency/2***	Indicates the resulting SPI clock frequency based on system clock frequency and divider values. Informational only. *SPI Clock Frequency (MHz) = System Clock Frequency (MHz)/ SPI Clock Frequency Divider*
Transmit FIFO Configuration		
Enable Transmit FIFO	Checked, **Unchecked**	Enables transmit FIFO.
Address Depth	16, 32, 64, **128**	Specifies the number of FIFO levels. Visible and editable only when *Enable Transmit FIFO == Checked.*

Attribute	Selectable Values	Description
Receive FIFO Configuration		
Enable Receive FIFO	Checked, **Unchecked**	Enables receive FIFO.
Address Depth	16, 32, 64, 128, 256, **512**	Specifies the number of FIFO levels. Visible and editable only when *Enable Receive FIFO == Checked.*
Flash Address Mapping		
Enable Flash Address Mapping	Checked, **Unchecked**	Enables mapping of the flash device to the interface memory map.
Register Space Size (KB)	**1**	Memory map size for the registers of the IP. Informational only. Visible only when *Enable Flash Address Mapping == Checked*.
Register Map Base Address (Hex)	**00000000**	This is the starting address of the register space. Visible and editable only when *Enable Flash Address Mapping == Checked*.
Minimum Flash Address Alignment (KB)	**4**–1048576	Indicates the minimum flash address alignment when multiple target flash devices are enabled. Visible and editable only when *Enable Flash Address Mapping == Checked*.
Flash Memory Map Size (KB)	**4**–1048576	Indicates the size of the flash memory to be mapped. When multiple target flash devices are enabled, this attribute refers to the total flash address space (Minimum Flash Address Alignment × Number of SPI Target) rounded off to the nearest power of 2. Visible and editable only when *Enable Flash Address Mapping == Checked.*
Starting Flash Address (Hex)	**00000000**	Refers to the starting real flash memory address to be mapped. This should be based on the minimum flash address alignment when multiple target flash devices are enabled. Visible and editable only when *Enable Flash Address Mapping == Checked.* *Minimum Flash Address Alignment* should be reflected on this attribute. Otherwise, the IP GUI DRC displays an error message.
AXI/AHB-L Address Map (Hex)	**00001000**	AXI/AHB-L main interface starting address of the target memory map space. Visible and editable only when *Enable Flash Address Mapping == Checked.* This attribute should be aligned to *Flash Memory Map Size.* When using the IP on Propel, flash address map can be placed in the processor's cacheable range to perform burst access. This can be done by setting the *AXI/AHB-L Address Map* value within the cacheable region defined by the CPU. Refer to the RISC-V RX CPU IP User Guide (FPGA-IPUG-02254).
Write Access Command	**Page Program**, Dual Input Fast Program, Extended Dual Input Fast Program, Quad Input Fast Program, Extended Quad Input Fast Program	Write command used on non-XiP direct access. Visible and editable only when *Enable Flash Address Mapping == Checked.*
Read Access Command	Read Data, **Fast Read**, Dual Output Fast Read, Dual Input/Output Fast Read, Quad Output Fast Read, Quad Input/Output Fast Read	Read command used on non-XiP direct access. Visible and editable only when *Enable Flash Address Mapping == Checked.*
Flash Address Width	24, **32**	Flash address width used for non-XiP access and XiP access. Visible and editable only when *Enable Flash Address Mapping == Checked.*

Attribute	Selectable Values	Description
Command Lane Width	**x1**, x2, x4	Number of lanes used by the controller for sending flash command opcode on non-XiP and XiP direct access. Visible and editable only when *Enable Flash Address Mapping == Checked.*
Address Lane Width	**x1**, x2, x4	Number of lanes used by the controller for sending flash address on non-XiP and XiP direct access. Visible and editable only when *Enable Flash Address Mapping == Checked.*
Data Lane Width	**x1**, x2, x4	Number of lanes used by the controller for sending and receiving data on non-XiP and XiP direct access. Visible and editable only when *Enable Flash Address Mapping == Checked.*
Dummy clock cycles	0, 2, 4, 6, **8**, …, 14	Number of clock cycles added after the controller sends address during read access operation on non-XiP and XiP direct access. Visible and editable only when *Enable Flash Address Mapping == Checked.*
eXecute-in-Place (XiP) Pattern	**8'hA5**	Indicates the byte value used by the target flash device to enable XiP mode. Check the data sheet of the flash device to be used for this information. Visible and editable only when *Enable Flash Address Mapping == Checked*.
Supported Flash Commands		
Read Status Register-1	**8'h05**	List of flash read, erase, and write commands that are supported on the soft logic of the IP.
Read Status Register-2	**8'h35**	
Read Status Register-3	**8'h15**	
Read Configuration Register	**8'hB5**	
Read ID	**8'h9F**	
Read Electronic ID	**8'hAB**	
Multiple I/O Read ID	**8'hAF**	
Read Manufacturer and Device ID	**8'h90**	
Read Manufacturer and Device ID Dual I/O	**8'h92**	
Read Manufacturer and Device ID Quad I/O	**8'h94**	
Read Data	**8'h03**	
Fast Read	**8'h0B**	
Dual Output Fast Read	**8'h3B**	
Dual Input/Output Fast Read	**8'hBB**	
Quad Output Fast Read	**8'h6B**	
Quad Input/Output Fast Read	**8'hEB**	
Block Erase Type 1	**8'h20**	
Block Erase Type 2	**8'h52**	
Block Erase Type 3	**8'hD8**	
Chip Erase	**8'h60**	
Write Enable	**8'h06**	
Write Disable	**8'h04**	
Write Status/Configuration Register	**8'h01**	
Page Program	**8'h02**	
Dual Input Fast Program	**8'hA2**	
Extended Dual Input Fast Program	**8'hD2**	
Quad Input Fast Program	**8'h32**	

Attribute	Selectable Values	Description
Extended Quad Input Fast Program	**8'h38**	
Enter 4-Byte Address Mode	**8'hB7**	
Exit 4-Byte Address Mode	**8'hE9**	
Enter Quad Input/Output Mode	**8'h35**	
Reset Quad Input/Output Mode	**8'hF5**	

3.2. IP Parameter Settings for Example Use Cases

3.2.1. SPI Protocols

The IP core supports different types of SPI protocols: standard, extended (dual or quad), dual, and quad SPI. Support for these SPI protocols are distinguished by the number of data lanes. Standard SPI uses the common four signals: CS, SCK, SI and SO, which are equivalent to the IP's ss_n_o, sclk_o, io0_i, and io0_o signals, as shown in Figure 3.1. Extended SPI supports the standard SPI operations and extends it to support multiple data lanes in address, dummy, and data phase as shown in Figure 3.2. Dual SPI uses two tri-state data lanes (IO0 and IO1 which are equivalent to IP's qspi_io0 and qspi_io1 signals) to double the performance of standard SPI as shown in Figure 3.3. Quad SPI uses four tri-state data lanes (IO0, IO1, IO2, and IO3 which are equivalent to IP's qspi_io0, qspi_io1, qspi_io2, and qspi_io3 signals) to quadruple the performance of standard SPI as shown in Figure 3.4.

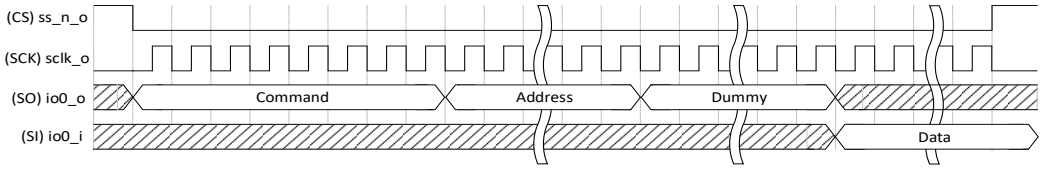

Figure 3.1. Standard SPI Timing Diagram

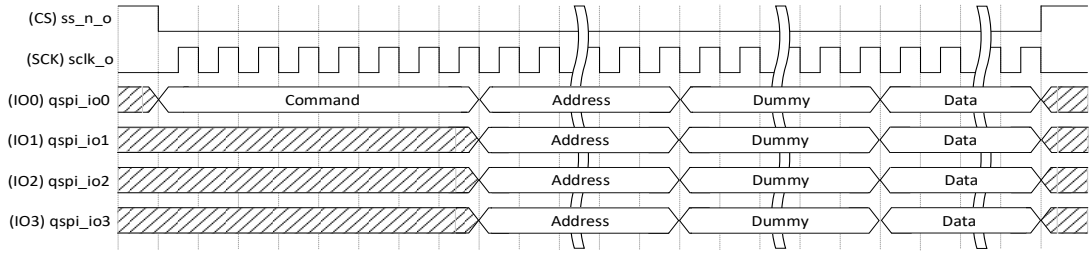

Figure 3.2. Extended SPI Timing Diagram

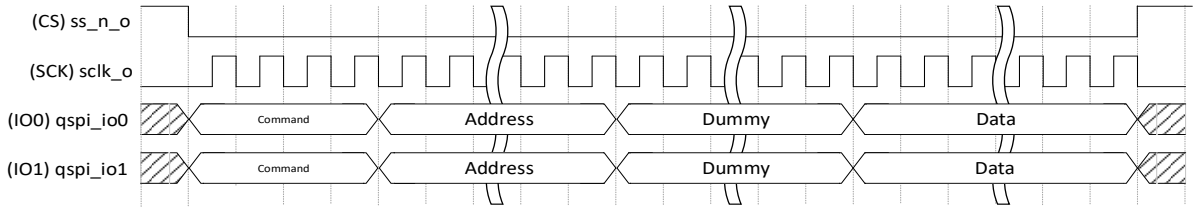

Figure 3.3. Dual SPI Timing Diagram

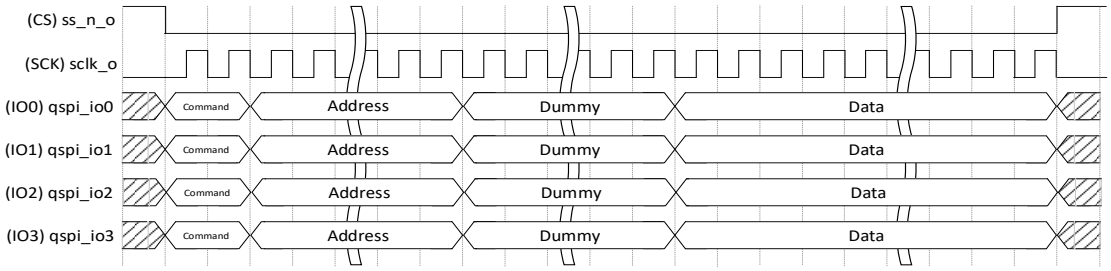

Figure 3.4. Quad SPI Timing Diagram

3.2.2. SPI Clock Phase and Polarity

SPI clock polarity and SPI clock phase can be set through the IP GUI *Clock Mode* attribute or configuration register. SPI clock polarity sets the polarity of the sclk_o signal during the idle state (ss_n_o bit is inactive). Transitioning of the ss_n_o signal from inactive logic to active logic marks the start of the transmission and transitioning from active logic to inactive logic marks the end of the transmission. SPI clock phase selects the sclk_o phase. It sets whether the rising or falling clock edge is used to sample and shift the data. The IP must be set to the SPI clock polarity and SPI clock phase corresponding to the requirements of the flash. There are two SPI clocking modes available, which are shown in Table 3.2.

Table 3.2. Clocking Modes

Clocking Mode	SPI Clock Polarity	SPI Clock Phase	Description
0	0	0	The ss_n_o signal transition to active logic shifts out a data bit. The first clock transition samples the data. Data is sampled on the sclk_o rising edge and is shifted out on the falling edge.
3	1	1	The first clock transition shifts out a data bit and the second clock transition samples the data. Data is sampled on the sclk_o rising edge and is shifted out on the falling edge.

The sample waveforms for the SPI clocking modes are shown in Figure 3.5 and Figure 3.6.

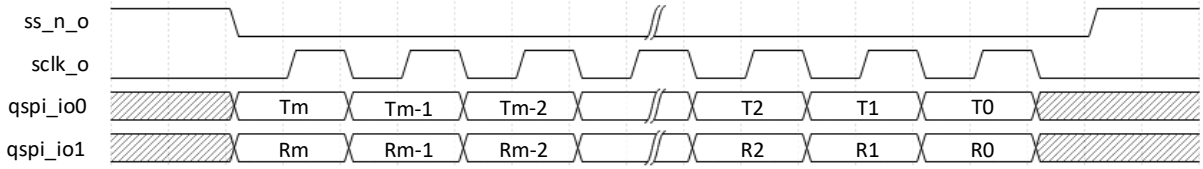

Figure 3.5. Clocking Mode 0 (SPI Clock Polarity=0, SPI Clock Phase=0) Waveforms

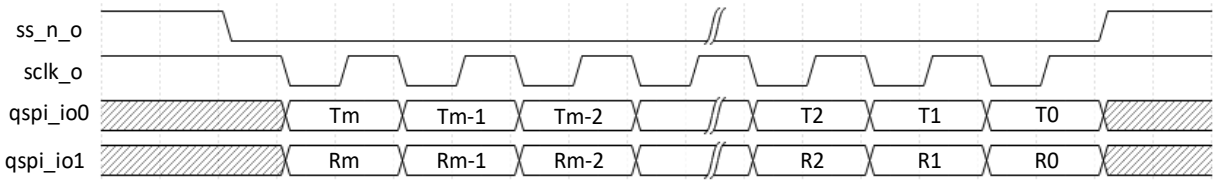

Figure 3.6. Clocking Mode 3 (SPI Clock Polarity=1, SPI Clock Phase=1) Waveforms

3.2.3. Big and Little Endian

The data byte order in the bus interface can be configured through the IP GUI *Data Endianness* attribute or by programming the data_endianness register bit of the configuration register 0 with offset address 0x0000_0004.

In little endian, the lowest significant byte is stored in the lowest address. In big endian, the most significant byte is stored in the lowest address.

Table 3.3 shows an example on how the controller interprets an AXI4 address 0x0000_1000 with AXI4 Data of 0x1234_5678 on both data endianness. Note that the controller will always send the data from the lowest byte address first on the SPI interface.

Table 3.3. Bus Interface Data Endianness

Byte Address	Data Stored	
	Little Endian	Big Endian
0x0000_1000	0x78	0x12
0x0000_1001	0x56	0x34
0x0000_1002	0x34	0x56
0x0000_1003	0x12	0x78

3.2.4. MSB or LSB First Transmitted Bit Transaction

The *First Transmitted Bit* attribute defines the first bit of the data byte to be send on the SPI interface, which is either MSB or LSB.

3.2.5. Multiple SPI Targets

The *Number of Target* attribute defines the number of targets connected to the controller. When there are multiple targets, the controller IP generates a multi-bit ss_n_o signal where each bit is dedicated to the chip select pin of a target device.

For register-based flash transactions, the IP checks if the multiple_flash_target packet header 0 register bit field is enabled to activate the specific target by asserting only one bit of the ss_n_o signal at a time determined through the tgt_cs packet header 1 register bit field value. Refer to the Packet Header 0 Register for Supported Flash Commands, Packet Header 0 Register for Generic SPI Transactions, and Packet Header 1 Register sections.

When using the IP in direct access mode and with the number of SPI targets greater than 1, the IP automatically decodes the target device based on the AXI/AHB-L address map and the minimum flash address alignment. For example, when the number of SPI target devices is two (each target device size is 4 KB), the IP enables the first target when the lower 4 KB flash memory map is being accessed; the IP enables the second target when the upper 4 KB flash memory map is being accessed. Refer to the AXI/AHB-L Address Map Register section.

3.2.6. Transmit and Receive FIFO

FIFOs are optional. When TX FIFO is enabled, depth can be set up to 128. This depth cannot be fully used on the programming flow because the IP only supports 256 bytes for flash program operations with an additional depth for packet header values. When RX FIFO is enabled, depth of up to 512 levels can be used to store read flash data. This feature allows for writing of data to the TX FIFO before start of a page program operation and to the RX FIFO after start of a read data operation to flash before the data is fetched to the interface.

When disabled, the IP has packet data register that assumes the role of the FIFO. Two packet data registers can be used for programming operations while only one packet data register can be used for read operations.

4. Signal Description

This section describes the QSPI Flash Controller IP ports.

Table 4.1. Top-Level Ports

Port	Width	Type	Output Reset Value	Description
Clock and Reset				
a_clk_i	1	Input	—	System clock
a_reset_n_i	1	Input	—	Asynchronous active low reset
Interrupt				
int_o	1	Output	1'b0	Interrupt signal
SPI				
sclk_o	1	Output	*SPI Clock Polarity*	SPI controller clock output
ss_n_o	*Number of SPI Target*	Output	1	SPI controller target select output Active-low one-hot encoded
Disabled IO Buffer				
io0_i	1	Input	—	SPI data input io0 if Standard/Dual/Quad SPI mode is selected.
io1_i	1	Input	—	SPI data input io1 if Dual/Quad SPI mode is selected.
io2_i	1	Input	—	SPI data input io2 if Quad SPI mode is selected.
io3_i	1	Input	—	SPI data input io3 if Quad SPI mode is selected.
io0_o	1	Output	1'b0	SPI data output io0 if Standard/Dual/Quad SPI mode is selected.
io1_o	1	Output	1'b0	SPI data output io1 if Dual/Quad SPI mode is selected.
io2_o	1	Output	1'b0	SPI data output io2 if Quad SPI mode is selected.
io3_o	1	Output	1'b0	SPI data output io3 if Quad SPI mode is selected.
io0_oe_o	1	Output	1'b0	Direction control for io0 I/O pad (1=output, 0=input).
io1_oe_o	1	Output	1'b0	Direction control for io1 I/O pad (1=output, 0=input).
io2_oe_o	1	Output	1'b0	Direction control for io2 I/O pad (1=output, 0=input).
io3_oe_o	1	Output	1'b0	Direction control for io3 I/O pad (1=output, 0=input).
Enabled IO Buffer				
qspi_io0	1	Inout	—	SPI data inout io0 if Standard/Dual/Quad SPI mode is selected.
qspi_io1	1	Inout	—	SPI data inout io1 if Dual/Quad SPI mode is selected.
qspi_io2	1	Inout	—	SPI data inout io2 if Quad SPI mode is selected.
qspi_io3	1	inout	—	SPI data inout io3 if Quad SPI mode is selected.

Port	Width	Type	Output Reset Value	Description
AHB-Lite Interface				
ahbl_hsel_i	1	Input	—	AHB-Lite select signal Indicates the device is selected and transfer is required.
ahbl_hready_i	1	Input	—	AHB-Lite ready input signal Indicates data phase of previous transfer is completed.
ahbl_haddr_i	32	Input	—	AHB-Lite address signal
ahbl_hburst_i	3	Input	—	AHB-Lite burst type signal Indicates if the transfer is a single transfer or forms part of a burst.
ahbl_hsize_i	3	Input	—	AHB-Lite transfer size signal Indicates the size of the transfer that is a byte, halfword, or word.
ahbl_hmastlock_i	1	Input	—	AHB-Lite lock signal This signal is unused.
ahbl_hprot_i	4	Input	—	AHB-Lite protection control This signal is unused.
ahbl_htrans_i	2	Input	—	AHB-Lite transfer type signal Indicates the transfer type of the current transfer.
ahbl_hwrite_i	1	Input	—	AHB-Lite transfer direction signal Indicates the transfer direction. When HIGH, a write transfer; when LOW, a read transfer.
ahbl_hwdata_i	32	Input	—	AHB-Lite write data signal
ahbl_hreadyout_o	1	Output	1'b1	AHB-Lite ready output signal Indicates that a transfer has finished on the bus. The subordinate uses this signal to extend an AHB-Lite transfer.
ahbl_hrdata_o	32	Output	32'b0	AHB-Lite read data signal
ahbl_hresp_o	1	Output	1'b0	AHB-Lite transfer response signal
AXI4-Lite Interface				
axil_awaddr_i	32	Input	—	AXI4-Lite write address
axil_awvalid_i	1	Input	—	AXI4-Lite write address valid Indicates that the write address is valid.
axil_awprot	3	Input	—	AXI4-Lite write address protect Indicates the protection attributes of a write transaction: privilege, security level, and access type. This signal is unused.
axil_awready_o	1	Output	1'b0	AXI4-Lite write address ready Indicates that a transfer on the write address channel can be accepted.
axil_wdata_i	32	Input	—	AXI4-Lite write data
axil_wvalid_i	1	Input	—	AXI4-Lite write valid Indicates that the write data is valid.
axil_wstrb_i	4	Input	—	AXI4-Lite write strobe Indicates which byte lanes hold valid data.
axil_wready_o	1	Output	1'b0	AXI4-Lite write ready Indicates that a transfer on the write data channel can be accepted.
axil_bresp_o	2	Output	2'b00	AXI4-Lite write response Indicates the status of a bus write transaction.
axil_bvalid_o	1	Output	1'b0	AXI4-Lite write response valid Indicates that the write response is valid.

Port	Width	Type	Output Reset Value	Description
axil_bready_i	1	Input	—	AXI4-Lite write response ready Indicates that a transfer on the write response channel can be accepted.
axil_araddr_i	32	Input	—	AXI4-Lite read address
axil_arvalid_i	1	Input	—	AXI4-Lite read address valid Indicates that the read address is valid.
axil_arprot	3	Input	—	AXI4-Lite read address protect Indicates the protection attributes of a read transaction: privilege, security level, and access type. This signal is unused.
axil_arready_o	1	Output	1'b0	AXI4-Lite read address ready Indicates that a transfer on the read address channel can be accepted.
axil_rdata_o	32	Output	All 0	AXI4-Lite read data
axil_rvalid_o	1	Output	1'b0	AXI4-Lite read data valid Indicates that the read data is valid.
axil_rresp_o	2	Output	2'b00	AXI4-Lite read data response Indicates the status of a bus read transaction.
axil_rready_i	1	Input	—	AXI4-Lite read data ready Indicates that a transfer on the read data channel can be accepted.
AXI4 Interface				
AXI4 Write Address Channel				
axi_awid_i	*AXI4 ID Width*	Input	—	AXI4 write address ID Indicates the identification tag for a write transaction.
axi_awaddr_i	32	Input	—	AXI4 write address Indicates the address of the first transfer in a write transaction.
axi_awlen_i	8	Input	—	AXI4 write address length Indicates the exact number of data transfers in a write transaction.
axi_awsize_i	3	Input	—	AXI4 write address size Indicates the number of bytes in each data transfer in a write transaction.
axi_awburst_i	2	Input	—	AXI4 write address burst Indicates how address changes between each transfer in a write transaction. Only FIXED burst type access is supported.
axi_awlock_i	1	Input	—	AXI4 write address lock Provides information about the atomic characteristics of a write transaction. This signal is unused.
axi_awcache_i	4	Input	—	AXI4 write address cache Indicates how a write transaction is required to progress through a system. This signal is unused.
axi_awprot_i	3	Input	—	AXI4 write address protect Indicates the protection attributes of a write transaction: privilege, security level, and access type. This signal is unused.

Port	Width	Type	Output Reset Value	Description
axi_awvalid_i	1	Input	—	AXI4 write address valid Indicates that the write address channel signals are valid.
axi_awready_o	1	Output	1'b0	AXI4 write address ready Indicates that a transfer on the write address channel can be accepted.
AXI4 Write Data Channel				
axi_wid_i	*AXI4 ID Width*	Input	—	AXI4 write data ID Indicates the ID tag of the write data transfer.
axi_wdata_i	32	Input	—	AXI4 write data
axi_wstrb_i	4	Input	—	AXI4 write data strobe Indicates which byte lanes hold valid data.
axi_wlast_i	1	Input	—	AXI4 write data last Indicates whether this is the last data transfer in a write transaction.
axi_wvalid_i	1	Input	—	AXI4 write data valid Indicates that the write data channel signals are valid.
axi_wready_o	1	Output	1'b0	AXI4 write data ready Indicates that a transfer on the write data channel can be accepted.
AXI4 Write Response Channel				
axi_bid_o	*AXI4 ID Width*	Output	0	AXI4 write response ID Indicates the identification tag for a write response.
axi_bresp_o	2	Output	2'b00	AXI4 write response Indicates the status of a write transaction.
axi_bvalid_o	1	Output	1'b0	AXI4 write response valid Indicates that the write response channel signals are valid.
axi_bready_i	1	Input	—	AXI4 write response ready Indicates that a transfer on the write response channel can be accepted.
AXI4 Read Address Channel				
axi_arid_i	4	Input	—	AXI4 read address ID Indicates the identification tag for a read transaction.
axi_araddr_i	32	Input	—	AXI4 read address Indicates the address of the first transfer in a read transaction.
axi_arlen_i	8	Input	—	AXI4 read address length Indicates the exact number of data transfers in a read transaction.
axi_arsize_i	3	Input	—	AXI4 read address size Indicates the number of bytes in each data transfer in a read transaction.
axi_arburst_i	2	Input	—	AXI4 read address burst Indicates how address changes between each transfer in a read transaction. Only FIXED burst type in RD_DATA_REG access is supported.
axi_arlock_i	1	Input	—	AXI4 read address lock Provides information about the atomic characteristics of a read transaction.

Port	Width	Type	Output Reset Value	Description
				This signal is unused.
axi_arcache_i	4	Input	—	AXI4 read address cache Indicates how a read transaction is required to progress through a system. This signal is unused.
axi_arprot_i	3	Input	—	AXI4 read address protect Indicates the protection attributes of a read transaction: privilege, security level, and access type. This signal is unused.
axi_arvalid_i	1	Input	—	AXI4 read address valid Indicates that the read address channel signals are valid.
axi_arready_o	1	Output	1'b0	AXI4 read address ready Indicates that a transfer on the read address channel can be accepted.
AXI4 Read Data Channel				
axi_rid_o	*AXI4 ID Width*	Input	0	AXI4 read data ID Indicates the ID tag of the read data transfer.
axi_rdata_o	32	Output	All 0	AXI4 read data
axi_rresp_o	2	Output	2'b00	AXI4 read data response Indicates the status of a read transfer.
axi_rlast_o	1	Output	1'b0	AXI4 read data last Indicates whether this is the last data transfer in a read transaction.
axi_rvalid_o	1	Output	1'b0	AXI4 read data valid Indicates that the read data channel signals are valid.
axi_rready_i	1	Input	—	AXI4 read data ready Indicates that a transfer on the read data channel can be accepted.

5. Register Description

5.1. Overview

The host can control the QSPI Flash Controller IP core by writing and reading through the 32-bit address space. The registers are classified as configuration, status, interrupt, and control registers.

Table 5.1 defines the register access types. Table 5.2 lists the address map and specifies the registers available.

Table 5.1. Register Access Types

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

Table 5.2. Summary of QSPI Flash Controller IP Core Registers

Offset Address	Register Name	Description
Configuration Registers		
0x0000_0000	Reserved	Reserved address for configuration register
0x0000_0004	QSPI Configuration Register 0	Programmable registers to configure standard/dual/quad SPI transactions
0x0000_0008	QSPI Configuration Register 1	Programmable registers to configure standard/dual/quad SPI transactions
0x0000_000C	Flash Command Code 0	Supported flash read command codes
0x0000_0010	Flash Command Code 1	Supported flash read command codes
0x0000_0014	Flash Command Code 2	Supported flash read command codes
0x0000_0018	Flash Command Code 3	Supported flash read command codes
0x0000_001C	Flash Command Code 4	Supported flash write command codes
0x0000_0020	Flash Command Code 5	Supported flash write command codes
0x0000_0024	Flash Command Code 6	Supported flash write command codes
0x0000_0028	Flash Command Code 7	Supported flash write command codes
0x0000_002C	Minimum Flash Address Alignment	Flash address alignment size
0x0000_0030	Starting Flash Address	Starting actual flash memory address
0x0000_0034	Flash Memory Map Size	Size of the virtual flash memory space
0x0000_0038	AXI/AHB-L Address Map	Address map for flash memory
0x0000_003C to 0x0000_00FF	Reserved	Reserved addresses for configuration registers
Status and Interrupt Registers		
0x0000_0100	Transaction Status	Indicates the IP transaction status
0x0000_0104	Interrupt Status	Interrupt capability
0x0000_0108	Interrupt Enable	Interrupt capability
0x0000_010C	Interrupt Set	Interrupt capability
0x0000_0110	Supported Flash Command Packet Header and Data Transfer Status	Supported flash command packet transfer counter
0x0000_0114	Generic SPI Transactions Packet Header and Data Transfer Status	Generic SPI transaction packet transfer counter
0x0000_0118 to 0x0000_01FF	Reserved	Reserved addresses for status and interrupt registers

Offset Address	Register Name	Description
Control Registers		
0x0000_0200	TX FIFO Mapping	Transmit FIFO mapping
0x0000_0204	RX FIFO Mapping	Receive FIFO mapping
0x0000_0208	Packet Header 0	User packet structure for supported flash commands and generic SPI transactions
0x0000_020C	Packet Header 1	User packet structure for supported flash commands
0x0000_0210	Packet Header 2	User packet structure for supported flash commands
0x0000_0214	Packet Header 3	User packet structure for supported flash commands
0x0000_0218	Packet Data 0	User packet structure for supported flash commands and generic SPI transactions
0x0000_021C	Packet Data 1	User packet structure for supported flash commands and generic SPI transactions
0x0000_0220	Start Transaction	Start IP operation to generate SPI transaction
0x0000_0224 to 0x0000_02FF	Reserved	Reserved addresses for control registers

5.2. Configuration Registers

These programmable registers must be held static during QSPI Flash Controller IP core operations. Any change on these registers would require a reset.

For sections from Flash Command Code 0 Register through Flash Command Code 7 Register, commands included are based on flash devices such as Micron MT25QU128, Macronix MX25L12833F and MX25L51245G, and Winbond W25Q512JV.

5.2.1. QSPI Configuration Register 0

Table 5.3 lists the register bit fields for SPI transfer configuration, clock frequency divider, data endianness of the interface, and frame end done counter. Bits [15:0] are used on all types of transactions while bits [31:16] are specific for non-XiP access and XiP access, which are available only if *Enable Flash Address Mapping* is selected on the IP GUI. Thus, bits [31:16] are read-only when *Enable Flash Address Mapping* is not selected.

Table 5.3. QSPI Configuration Register 0

Field	Name	Description	Access	Default
[31:30]	dat_lane_width	Data can be sent to target flash memory in different lane widths during non-XiP and XiP access. 2'b00 – Use x1 lane. 2'b01 – Use x2 lane. 2'b10 – Use x4 lane.	RW	*Data Lane Width*
[29:28]	addr_lane_width	Address can be sent to target flash memory in different lane widths during non-XiP and XiP access. 2'b00 – Use x1 lane. 2'b01 – Use x2 lane. 2'b10 – Use x4 lane.	RW	*Address Lane Width*
[27:26]	cmd_lane_width	Command can be sent to target flash memory in different lane widths during non-XiP and XiP access. 2'b00 – Use x1 lane. 2'b01 – Use x2 lane. 2'b10 – Use x4 lane.	RW	*Command Lane Width*
[25]	xip_addr_width	Flash address width used by the controller for non-XiP and XiP access. 1'b0 – 24-bit flash address 1'b1 – 32-bit flash address	RW	*Flash Address Width*

Field	Name	Description	Access	Default
[24:22]	dummy_clock_cycles	Number of dummy clock cycles used by the controller for non-XiP and XiP read access. Number of Dummy Clock Cycles = Value of Setting × 2 3'b000 – 0 × 2 dummy clock cycles 3'b001 – 1 × 2 dummy clock cycles 3'b010 – 2 × 2 dummy clock cycles … 3'b111 – 7 × 2 dummy clock cycles	RW	*Dummy Clock Cycles*/2
[21:19]	flash_map_en_rd_access_cmd	This command, whose opcode is based on Flash Command 2 and 3 registers, is used by the controller for non-XiP and XiP flash read access. 3'b000 – Read Data 3'b001 – Fast Read 3'b010 – Dual Output Fast Read 3'b011 – Dual Input/Output Fast Read 3'b100 – Quad Output Fast Read 3'b101 – Quad Input/Output Fast Read	RW	*Read Access Command*
[18:16]	flash_map_en_wr_access_cmd	This command, whose opcode is based on Flash Command 5 and 6 registers, is used by the controller for non-XiP and XiP flash write access. 3'b000 – Page Program 3'b001 – Dual Input Fast Program 3'b010 – Extended Dual Input Fast Program 3'b011 – Quad Input Fast Program 3'b100 – Extended Quad Input Fast Program	RW	*Write Access Command*
[15]	en_frame_end_done_cntr	Enables frame end as done counter for executing generic SPI transaction. 1'b0 – The generic SPI transaction status register increments whenever generic packet header and data are set and processed by the controller. 1'b1 – The generic SPI transaction status register only increments when a frame end is detected on packet header 0 register.	RW	0x0
[14]	reserved	Reserved	RO	0x0
[13]	data_endianness	0 – little-endian 1 – big-endian	RW	*Data Endianness*
[12:8]	sck_rate	Clock divider value. 5'b0_0001 – Divide by 2 of sck_src_i 5'b0_0010 – Divide by 4 of sck_src_i … SPI Clock Frequency = sck_src_i / (2 × sck_rate) The default value of the sck_rate register is based on the GUI setting of *SPI Clock Frequency Divider* decoded as 5-bit register. When *SPI Clock Frequency Divider = 2*, sck_rate default value is 1.	RW	*SPI Clock Frequency Divider*/2
[7:3]	reserved	Reserved	RO	0x0
[2]	cpha	Clock phase. Specifies clock edge for shifting out data in io*_o and sampling data in io*_i. Only SPI clock modes 0 and 3 are supported. Set this register accordingly together with the cpol register bit. 1'b0 – Sampling of data occurs at odd edges (1, 3, 5, …, 15) of the sclk_o clock. 1'b1 – Sampling of data occurs at even edges (2, 4, 6, …, 16) of the sclk_o clock.	RW	*Clock Mode* Bit 0

Field	Name	Description	Access	Default
[1]	cpol	Clock polarity. Specifies the logic level of sclk_o during idle. Only SPI clock modes 0 and 3 are supported. Set this register accordingly together with the cpha register bit. 1'b0 – Active-low clocks selected. In idle state, sclk_o is low. 1'b1 – Active-high clocks selected. In idle state, sclk_o is high.	RW	*Clock Mode* Bit 1
[0]	first_bit_transfer	1'b0 – Data is transferred with MSB first. 1'b1 – Data is transferred with LSB first.	RW	*First Transmitted Bit*

5.2.2. QSPI Configuration Register 1

Table 5.4. QSPI Configuration Register 1

Field	Name	Description	Access	Default
[31:16]	tgt_rd_trans_cnt	Target count for read transaction using supported or generic SPI commands to generate an interrupt. This should be a non-zero value. It is initially set at the start of operation and should only be changed after a previous tgt_rd_trns_cnt_hit_int is already cleared.	RW	0x1
[15:0]	tgt_wr_trans_cnt	Target count for write transaction using supported or generic SPI commands to generate an interrupt. This should be a non-zero value. It is initially set at the start of operation and should only be changed after a previous tgt_wr_trns_cnt_hit_int is already cleared.	RW	0x1

5.2.3. Flash Command Code 0 Register

This register is used by the IP on supported command transactions only.

Table 5.5. Flash Command Code 0 Register

Field	Name	Description	Access	Default
[31:24]	read_status_register_1_cmd _code	Read commands for status and configuration registers. Flash devices have at least one readable status register. Check the data sheet of the target flash device to determine if the commands shown are also supported then update the corresponding command code on the IP GUI or registers accordingly. These registers are the actual command opcode of a particular flash device. Values at reset depend on the command opcode set on the IP GUI.	RW	*Read Status Register-1*
[23:16]	read_status_register_2_cmd _code		RW	*Read Status Register-2*
[15:8]	read_status_register_3_cmd _code		RW	*Read Status Register-3*
[7:0]	read_configuration_register _cmd_code		RW	*Read Configuration Register*

5.2.4. Flash Command Code 1 Register

This register is used by the IP on supported command transactions only.

Table 5.6. Flash Command Code 1 Register

Field	Name	Description	Access	Default
[31:24]	read_id_cmd_code	Read ID commands. Check the data sheet of the target flash device to determine if the commands shown are also supported then update the corresponding command code on the IP GUI or registers accordingly. These registers are the actual command opcode of a	RW	*Read ID*
[23:16]	read_electronic_id_cmd_code		RW	*Read Electronic ID*
[15:8]	multiple_io_read_id_cmd_code		RW	*Multiple I/O Read ID*

[7:0]	read_manufacturer_device_id_cmd_code	particular flash device. Values at reset depend on the command opcode set on the IP GUI.	RW	*Read Manufacturer and Device ID*

5.2.5. Flash Command Code 2 Register

This register is used by the IP on supported command transactions only.

Table 5.7. Flash Command Code 2 Register

Field	Name	Description	Access	Default
[31:24]	read_manufacturer_device_id_dual_io_cmd_code	Read ID and data commands. Check the data sheet of the target flash device to determine if the commands shown are also supported then update the corresponding command code on the IP GUI or registers accordingly. For some flash devices, read data commands can have different opcodes depending on the address mode such as 3- or 4-byte address. These registers are the actual command opcode of a particular flash device. Values at reset depend on the command opcode set on the IP GUI.	RW	*Read Manufacturer and Device ID Dual I/O*
[23:16]	read_manufacturer_device_id_quad_io_cmd_code		RW	*Read Manufacturer and Device ID Quad I/O*
[15:8]	read_data_cmd_code		RW	*Read Data*
[7:0]	fast_read_cmd_code		RW	*Fast Read*

5.2.6. Flash Command Code 3 Register

This register is used by the IP on supported command transactions only.

Table 5.8. Flash Command Code 3 Register

Field	Name	Description	Access	Default
[31:24]	dual_output_fast_read_cmd_code	Read data commands. Check the data sheet of the target flash device to determine if the commands shown are also supported then update the corresponding command code on the IP GUI or registers accordingly. For some flash devices, read data commands can have different opcodes depending on the address mode such as 3- or 4-byte address. These registers are the actual command opcode of a particular flash device. Values at reset depend on the command opcode set on the IP GUI.	RW	*Dual Output Fast Read*
[23:16]	dual_input_output_fast_read_cmd_code		RW	*Dual Input/Output Fast Read*
[15:8]	quad_output_fast_read_cmd_code		RW	*Quad Output Fast Read*
[7:0]	quad_input_output_fast_read_cmd_code		RW	*Quad Input/Output Fast Read*

5.2.7. Flash Command Code 4 Register

This register is used by the IP on supported command transactions only.

Table 5.9. Flash Command Code 4 Register

Field	Name	Description	Access	Default
[31:24]	block_erase_type_1_cmd_code	Erase commands. Check the data sheet of the target flash device to determine if the commands shown are also supported then update the corresponding command code on the IP GUI or registers accordingly. These registers are the actual command opcode of a particular flash device. Values at reset depend on the command opcode set on the IP GUI.	RW	*Block Erase Type 1*
[23:16]	block_erase_type_2_cmd_code		RW	*Block Erase Type 2*
[15:8]	block_erase_type_3_cmd_code		RW	*Block Erase Type 3*
[7:0]	chip_erase_cmd_code		RW	*Chip Erase*

5.2.8. Flash Command Code 5 Register

This register is used by the IP on supported command transactions only.

Table 5.10. Flash Command Code 5 Register

Field	Name	Description	Access	Default
[31:24]	write_enable_cmd_code	Write operation commands. Check the data sheet of the target flash device to determine if the commands shown are also supported then update the corresponding command code on the IP GUI or registers accordingly. For some flash devices, page program command can have different opcodes depending on the address mode such as 3- or 4-byte address. These registers are the actual command opcode of a particular flash device. Values at reset depend on the command opcode set on the IP GUI. **Note:** For page program flash command, the IP only supports continuous byte write up to the flash page size. If the number of bytes is more than the flash page size, you need to access the packet header and data for another set of page program.	RW	*Write Enable*
[23:16]	write_disable_cmd_code		RW	*Write Disable*
[15:8]	write_status_configuration _register_cmd_code		RW	*Write Status Register*
[7:0]	page_program_cmd_code		RW	*Page Program*

5.2.9. Flash Command Code 6 Register

This register is used by the IP on supported command transactions only.

Table 5.11. Flash Command Code 6 Register

Field	Name	Description	Access	Default
[31:24]	dual_input_fast_program_cmd _code	Page program commands. Check the data sheet of the target flash device to determine if the commands shown are also supported then update the corresponding command code on the IP GUI or registers accordingly. For some flash devices, page program commands can have different opcodes depending on the address mode such as 3- or 4-byte address. These registers are the actual command opcode of a particular flash device. Values at reset depend on the command opcode set on the IP GUI.	RW	*Dual Input Fast Program*
[23:16]	extended_dual_input_fast _program_cmd_code		RW	*Extended Dual Input Fast Program*
[15:8]	quad_input_fast_program _cmd_code		RW	*Quad Input Fast Program*
[7:0]	extended_quad_input_fast _program_cmd_code		RW	*Extended Quad Input Fast Program*

5.2.10. Flash Command Code 7 Register

This register is used by the IP on supported command transactions only.

Table 5.12. Flash Command Code 7 Register

Field	Name	Description	Access	Default
[31:24]	enter_4_byte_address_mode _cmd_code	Flash commands to configure the flash on different modes. Check the data sheet of the target flash device to determine if the commands shown are also supported then update the corresponding command code on the IP GUI or registers accordingly. These registers are the actual command opcode of a particular flash device. Values at reset depend on the command opcode set on the IP GUI.	RW	*Enter 4-Byte Address Mode*
[23:16]	exit_4_byte_address mode_ cmd_code		RW	*Exit 4-Byte Address Mode*
[15:8]	enter_quad_input_output_ mode_cmd_code		RW	*Enter Quad Input/Output Mode*
[7:0]	reset_quad_input_output_ mode_cmd_code		RW	*Reset Quad Input/Output Mode*

5.2.11. Minimum Flash Address Alignment Register

Table 5.13. Minimum Flash Address Alignment Register

Field	Name	Description	Access	Default
[31:0]	min_flash_addr_alignment	Flash address alignment in terms of KB size. Values are in power of 2. Starts at 1 KB, 2 KB, 4 KB, …, 32 KB. Available only if *Enable Flash Address Mapping* is selected on the IP GUI. This register identifies the target address alignment based on the size of each SPI target on the memory map. The lower bits that are within the memory map size value are set to 0 while the upper bits are set to 1. For example, if the memory map size is 4KB (0x1000), then the lower 12 bits ([11:0]) correspond to the offset and the upper bits [31:12] correspond to the base address. Thus, the min_flash_addr_alignment value is 0xFFFF_F000.	RW	0xFFFF_F000

5.2.12. Starting Flash Address Space Mapping Register

Table 5.14. Starting Flash Address Space Mapping Register

Field	Name	Description	Access	Default
[31:0]	start_flash_address_space_map	Starting flash address of the flash memory to be mapped to the address space. Available only if *Enable Flash Address Mapping* is selected on the IP GUI. This is the actual starting address of the flash memory map, which must be aligned to the minimum flash address alignment register value. Below are sample starting flash address values when minimum flash address alignment value is 4 KB: 0x0000_0000 – Accesses the SPI target 0x0000_0000 to 0x0000_0FFF address which is the first 4KB partition of the flash. 0x0000_1000 – Accesses the SPI target 0x0000_1000 to 0x0000_1FFF address which is the second 4KB partition of the flash. From the samples above, notice that the lower 12 bits ([11:0]) are always 0, the same with the minimum flash address alignment requirement.	RW	*Starting Flash Address*

5.2.13. Flash Memory Map Size Register

Table 5.15. Flash Memory Map Size Register

Field	Name	Description	Access	Default
[31:0]	flash_memory_map_size	This setting identifies the total size of the flash memory to be mapped to the interface memory map. For single target only, it should have the same value as the minimum target address alignment. The lower bits that are within the memory map size value are set to 0 while the upper bits are set to 1. For example, if the total memory map size is 64KB (0x10000), then the lower 16 bits ([15:0]) correspond to the offset and the upper bits [31:16] correspond to the base address. Thus, the flash_memory_map_size value is 0xFFFF_0000. Available only if *Enable Flash Address Mapping* is selected on the IP GUI.	RW	0xFFFF_F000

5.2.14. AXI/AHB-L Address Map Register

Table 5.16. AXI/AHB-L Address Map Register

Field	Name	Description	Access	Default
[31:0]	axi_ahbl_address_map	AXI/AHB-L base address of the target memory map. This register value indicates if the address being accessed on the main interface is a direct or non-direct access. When flash address mapping is enabled on the IP GUI, with *Number of SPI Target* set to 4, *Minimum Flash Address Alignment* of 4 KB, and starting flash address of 0x0000_1000, the resulting *Flash Memory Map Size* is 16 KB (0x4000). When set via register, min_flash_addr_alignment and flash_memory_map_size should be set as 0xFFFF_F000 and 0xFFFF_C000 respectively. Therefore, the AXI/AHB-L address map value should have the lower 14 bits ([13:0]) set to 0. Notice that the AXI/AHB-L address map value follows the requirement of the flash_memory_map_size. The AXI/AHB-L address map value (together with the minimum flash address alignment) is also used by the IP to control the ss_n_o behavior for multiple SPI targets. From the sample above, when the AXI/AHB-L address map is set to 0x0000_4000, the mapping for each SPI target is as follows: • Direct access on AXI/AHB-L address of 0x0000_4000 – 0x0000_4FFF selects Target 0 through ss_n_o[0]. • Direct access on AXI/AHBL address of 0x0000_5000 – 0x0000_5FFF selects Target 1 through ss_n_o[1]. • Direct access on AXI/AHBL address of 0x0000_6000 – 0x0000_6FFF selects Target 2 through ss_n_o[2]. • Direct access on AXI/AHBL address of 0x0000_7000 – 0x0000_7FFF selects Target 3 through ss_n_o[3]. Available only if *Enable Flash Address Mapping* is selected on the IP GUI.	RW	*AXI/AHB-L Address Map*

5.3. Status and Interrupt Registers

The interrupts of the IP are implemented based on the Lattice Interrupt Interface definition.

5.3.1. Transaction Status Register

This register defines the IP status when a transaction begins.

Table 5.17. Transaction Status Register

Field	Name	Description	Access	Default
[31:1]	reserved	Reserved	RO	0x0
[0]	trans_stat	Run status of the IP 1'b0 – Idle state. If previous transfer has already started, this indicates that transfer is done. 1'b1 – Transfer has started and is still running.	RO	0x0

5.3.2. Interrupt Status Register

5.3.2.1. Interrupt Status Register when FIFO is Enabled

When FIFO is enabled, this register defines the Receive and Transmit FIFO interrupt statuses. An interrupt status asserts only when a change in the corresponding FIFO status occurs.

Table 5.18. Interrupt Status Register when FIFO is Enabled

Field	Name	Description	Access	Default
[31:10]	reserved	Reserved	RO	0x0
[9]	tgt_rd_trns_cnt_hit_int	Target read transaction count hit interrupt. 1'b0 – Read transaction counter is not equal to tgt_rd_trans_cnt. 1'b1 – Read transaction counter is equal to tgt_rd_trans_cnt.	RW1C	0x0
[8]	tgt_wr_trns_cnt_hit_int	Target write transaction count hit interrupt. 1'b0 – Write transaction counter is not equal to tgt_wr_trans_cnt. 1'b1 – Write transaction counter is equal to tgt_wr_trans_cnt.	RW1C	0x0
[7]	rx_fifo_full_int	Receive FIFO full interrupt. This interrupt asserts when FIFO status changes from non-empty to full. 1'b0 – Indicates that data available on the Receive FIFO is still less than Receive FIFO Address Depth × 4 bytes after the read transaction has started. 1'b1 – Indicates that Receive FIFO Address Depth × 4 bytes of data are already available on the Receive FIFO after the read transaction has started.	RW1C	0x0
[6]	rx_fifo_almost_full_int	Receive FIFO almost full interrupt. This interrupt asserts when FIFO status changes from non-empty to almost full. 1'b0 – Indicates that data available on the Receive FIFO is still less than (Receive FIFO Address Depth – 4) × 4 bytes after the read transaction has started. 1'b1 – Indicates that (Receive FIFO Address Depth – 4) × 4 bytes of data are already available on the Receive FIFO after the read transaction has started.	RW1C	0x0
[5]	rx_fifo_almost_empty_int	Receive FIFO almost empty interrupt. This interrupt asserts when FIFO status changes from non-empty to almost empty. 1'b0 – Indicates that the remaining data available on the Receive FIFO after the read transaction completion and reading on the FIFO has started is still greater than 1 dword or 4 bytes only.	RW1C	0x0

Field	Name	Description	Access	Default
		1'b1 – Indicates that the remaining data available on the Receive FIFO after the read transaction completion and reading on the FIFO has started is already 1 dword or 4 bytes only.		
[4]	rx_fifo_empty_int	Receive FIFO empty interrupt. This interrupt asserts when FIFO status changes from non-empty to empty. 1'b0 – Indicates that there are still data available on Receive FIFO after read transaction completion. 1'b1 – Indicates that Receive FIFO Address Depth × 4 bytes of data are already read after read transaction completion.	RW1C	0x0
[3]	tx_fifo_full_int	Transmit FIFO full interrupt. This interrupt asserts when FIFO status changes from non-empty to full. 1'b0 – Indicates that data stored on the Transmit FIFO is still less than Transmit FIFO Address Depth × 4 bytes before the transaction has started. 1'b1 – Indicates that Transmit FIFO Address Depth × 4 bytes of data are already stored on the Transmit FIFO before the transaction has started.	RW1C	0x0
[2]	tx_fifo_almost_full_int	Transmit FIFO almost full interrupt. This interrupt asserts when FIFO status changes from non-empty to almost full. 1'b0 – Indicates that data stored on the Transmit FIFO is still less than (Transmit FIFO Address Depth – 4) × 4 bytes before the transaction has started. 1'b1 – Indicates that (Transmit FIFO Address Depth – 4) × 4 bytes of data are already stored on the Transmit FIFO before the transaction has started.	RW1C	0x0
[1]	tx_fifo_almost_empty_int	Transmit FIFO almost empty interrupt. This interrupt asserts when FIFO status changes from non-empty to almost empty. 1'b0 – Indicates that the remaining data on Transmit FIFO is greater than 1 dword or 4 bytes after the transaction has started. 1'b1 – Indicates that the remaining data available on the Transmit FIFO after the transaction has started is already 1 dword or 4 bytes only.	RW1C	0x0
[0]	tx_fifo_empty_int	Transmit FIFO empty interrupt. This interrupt asserts when FIFO status changes from non-empty to empty. 1'b0 – Indicates that there are still data bytes available on the Transmit FIFO after the transaction started. 1'b1 – Indicates that all data bytes are already sent from Transmit FIFO to SPI bus after the transaction started.	RW1C	0x0

5.3.2.2. Interrupt Status Register when FIFO is Disabled

When FIFO is disabled, this register defines the packet data interrupt status during IP operations.

Table 5.19. Interrupt Status Register when FIFO is Disabled

Field	Name	Description	Access	Default
[31:10]	reserved	Reserved	RO	0x0
[9]	tgt_rd_trns_cnt_hit_int	Target read transaction count hit interrupt. 1'b0 – Read transaction counter is not equal to tgt_rd_trans_cnt. 1'b1 – Read transaction counter is equal to tgt_rd_trans_cnt.	RW1C	0x0
[8]	tgt_wr_trns_cnt_hit_int	Target write transaction count hit interrupt. 1'b0 – Write transaction counter is not equal to tgt_wr_trans_cnt. 1'b1 – Write transaction counter is equal to tgt_wr_trans_cnt.	RW1C	0x0
[7:5]	reserved	Reserved	RO	0x0
[4]	rd_pkt_data_0_not_empty_int	Interrupt status of packet data during read operations. 1'b0 – Read packet data register is empty. 1'b1 – There is already an available read data on read packet data register.	RW1C	0x0
[3:2]	reserved	Reserved	RO	0x0
[1]	wr_pkt_data_1_empty_int	Interrupt status of packet data during write operations with payload. Once write transaction is completed, this packet data register interrupt status is reset to 1. This interrupt status should not be cleared if any transaction without payload is executed. 1'b0 – There is already available data in packet data that can be transmitted. 1'b1 – Packet data is empty. New data can be written.	RW1C	0x1
[0]	wr_pkt_data_0_empty_int	Interrupt status of packet data during write operations with payload. Once write transaction is completed, this packet data register interrupt status is reset to 1. This interrupt status should not be cleared if any transaction without payload is executed. 1'b0 – There is already available data in packet data that can be transmitted. 1'b1 – Packet data is empty. New data can be written.	RW1C	0x1

5.3.3. Interrupt Enable Register

5.3.3.1. Interrupt Enable Register when FIFO is Enabled

When FIFO is enabled, this register defines the Receive and Transmit FIFO interrupt enable.

Table 5.20. Interrupt Enable Register when FIFO is Enabled

Field	Name	Description	Access	Default
[31:10]	reserved	Reserved	RO	0x0
[9]	tgt_rd_trns_cnt_hit_int_en	Enables target transaction count hit interrupt.	RW	0x0
[8]	tgt_wr_trns_cnt_hit_int_en	Enables target transaction count hit interrupt.	RW	0x0
[7]	rx_fifo_full_int_en	Enables Receive FIFO full interrupt.	RW	0x0
[6]	rx_fifo_almost_full_int_en	Enables Receive FIFO almost full interrupt.	RW	0x0
[5]	rx_fifo_almost_empty_int_en	Enables Receive FIFO almost empty interrupt.	RW	0x0
[4]	rx_fifo_empty_int_en	Enables Receive FIFO empty interrupt.	RW	0x0
[3]	tx_fifo_full_int_en	Enables Transmit FIFO full interrupt.	RW	0x0
[2]	tx_fifo_almost_full_int_en	Enables Transmit FIFO almost full interrupt.	RW	0x0
[1]	tx_fifo_almost_empty_int_en	Enables Transmit FIFO almost empty interrupt.	RW	0x0
[0]	tx_fifo_empty_int_en	Enables Transmit FIFO empty interrupt.	RW	0x0

5.3.3.2. Interrupt Enable Register when FIFO is Disabled

When FIFO is disabled, this register defines the packet data interrupt enable during IP operations.

Table 5.21. Interrupt Enable Register when FIFO is Disabled

Field	Name	Description	Access	Default
[31:10]	reserved	Reserved	RO	0x0
[9]	tgt_rd_trns_cnt_hit_int_en	Enables target transaction count hit interrupt.	RW	0x0
[8]	tgt_wr_trns_cnt_hit_int_en	Enables target transaction count hit interrupt.	RW	0x0
[7:5]	reserved	Reserved	RO	0x0
[4]	rd_pkt_data_0_not_empty_int_en	Interrupt enable of packet data during read operations.	RW	0x0
[3:2]	reserved	Reserved	RO	0x0
[1]	wr_pkt_data_1_empty_int_en	Interrupt enable of packet data during write operations with payload.	RW	0x0
[0]	wr_pkt_data_0_empty_int_en	Interrupt enable of packet data during write operations with payload.	RW	0x0

5.3.4. Interrupt Set Register

5.3.4.1. Interrupt Set Register when FIFO is Enabled

When FIFO is enabled, this register defines the Receive and Transmit FIFO interrupt set.

Table 5.22. Interrupt Set Register when FIFO is Enabled

Field	Name	Description	Access	Default
[31:10]	reserved	Reserved	RO	0x0
[9]	tgt_rd_trns_cnt_hit_int_set	Sets target transaction count hit interrupt.	RW	0x0
[8]	tgt_wr_trns_cnt_hit_int_set	Sets target transaction count hit interrupt.	RW	0x0
[7]	rx_fifo_full_int_set	Sets Receive FIFO full interrupt.	RW	0x0
[6]	rx_fifo_almost_full_int_set	Sets Receive FIFO almost full interrupt.	RW	0x0
[5]	rx_fifo_almost_empty_int_set	Sets Receive FIFO almost empty interrupt.	RW	0x0
[4]	rx_fifo_empty_int_set	Sets Receive FIFO empty interrupt.	RW	0x0
[3]	tx_fifo_full_int_set	Sets Transmit FIFO full interrupt.	RW	0x0
[2]	tx_fifo_almost_full_int_set	Sets Transmit FIFO almost full interrupt.	RW	0x0
[1]	tx_fifo_almost_empty_int_set	Sets Transmit FIFO almost empty interrupt.	RW	0x0
[0]	tx_fifo_empty_int_set	Sets Transmit FIFO empty interrupt.	RW	0x0

5.3.4.2. Interrupt Set Register when FIFO is Disabled

When FIFO is disabled, this register defines the packet data interrupt set during IP operations.

Table 5.23. Interrupt Set Register when FIFO is Disabled

Field	Name	Description	Access	Default
[31:10]	reserved	Reserved	RO	0x0
[9]	tgt_rd_trns_cnt_hit_int_set	Sets target transaction count hit interrupt.	RW	0x0
[8]	tgt_wr_trns_cnt_hit_int_set	Sets target transaction count hit interrupt.	RW	0x0
[7:5]	reserved	Reserved	RO	0x0
[4]	rd_pkt_data_0_not_empty_int_set	Interrupt set of packet data during read operations.	RW	0x0
[3:2]	reserved	Reserved	RO	0x0
[1]	wr_pkt_data_1_empty_int_set	Interrupt set of packet data during write operations with payload.	RW	0x0
[0]	wr_pkt_data_0_empty_int_set	Interrupt set of packet data during write operations with payload.	RW	0x0

5.3.5. Supported Flash Command Packet Header and Data Transfer Status Register

This register tracks the number of supported flash command packet headers and data sets transferred during write and read operations.

Table 5.24. Supported Flash Command Transaction Status Register

Field	Name	Description	Access	Default
[31:16]	sup_flash_cmd_pkt_header_data _xfer_wr_ cntr	Supported flash command packet header and data transfer during write operations counter. This counter increments when a set of packet header and data is transferred by the controller to the external flash.	RO	0x0
[15:0]	sup_flash_cmd_pkt_header_data _xfer_rd_ cntr	Supported flash command packet header and data transfer during read operations counter. This counter increments when a packet data is transferred by the controller and when data is received from the flash memory.	RO	0x0

5.3.6. Generic SPI Transaction Packet Header and Data Transfer Status Register

This register tracks the number of generic packet headers and data sets transferred during write and read operations.

Table 5.25. Generic SPI Transaction Status Register

Field	Name	Description	Access	Default
[31:16]	gen_flash_cmd_pkt_header_data _xfer_wr_ cntr	Generic SPI transaction packet header and data transfer during write operations counter. This counter increments when a packet data is transferred by the generic SPI controller.	RO	0x0
[15:0]	gen_flash_cmd_pkt_header_data _xfer_rd_ cntr	Generic SPI transaction packet header and data transfer during read operations counter. This counter increments when a packet data is received from target flash.	RO	0x0

5.4. Control Registers

Control registers include Transmit or Receive FIFO mapping, packet header and data, and start transaction registers.

5.4.1. TX FIFO Mapping

Table 5.26. TX FIFO Mapping Register

Field	Name	Description	Access	Default
[31:0]	tx_fifo_data	This register should be used only when *Enable Transmit FIFO* == Checked. When FIFO is enabled, data to be written to the FIFO should follow the user packet frame structure described in the User Packet Frame Structure section. Endianness mode applies to the register bit fields of the user packet frame structure that corresponds to packet header 2 and 3 and packet data 0 and 1 registers.	WO	0x0

5.4.2. RX FIFO Mapping

Table 5.27. RX FIFO Mapping Register

Field	Name	Description	Access	Default
[31:0]	rx_fifo_data	This register should be used only when *Enable Receive FIFO* == Checked. Endianness mode applies to the read data from the FIFO.	RO	0x0

5.4.3. Packet Header 0 Register

The bit fields of this register should be set depending on the supported flash command to be executed.

5.4.3.1. Packet Header 0 Register for Supported Flash Commands

For supported flash commands, the IP automatically generates the flash command sequence to perform flash erase, write, or read transaction as described in the Supported Commands Flash Operation section.

Table 5.28 shows the Packet Header 0 Register bit field definitions when sup_flash_cmd is set to 1'b1. If FIFO is enabled, this register is read-only access.

Table 5.28. Packet Header 0 Register – Supported Flash Commands

Field	Name	Description	Access	Default
[31:16]	xfer_len_bytes	Transfer length in bytes. The total number of bytes goes up to 65,535 bytes (64 KB). 16'h1 – 1 byte. 16'h2 – 2 bytes. … 16'hFFFF – 65,535 bytes. 16'h0 – 64 K bytes. For program operations, set xfer_len_bytes to multiples of 4 or 8. If the number of bytes in the transfer is not multiples of 4 or 8, use the generic command packet instead.	RW	0x0
[15:8]	num_wait_state	Indicates the number of wait cycles or dummy cycles after the flash address is transmitted during read commands. 0 – No wait cycle. 1 – One wait cycle. … 255 – 255 wait cycles.	RW	0x0
[7]	multiple_flash_target	0 – Single flash target. The tgt_cs register value is ignored. 1 – Multiple flash targets. The tgt_cs register value is used for target chip select.	RW	0x0
[6:2]	flash_command_code	The values 0 to 31 map to specific vendor flash command codes. Table 5.29 lists the supported flash commands.	RW	0x0
[1]	with_payload	0 – Payload will not be present in this packet. 1 – Payload is present in this packet. The total payload should be specified in the xfer_len_bytes register field.	RW	0x0
[0]	sup_flash_cmd	0 – Generic command packet of the IP will be used to perform SPI transaction. 1 – Supported flash command sequence of the IP will be used.	RW	0x0

Table 5.29. Supported Flash Commands

flash_command_code	Flash Command
5'h00	Read Status Register-1
5'h01	Read Status Register-2
5'h02	Read Status Register-3
5'h03	Read Configuration Register
5'h04	Read ID
5'h05	Read Electronic ID
5'h06	Multiple I/O Read ID
5'h07	Read Manufacturer and Device ID
5'h08	Read Manufacturer and Device ID Dual I/O
5'h09	Read Manufacturer and Device ID Quad I/O
5'h0A	Read Data
5'h0B	Fast Read
5'h0C	Dual Output Fast Read
5'h0D	Dual Input/Output Fast Read
5'h0E	Quad Output Fast Read
5'h0F	Quad Input/Output Fast Read
5'h10	Block Erase Type 1
5'h11	Block Erase Type 2
5'h12	Block Erase Type 3
5'h13	Chip Erase
5'h14	Write Enable
5'h15	Write Disable
5'h16	Write Status/Configuration Register
5'h17	Page Program
5'h18	Dual Input Fast Program
5'h19	Extended Dual Input Fast Program
5'h1A	Quad Input Fast Program
5'h1B	Extended Quad Input Fast Program
5'h1C	Enter 4-Byte Address Mode
5'h1D	Exit 4-Byte Address Mode
5'h1E	Enter Quad Input/Output Mode
5'h1F	Reset Quad Input/Output Mode

5.4.3.2. Packet Header 0 Register for Generic SPI Transactions

For generic SPI transactions, you can manually create the flash sequence necessary to perform a flash operation.

Table 5.30 shows the Packet Header 0 Register bit field definitions when sup_flash_cmd is set to 1'b0. If FIFO is enabled, this register is read-only access.

Table 5.30. Packet Header 0 Register – Generic SPI Transactions

Field	Name	Description	Access	Default
[31:16]	xfer_len_bytes	Transfer length in bytes. The total number of bytes goes up to 65535 bytes (64 KB). This register must be set to a non-zero value.	RW	0x1
[15:13]	num_wait_sck	Indicates the number of wait cycles or dummy cycles after the flash address is transmitted during read commands. 0 – No wait cycle. 1 – One wait cycle. … 7 – Seven wait cycles. Additional wait cycles (multiples of 8 SCK) can be generated by sending additional dummy write transactions after sending the address and before the actual read transaction using generic packet data register.	RW	0x0
[12:8]	tgt_cs	Chip select 0 to 31 targets can be selected.	RW	0x0
[7]	multiple_target	0 – Single flash target. The tgt_cs register value is ignored. 1 – Multiple flash targets. The tgt_cs register value is used for target chip select.	RW	0x0
[6]	frm_end	Frame end 1'b0 – Stop/Pause SCK if there are no further transactions. Chip select is not de-asserted after the last byte transfer. 1'b1 – Stop/Pause SCK and de-assert chip select after the last byte transfer.	RW	0x0
[5]	frm_start	Frame start 1'b0 – Indicates that the current transfer is not the start of the frame. This packet continues the previous transaction from asserted chip select held from previous transaction. 1'b1 – Indicates that the current transfer is the start of the frame. The target chip select transitions from a de-asserted state to an asserted state.	RW	0x0
[4]	data_rate	1'b0 – Single data rate. 1'b1 – Invalid value. Only single data rate is supported for this IP release.	RW	0x0
[3:2]	lane_width	2'b00 – x1 (Standard SPI) 2'b01 – x2 (Dual SPI) 2'b10 – x4 (Quad SPI) 2'b11 – Reserved	RW	0x0
[1]	cmd_type	1'b0 – SPI read transaction 1'b1 – SPI write transaction (payload may or may not be present)	RW	0x0

Field	Name	Description	Access	Default
[0]	sup_flash_cmd	Supported flash command 0 – The flash command to be sent is not supported by the IP. 1 – The flash command to be sent is supported by the IP.	RW	0x0

5.4.4. Packet Header 1 Register

This register can be used only for supported flash commands. If FIFO is enabled, this register is read-only access.

Table 5.31. Packet Header 1 Register

Field	Name	Description	Access	Default
[31:16]	reserved	Reserved	RO	0x0
[15:13]	flash_addr_width	Flash address width to be used in the transaction. 3'b000 – No address 3'b001 – 16-bit address 3'b010 – 24-bit address 3'b011 – 32-bit address 3'b100 – 40-bit address 3'b101 – 48-bit address 3'b110 – 56-bit address 3'b111 – 64-bit address	RW	0x0
[12:8]	tgt_cs	Target chip select. This should be set when multiple_flash_target is set to 1. Supports up to 32 targets (Target 0, Target 1, …, Target 31).	RW	0x0
[7:6]	reserved	Reserved	RO	0x0
[5:4]	dat_lane_width	Data can be sent to target flash memory in different lane widths. 2'b00 – Use x1 lane. 2'b01 – Use x2 lane. 2'b10 – Use x4 lane.	RW	0x0
[3:2]	addr_lane_width	Address can be sent to target flash memory in different lane widths. 2'b00 – Use x1 lane. 2'b01 – Use x2 lane. 2'b10 – Use x4 lane.	RW	0x0
[1:0]	cmd_lane_width	Command can be sent to target flash memory in different lane widths. 2'b00 – Use x1 lane. 2'b01 – Use x2 lane. 2'b10 – Use x4 lane.	RW	0x0

5.4.5. Packet Header 2 Register

This register should be set for supported flash commands and when flash_addr_width is greater than 3'b000 but less than 3'b100. The address starts at the most significant bit. For example, if flash_addr_width is 3'b010 for a 24-bit address, then bits [31:8] should contain the flash_address value. If FIFO is enabled, this register is read-only access.

Table 5.32. Packet Header 2 Register

Field	Name	Description	Access	Default
[31:24]	flash_address	Endianness mode applies to this register. For supported commands IP transaction when *Enable Flash Address Mapping* == *Unchecked*, little endian mode, and a 32-bit flash address, packet header 2 bits [7:0] should contain flash_address [31:24]. For little endian mode and a 24-bit flash address, packet header 2 bits [15:8] should contain the flash_address [23:16]. For big endian mode, packet header 2 bits [31:24] are treated collectively as the most significant byte and therefore, should contain the flash_address most significant byte that will be sent first up to the last valid flash_address based on flash_addr_width.	RW	0x00
[23:16]	flash_address		RW	0x00
[15:8]	flash_address		RW	0x00
[7:0]	flash_address		RW	0x00

5.4.6. Packet Header 3 Register

This register should be set for supported flash commands and when flash_addr_width is greater than 3'b011 and flash address is greater than 32 bits. If FIFO is enabled, this register is read-only access.

Table 5.33. Packet Header 3 Register

Field	Name	Description	Access	Default
[31:24]	flash_address	Endianness mode applies to this register. This packet header works with packet header 2 and starts with the most significant bit. For example, if flash_addr_width is 3'b101 for a 48-bit address, packet header 2 contains flash_address [47:16] while packet header 3 [31:16] contains flash_address [15:0].	RW	0x00
[23:16]	flash_address		RW	0x00
[15:8]	flash_address		RW	0x00
[7:0]	flash_address		RW	0x00

5.4.7. Packet Data 0 Register

This should be present only for commands with payload and there is no FIFO enabled. If FIFO is enabled, this register is read-only access.

Table 5.34. Packet Data 0 Register

Field	Name	Description	Access	Default
[31:0]	pkt_data_0	The payload starts at the lowest significant data bit. You should keep track of the status of this register to know when to write or read new data. Endianness mode applies to this register.	RW	0x00

5.4.8. Packet Data 1 Register

This should be present only for commands with payload and there is no FIFO enabled. If FIFO is enabled, this register is read-only access.

Table 5.35. Packet Data 1 Register

Field	Name	Description	Access	Default
[31:0]	pkt_data_1	The payload starts at the lowest significant data bit. You should keep track of the status of this register to know when to write new data. Endianness mode applies to this register.	RW	0x00

5.4.9. Start Transaction Register

When all the configuration and control registers are already set, set this register to start flash operations. This register also enables the XiP support of the IP.

Table 5.36. Start Transaction Register

Field	Name	Description	Access	Default
[31:2]	reserved	Reserved	RO	0x0
[1]	xip_enable	eXecute-in-Place (XiP) Enable 1'b0 – Flash device XiP mode of operation will not be used. 1'b1 – Flash device XiP mode of operation will be used on the transaction.	RW	0x0
[0]	start_trans	Used to start the transfer from controller to the targeted flash. 0 – Idle or no operation. 1 – Start transfer.	RW	0x0

5.5. Operation Details/Programming Flow

The QSPI Flash Controller IP usage is dependent on the use of either the transmit FIFO or receive FIFO. If FIFOs are enabled, all data in the packet header and data format are written to the FIFOs. Otherwise, all user packet registers should be set before starting a transaction. Read data can either be stored in a receive FIFO or read through packet data 0 register.

5.5.1. Flash Write/Erase Supported Commands with FIFO Enabled

For FIFO enabled configurations, all data necessary to perform an operation should be written first to the FIFO before starting the transaction.

Below is an example on how to perform a flash operation which requires a command to be sent to the SPI interface. To perform the "Enter Quad Input/Output Mode" operation, send all the transactions listed in Table 5.37 to the interface. Then, perform the polling mode as described in the Polling Mode for Transaction Status section.

Other supported commands, such as Enter/Exit 4-Bytes Mode, have the same sequence below except that the flash_command_code bit fields should be set according to the equivalent 5-bit value listed in Table 5.29. For operations where the target flash is set to operate on dual or quad SPI mode, cmd_lane_width should be set according to the I/O lanes requirement of the target flash.

Table 5.37. Programming Flow: Enter Quad Input/Output Mode, Without Payload, Standard SPI

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x200	0x0000_0079	[31:16] xfer_len_bytes = 16'h0
		[15:8] num_wait_state = 8'h0
		[7] multiple_flash_target = 1'h0
		[6:2] flash_command_code = 5'h1E (Enter Quad Input/Output Mode)
		[1] with_payload = 1'h0
		[0] sup_flash_cmd = 1'h1 (Supported Command)
	0x0000_0000	[31:16] reserved bits
		[15:13] flash_addr_width = 3'h0
		[12:8] tgt_cs = 5'h0
		[7:6] reserved bits
		[5:4] dat_lane_width = 2'h0
		[3:2] addr_lane_width = 2'h0
		[1:0] cmd_lane_width = 2'h0 (x1 lane)
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

Below is an example on how to perform a flash operation which requires a command and address to be sent to the SPI interface. To perform “Block Erase Type 1” operation, send all the transactions listed in Table 5.38 to the interface. Then, perform the polling mode as described in the Polling Mode for Transaction Status section.

Other supported commands, such as Block Erase Type 2 and 3, have the same sequence below except that the flash_command_code bit fields should be set according to the equivalent 5-bit value listed in Table 5.29. For operations where the target flash is set to operate on dual or quad SPI mode, cmd_lane_width and addr_lane_width should be set according to the I/O lanes requirement of the target flash.

Table 5.38. Programming Flow: Block Erase Type 1, Without Payload, Big Endian, Standard SPI

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x200	0x0000_0041	[31:16] xfer_len_bytes = 16’h0
		[15:8] num_wait_state = 8’h0
		[7] multiple_flash_target = 1’h0
		[6:2] flash_command_code = 5’h10 (Block Erase Type 1)
		[1] with_payload = 1’h0
		[0] sup_flash_cmd = 1’h1 (Supported Command)
	0x0000_4000	[31:16] reserved bits
		[15:13] flash_addr_width = 3’h2 (24-bit address mode)
		[12:8] tgt_cs = 5’h0
		[7:6] reserved bits
		[5:4] dat_lane_width = 2’h0
		[3:2] addr_lane_width = 2’h0 (x1 lane)
		[1:0] cmd_lane_width = 2’h0 (x1 lane)
	0x0010_0000	[31:8] flash_address = 24’h00_1000
		[7:0] unused bits
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

Below is an example on how to perform a flash operation which requires a command and data to be sent to the SPI interface. To perform the “Write Status/Configuration Register” operation, send all the transactions listed in Table 5.39 to the interface. Then, perform the polling mode as described in the Polling Mode for Transaction Status section. For operations where the target flash is set to operate on dual or quad SPI mode, cmd_lane_width and data_lane_width should be set according to the I/O lanes requirement of the target flash.

Table 5.39. Programming Flow: Write Status/Configuration Register, With Payload, Big Endian, Standard SPI

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x200	0x0001_005B	[31:16] xfer_len_bytes = 16’h1 (1 byte)
		[15:8] num_wait_state = 8’h0
		[7] multiple_flash_target = 1’h0
		[6:2] flash_command_code = 5’h16 (Write Status/Configuration Register)
		[1] with_payload = 1’h1 (with write data)
		[0] sup_flash_cmd = 1’h1 (Supported Command)
	0x0000_0000	[31:16] reserved bits
		[15:13] flash_addr_width = 3’h0
		[12:8] tgt_cs = 5’h0
		[7:6] reserved bits
		[5:4] dat_lane_width = 2’h0 (x1 lane)
		[3:2] addr_lane_width = 2’h0
		[1:0] cmd_lane_width = 2’h0 (x1 lane)
	0x4000_0000	Payload DW1
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

Below are examples on how to perform a flash operation which requires a command, address, and data to be sent to the SPI interface. To perform “Page Program” and “Extended Quad Input Fast Program” operations, send all the transactions listed in Table 5.40 and Table 5.41 to the interface, respectively. After the transaction is started, follow the procedures in the Polling Mode after a Write Transaction is started for FIFO Enabled Configuration section.

All other program supported commands have the same sequence below except that the flash_command_code bit fields should be set according to the equivalent 5-bit value listed in Table 5.29 and the cmd_lane_width, addr_lane_width, and dat_lane_width bit fields should be set according to the I/O lanes requirement of the target flash.

Flash devices may need additional commands or sequence to set it to a different SPI mode to perform other program commands. These necessary steps should be done prior to performing the flash operation.

Table 5.40. Programming Flow: 256-bytes Page Program, FIFO Enabled, Big Endian, Standard SPI

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x200	0x0100_005F	[31:16] xfer_len_bytes = 16’h100 (256-bytes)
		[15:8] num_wait_state = 8’h0
		[7] multiple_flash_target = 1’h0
		[6:2] flash_command_code = 5’h17 (Page Program)
		[1] with_payload = 1’h1 (with write data)
		[0] sup_flash_cmd = 1’h1 (Supported Command)
	0x0000_4000	[31:16] reserved bits
		[15:13] flash_addr_width = 3’h2 (24-bit address mode)
		[12:8] tgt_cs = 5’h0
		[7:6] reserved bits
		[5:4] dat_lane_width = 2’h0 (x1 lane)
		[3:2] addr_lane_width = 2’h0 (x1 lane)
		[1:0] cmd_lane_width = 2’h0 (x1 lane)
	0x0010_0000	[31:8] flash_address = 24’h00_1000
		[7:0] unused bits
	0x1234_5678	Payload DW1
	0x9ABC_DEF0	Payload DW2
	…	Payload DWx
	0xDEAD_BEEF	Payload DW63
	0xA5A5_C3C3	Payload DW64
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

Table 5.41. Programming Flow: 256-bytes Extended Quad Input Fast Program, FIFO Enabled, Little Endian

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x200	0x0100_006F	[31:16] xfer_len_bytes = 16’h100 (256-bytes)
		[15:8] num_wait_state = 8’h0
		[7] multiple_flash_target = 1’h0
		[6:2] flash_command_code = 5’h1B (Page Program)
		[1] with_payload = 1’h1 (with write data)
		[0] sup_flash_cmd = 1’h1 (Supported Command)
	0x0000_4028	[31:16] reserved bits
		[15:13] flash_addr_width = 3’h2 (24-bit address mode)
		[12:8] tgt_cs = 5’h0
		[7:6] reserved bits

Interface Address	Write Data	Description/Comment
		[5:4] dat_lane_width = 2'h2 (x4 lane)
		[3:2] addr_lane_width = 2'h2 (x4 lane)
		[1:0] cmd_lane_width = 2'h0 (x1 lane)
	0x0010_0000	[31:8] flash_address = 24'h00_1000
		[7:0] unused bits
	0x7856_3412	Payload DW1
	0xF0DE_BC9A	Payload DW2
	…	Payload DWx
	0xEFBE_ADDE	Payload DW63
	0xC3C3_A5A5	Payload DW64
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

5.5.2. Flash Read Supported Commands with FIFO Enabled

Below are examples on how to perform a flash read operation which requires only a command to be sent to the SPI interface to have data read from flash. To perform “Read Status Register-1” and “Quad Input/Output Fast Read” operations, send all the transactions listed in Table 5.42 and Table 5.43 to the interface, respectively. To continue the operation, follow the procedure in the Polling Mode after a Read Transaction is started for FIFO Enabled Configuration section.

All other read status and ID supported commands have the same sequence below except that the flash_command_code bit fields should be set according to the equivalent 5-bit value listed in Table 5.29 and xfer_len_bytes bit fields should align to the expected read data from flash.

Table 5.42. Programming Flow: Read Status Register, FIFO Enabled, Standard SPI

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x200	0x0001_0001	[31:16] xfer_len_bytes = 16’h1
		[15:8] num_wait_state = 8’h0
		[7] multiple_flash_target = 1’h0
		[6:2] flash_command_code = 5’h00 (Read Status Register-1)
		[1] with_payload = 1’h0
		[0] sup_flash_cmd = 1’h1 (Supported Command)
	0x0000_0000	[31:16] reserved bits
		[15:13] flash_addr_width = 3’h0
		[12:8] tgt_cs = 5’h0
		[7:6] reserved bits
		[5:4] dat_lane_width = 2’h0
		[3:2] addr_lane_width = 2’h0
		[1:0] cmd_lane_width = 2’h0 (x1 lane)
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

All other read data supported commands have the same sequence below except that the flash_command_code bit fields should be set according to the equivalent 5-bit value listed in Table 5.29 and cmd_lane_width, addr_lane_width, dat_lane_width, and num_wait_state bit fields should follow the target flash device requirements on these I/O lanes and dummy clock cycles.

Flash devices may need additional commands or sequence to set it to a different SPI mode and dummy clock cycle configuration to perform other read commands. These necessary steps should be done prior to performing the flash operation.

Table 5.43. Programming Flow: 256-bytes Quad Input/Output Fast Read, FIFO Enabled, Big Endian, Extended SPI

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x200	0x0100_063D	[31:16] xfer_len_bytes = 16’h100 (256-bytes)
		[15:8] num_wait_state = 8’h6 (6 dummy clock cycles)
		[7] multiple_flash_target = 1’h0
		[6:2] flash_command_code = 5’h0F (Quad Input/Output Fast Read)
		[1] with_payload = 1’h0
		[0] sup_flash_cmd = 1’h1 (Supported Command)
	0x0000_4028	[31:16] reserved bits
		[15:13] flash_addr_width = 3’h2 (24-bit address mode)

Interface Address	Write Data	Description/Comment
		[12:8] tgt_cs = 5'h0
		[7:6] reserved bits
		[5:4] dat_lane_width = 2'h2 (x4 lane)
		[3:2] addr_lane_width = 2'h2 (x4 lane)
		[1:0] cmd_lane_width = 2'h0 (x1 lane)
	0x0010_0000	[31:8] flash_address = 24'h000_1000
		[7:0] *unused bits*
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

5.5.3. Flash Supported Commands using Packet Header and Data Registers

Flash erase, write, or read operations with FIFO disabled configuration uses packet header and data registers. The same sample sequence and write data from the Flash Write/Erase Supported Commands with FIFO Enabled and Flash Read Supported Commands with FIFO Enabled sections apply. But instead of writing via the TX FIFO register address, interface write access uses the packet header and data register addresses. The first two dwords are the write data of packet header 0 and 1 registers. When address is present, depending on the flash_addr_width, these are the write data of packet header 2 and 3 registers. For write commands with payload, all the payloads are the write data of packet data registers. While for read commands with return data from flash, packet data 0 register is read instead of RX FIFO register address.

Table 5.44 shows a list of transactions to perform a page program. After starting the transaction, follow procedure in the Polling Mode after a Write Transaction is started for FIFO Disabled Configuration section to continue the writing of data to packet data registers.

Table 5.44. Programming Flow: 256-bytes Page Program, FIFO Disabled, Big Endian, Standard SPI

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x208	0x0100_005F	[31:16] xfer_len_bytes = 16'h100 (256-bytes)
		[15:8] num_wait_state = 8'h0
		[7] multiple_flash_target = 1'h0
		[6:2] flash_command_code = 5'h17 (Page Program)
		[1] with_payload = 1'h1 (with write data)
		[0] sup_flash_cmd = 1'h1 (Supported Command)
Register Map: Interface Base Address + 0x20C	0x0000_4000	[31:16] reserved bits
		[15:13] flash_addr_width = 3'h2 (24-bit address mode)
		[12:8] tgt_cs = 5'h0
		[7:6] reserved bits
		[5:4] dat_lane_width = 2'h0 (x1 lane)
		[3:2] addr_lane_width = 2'h0 (x1 lane)
		[1:0] cmd_lane_width = 2'h0 (x1 lane)
Register Map: Interface Base Address + 0x210	0x0010_0000	[31:8] flash_address = 24'h00_1000
		[7:0] unused bits
Register Map: Interface Base Address + 0x218	0x1234_5678	Payload DW1
Register Map: Interface Base Address + 0x21C	0x9ABC_DEF0	Payload DW2
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction
Register Map: Interface Base Address + 0x218 and Register Map: Interface Base Address + 0x21C	…	Payload DWx
Register Map: Interface Base Address + 0x218	0xDEAD_BEEF	Payload DW63
Register Map: Interface Base Address + 0x21C	0xA5A5_C3C3	Payload DW64

Table 5.45 shows a list of transactions to perform a read operation. After starting the transaction, follow the procedure in the Polling Mode after a Read Transaction is started for FIFO Disabled Configuration section to read the data from packet data registers.

Table 5.45. Programming Flow: 256-bytes Quad Input/Output Fast Read, FIFO Disabled, Big Endian, Extended SPI

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x208	0x0100_063D	[31:16] xfer_len_bytes = 16'h100 (256-bytes)
		[15:8] num_wait_state = 8'h6 (6 dummy clock cycles)
		[7] multiple_flash_target = 1'h0
		[6:2] flash_command_code = 5'h0F (Quad Input/Output Fast Read)
		[1] with_payload = 1'h0
		[0] sup_flash_cmd = 1'h1 (Supported Command)
Register Map: Interface Base Address + 0x20C	0x0000_4028	[31:16] reserved bits
		[15:13] flash_addr_width = 3'h2 (24-bit address mode)
		[12:8] tgt_cs = 5'h0
		[7:6] reserved bits
		[5:4] dat_lane_width = 2'h2 (x4 lane)
		[3:2] addr_lane_width = 2'h2 (x4 lane)
		[1:0] cmd_lane_width = 2'h0 (x1 lane)
Register Map: Interface Base Address + 0x210	0x0010_0000	[31:8] flash_address = 24'h000_1000
		[7:0] unused bits
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

5.5.4. Generic Flash Write Operation

This section provides examples on how to use the generic command packet of the IP to perform a generic flash write operation. The examples provided use the FIFO disabled configuration. For the FIFO enabled configuration, use the TX FIFO mapping register address instead of the packet header and data register addresses.

Below is an example on how to send a command opcode using generic command packet for the FIFO disabled configuration. To send a Write Enable command with 06h opcode, send all the transactions listed in Table 5.46 to the interface. Then, perform the polling mode as described in the Polling Mode for Transaction Status section.

Table 5.46. Programming Flow: Sending a Command, FIFO Disabled, Big Endian

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x208	0x0001_0062	[31:16] xfer_len_bytes = 16'h1 (1 byte)
		[15:13] num_wait_sck = 3'h0
		[12:8] tgt_cs = 5'h0
		[7] multiple_flash_target = 1'h0
		[6] frm_end = 1'h1 (end of the transaction)
		[5] frm_start = 1'h1 (start of the transaction)
		[4] data_rate = 1'h0 (single data rate)
		[3:2] lane_width = 2'h0 (x1 lane)
		[1] cmd_type = 1'h1 (write command)
		[0] sup_flash_cmd = 1'h0 (generic SPI transaction)
Register Map: Interface Base Address + 0x218	0x0600_0000	Payload
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

Through generic command packet, a flash operation with dwords to be written to a target flash device is also possible. First, set the write enable bit of the target device by sending the transactions listed in Table 5.46. Then, perform the transactions listed in Table 5.47 to program 256 bytes of data starting from the 24'h00_FF00 flash address. Since more than two dwords will be used, follow the procedure in the Polling Mode after a Write Transaction is started for FIFO Disabled Configuration section after starting the transaction.

Table 5.47. Programming Flow: Sending Command and Data, FIFO Disabled, Big Endian

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x208	0x0104_0062	[31:16] xfer_len_bytes = 16'h104 (1 byte for opcode, 3-bytes for address and 256-bytes to program on flash, a total of 260-bytes)
		[15:13] num_wait_sck = 3'h0
		[12:8] tgt_cs = 5'h0
		[7] multiple_flash_target = 1'h0
		[6] frm_end = 1'h1 (end of the transaction)
		[5] frm_start = 1'h1 (start of the transaction)
		[4] data_rate = 1'h0 (single data rate)
		[3:2] lane_width = 2'h0 (x1 lane)
		[1] cmd_type = 1'h1 (write command)
		[0] sup_flash_cmd = 1'h0 (generic SPI transaction)
Register Map: Interface Base Address + 0x218	0x0200_FF00	Payload DW1 (opcode and address)
Register Map: Interface Base Address + 0x21C	0x1234_5678	Payload DW2

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x218	0x9ABC_DEF0	Payload DW3
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction
Register Map: Interface Base Address + 0x218 and Register Map: Interface Base Address + 0x21C	…	Payload DWx
Register Map: Interface Base Address + 0x218	0xDEAD_BEEF	Payload DW64
Register Map: Interface Base Address + 0x21C	0xA5A5_C3C3	Payload DW65

5.5.5. Generic Flash Read Operation

This section provides an example on how to use the generic command packet of the IP to perform a generic flash read operation. The example provided uses the FIFO disabled configuration. For the FIFO enabled configuration, use the TX and RX FIFO mapping register addresses accordingly instead of the packet header and data register addresses.

Below is an example on how to read back 256 bytes of data after sending a read opcode 8'h03 starting from the 24'h00_FF00 flash address. The first transaction is a generic SPI write transaction to send the command opcode and address to the SPI bus while the second transaction is a generic SPI read transaction which reads the return data from the target device. After initiating the generic SPI write transaction, perform the polling mode as described in the Polling Mode for Transaction Status section. After initiating the generic SPI read transaction, follow the procedure in the Polling Mode after a Read Transaction is started for FIFO Disabled Configuration section.

Table 5.48. Programming Flow: Sending Command then Read Data, FIFO Disabled, Big Endian

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x208	0x0004_0022	[31:16] xfer_len_bytes = 16'h4 (1 byte for opcode, 3-bytes for address)
		[15:13] num_wait_sck = 3'h0
		[12:8] tgt_cs = 5'h0
		[7] multiple_flash_target = 1'h0
		[6] frm_end = 1'h0 (not yet the end of the transaction)
		[5] frm_start = 1'h1 (start of the transaction)
		[4] data_rate = 1'h0 (single data rate)
		[3:2] lane_width = 2'h0 (x1 lane)
		[1] cmd_type = 1'h1 (write transaction)
		[0] sup_flash_cmd = 1'h0 (generic SPI transaction)
Register Map: Interface Base Address + 0x218	0x0300_FF00	Payload DW1 (opcode and address)
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction
Polling Mode		
Register Map: Interface Base Address + 0x208	0x0100_0040	[31:16] xfer_len_bytes = 16'h100 (256-bytes of read data)
		[15:13] num_wait_sck = 3'h0
		[12:8] tgt_cs = 5'h0
		[7] multiple_flash_target = 1'h0
		[6] frm_end = 1'h1 (end of the transaction)
		[5] frm_start = 1'h0 (continuation of previous transaction)
		[4] data_rate = 1'h0 (single data rate)
		[3:2] lane_width = 2'h0 (x1 lane)
		[1] cmd_type = 1'h0 (read transaction)
		[0] sup_flash_cmd = 1'h0 (generic SPI transaction)
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

5.5.6. Non-XiP Access (Direct Write/Read Access) through Flash Memory Map

Below is a sample sequence on how to perform direct write or read within the flash memory map address range on the IP interface. In this example, a Macronix flash, MX25L51245G, is used as the target flash device. This device has a 512-Mbit size, which is equivalent to 64 MB or 65,536 KB.

1. Generate a QSPI protocol IP configuration with the following IP GUI settings:
   - Quad SPI supported protocol
   - *Enable Flash Address Mapping* selected
   - *Minimum Flash Address Alignment (KB)*: 65536 KB
   - *Flash Memory Map Size (KB)*: 65536 KB (same as the *Minimum Flash Address Alignment (KB)* since there is only one target flash device)
   - *Starting Flash Address (Hex)*: 0x0000_0000
   - *AXI/AHB-L Address Map (Hex)*: 0x0800_0000
     - Flash size is 65536 × 1024 × 8 for 512 Mb or 65536 × 1024 for 64 MB.
     - 2 × 512Mb or 2 × 64MB which is equivalent to 0x0800_0000.
     - The lower 64 MB will be enough for the IP's 1KB register.
     - Offset Address range for the whole flash: 0x0800_0000 – 0x0BFF_FFFF.
     - Offset Address 0x0800_000 maps to flash address 0x0000_0000 while offset address 0x0BFF_FFFF maps to flash address 0x03FF_FFFF.
   - *Write Access Command*: Extended Quad Input Fast Program
   - *Read Access Command*: Quad Input/Output Fast Read
   - *Flash Address Width*: 32
   - *Command Lane Width*: x1
   - *Address Lane Width*: x4
   - *Data Lane Width*: x4
   - *Dummy Clock Cycles*: 8

   Note: Transmit and receive FIFO can be optionally enabled to be used for supported command flash operations.
2. Set the target flash device to a 4-byte address mode. Flash devices may or may not have an equivalent command opcode for this operation. The example below assumes that the target flash device has an equivalent opcode. Some flash devices may have a different set of flash command opcodes for 3-byte and 4-byte address mode. If 4-byte address mode flash command opcode is used, set the opcodes through Flash Command Code 0-7 register addresses (refer to sections Flash Command Code 0 Register through Flash Command Code 7 Register).

   To perform the "Enter 4-byte Address Mode" operation, send all the transactions listed on Table 5.49 to the interface. Then, perform the polling mode as described in the Polling Mode for Transaction Status section.

Table 5.49. Programming Flow: Enter 4-byte Address Mode, FIFO Disabled, Big Endian

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x208	0x0000_0071	[31:16] xfer_len_bytes = 16'h0
		[15:8] num_wait_state = 8'h0
		[7] multiple_flash_target = 1'h0
		[6:2] flash_command_code = 5'h1C (Enter 4-byte Address Mode)
		[1] with_payload = 1'h0
		[0] sup_flash_cmd = 1'h1 (Supported Command)
Register Map: Interface Base Address + 0x20C	0x0000_0000	[31:16] reserved bits
		[15:13] flash_addr_width = 3'h0
		[12:8] tgt_cs = 5'h0

Interface Address	Write Data	Description/Comment
		[7:6] reserved bits
		[5:4] dat_lane_width = 2'h0
		[3:2] addr_lane_width = 2'h0
		[1:0] cmd_lane_width = 2'h0 (x1 lane)
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

3. Set the target device to quad mode. The example below assumes that there is a flash command which is Write Status Register with write data that should be done for this mode. The write data sets the specific bit of the register that enables the quad mode. Other flash devices and depending on the write or read flash command to be used, may have a specific command to enable this mode.

   To perform the "Write Status Register" operation, send all the transactions listed in Table 5.50 to the interface. Then, perform the polling mode as described in the Polling Mode for Transaction Status section.

Table 5.50. Programming Flow: Write Status Register, FIFO Disabled, Big Endian

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x208	0x0001_005B	[31:16] xfer_len_bytes = 16'h1 (1-byte data)
		[15:8] num_wait_state = 8'h0
		[7] multiple_flash_target = 1'h0
		[6:2] flash_command_code = 5'h16 (Write Status/Configuration Register)
		[1] with_payload = 1'h1
		[0] sup_flash_cmd = 1'h1 (Supported Command)
Register Map: Interface Base Address + 0x20C	0x0000_0000	[31:16] reserved bits
		[15:13] flash_addr_width = 3'h0
		[12:8] tgt_cs = 5'h0
		[7:6] reserved bits
		[5:4] dat_lane_width = 2'h0 (x1 lane)
		[3:2] addr_lane_width = 2'h0
		[1:0] cmd_lane_width = 2'h0 (x1 lane)
Register Map: Interface Base Address + 0x218	0x4000_0000	Payload DW1
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

4. Direct Write Access – 256-byte write using Quad Input/Output Fast Program supported command starting from 32'h0800_0000 address. Table 5.51 shows single write access to the main interface to perform the program operation.
   a. If a configuration change is required, set the configuration register 0. Bits [31:16] of this configuration register can be changed before starting direct access to update SPI transaction settings.
   b. Execute a write access on the main interface bus. For single write transfers, the output ready signal will stay de-asserted until the controller has sent the write data to the SPI bus and the flash is no longer busy. For burst write transfers, the output ready signal will stay de-asserted until the controller has sent the write data to the SPI bus. While the direct access is being performed on the main interface of the IP, AHB-L or AXI4, Polling Mode for Transaction Status can be performed on the AXI4L interface. Since it will take some time for the direct write access on the main interface to be processed and started by the IP, consider a minimum delay of ten SCLK cycles before starting the polling mode. This polling mode will consider the program time of the target device after the write data are transmitted on SPI bus to ensure that the flash device is already on idle state before requesting a new flash transaction.

Table 5.51. Programming Flow: Direct Write Access, Big Endian

Interface Address	Write Data	Description/Comment
Register Map: 0x0800_0000	0x1234_5678	DW1
Register Map: 0x0800_0004	0x9ABC_DEF0	DW2
…	…	…
Register Map: 0x0800_00F8	0xDEAD_BEEF	DW63
Register Map: 0x0800_00FC	0xA5A5_C3C3	DW64

5. Direct Read Access – 256-byte read using Quad Input/Output Fast Read supported command starting from 32'h0800_0000 address.
   a. If a configuration changed is required, set the configuration register 0. Bits [31:16] of this configuration register can be changed before starting direct access to update the SPI transaction settings.
   b. Execute a read access on the main interface bus. The interface output valid signal will assert whenever a 4-byte read data is available. While the direct access is being performed on the main interface of the IP, AHB-L or AXI4, Polling Mode for Transaction Status can be performed on the AXI4L interface. Since it will take some time for the direct read access on the main interface to be processed and started by the IP, consider a minimum delay of ten SCLK cycles before starting the polling mode. This polling mode will check the transaction status of the IP before requesting a new flash transaction.
6. For non-XiP and register-based transactions performed with either the AHB-L or AXI4 main interface and AXI4-L interface, respectively, transactions on one interface should be ensured done before requesting a new transaction to the other interface. For burst write transaction, after the transactions are done on the bus, consider the program time of the target device before issuing a new transaction to the SPI target.

5.5.7. XiP Access (Direct Read Access) through Flash Memory Map

The performance enhancement mode of the flash device is supported. Check the XiP activation/deactivation sequence of the target flash device and perform activation/deactivation using the generic SPI command packet to start and end the XiP-enabled transactions of the IP.

Below is a sample sequence on how to perform XiP access on a Macronix flash device, MX25L51245G, together with this IP. This flash device has a 512-Mbit size, which is equivalent to 64 MB or 65,536 KB. XiP activation of this flash device is performed in step 5 of this sample sequence.

1. Generate an IP configuration similar to as described in step 1 of the Non-XiP Access (Direct Write/Read Access) through Flash Memory Map section.
2. Enable the 4-byte address mode of the flash through the IP's supported command sequence. Refer to step 2 of the Non-XiP Access (Direct Write/Read Access) through Flash Memory Map section.
3. Set the Quad Enable (QE) bit of the status register to *1*. Refer to step 3 of the Non-XiP Access (Direct Write/Read Access) through Flash Memory Map section.
4. Enable Quad I/O mode of the flash through the IP's supported command sequence. Refer to Table 5.37 for the packet frame values. Instead of sending to the TX FIFO mapping address, send the packet frame values to packet header 0 and 1 registers accordingly.
5. Initiate a read transaction using Quad Input/Output Fast Read operation, 8'hEB opcode, along with the performance enhancement indicator to enable the XiP mode of the Macronix flash for succeeding reads.

   To send the command opcode, send all the transactions listed in Table 5.52 to the AXI4L interface. Then, perform the polling mode as described in the Polling Mode for Transaction Status section.

Table 5.52. Programming Flow: 1-Byte Quad Input/Output Fast Read, FIFO Disabled, Big Endian

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x208	0x0001_002A	[31:16] xfer_len_bytes = 16'h1 (1 byte)
		[15:13] num_wait_sck = 3'h0
		[12:8] tgt_cs = 5'h0
		[7] multiple_flash_target = 1'h0
		[6] frm_end = 1'h0 (not yet end of the transaction)
		[5] frm_start = 1'h1 (start of the transaction)
		[4] data_rate = 1'h0 (single data rate)
		[3:2] lane_width = 2'h2 (x2 lane)
		[1] cmd_type = 1'h1 (write command)
		[0] sup_flash_cmd = 1'h0 (generic SPI transaction)
Register Map: Interface Base Address + 0x218	0xEB00_0000	Payload (Flash command opcode)
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

To send the 4-byte address, send all the transactions listed in Table 5.53 to the AXI4L interface. Then, perform the polling mode as described in the Polling Mode for Transaction Status section.

Table 5.53. Programming Flow: 4-Byte Address, FIFO Disabled, Big Endian

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x208	0x0004_000A	[31:16] xfer_len_bytes = 16'h4 (4-byte address)
		[15:13] num_wait_sck = 3'h0
		[12:8] tgt_cs = 5'h0
		[7] multiple_flash_target = 1'h0
		[6] frm_end = 1'h0 (not yet end of the transaction)
		[5] frm_start = 1'h0 (continuation of previous transaction)
		[4] data_rate = 1'h0 (single data rate)
		[3:2] lane_width = 2'h2 (x4 lane)
		[1] cmd_type = 1'h1 (write command)
		[0] sup_flash_cmd = 1'h0 (generic SPI transaction)
Register Map: Interface Base Address + 0x218	0x0000_0000	Payload (Flash address equivalent to starting flash address value)
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

To send the XiP pattern and dummy, send all the transactions listed in Table 5.54 to the AXI4L interface. Then, perform the polling mode as described in the Polling Mode for Transaction Status section.

Table 5.54. Programming Flow: XiP Pattern and Dummy, FIFO Disabled, Big Endian

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x208	0x0001_800A	[31:16] xfer_len_bytes = 16'h1 (1-byte XiP pattern)
		[15:13] num_wait_sck = 3'h4 (4 dummy clock cycles for EBh command)
		[12:8] tgt_cs = 5'h0
		[7] multiple_flash_target = 1'h0
		[6] frm_end = 1'h0 (not yet end of the transaction)
		[5] frm_start = 1'h0 (continuation of previous transaction)
		[4] data_rate = 1'h0 (single data rate)
		[3:2] lane_width = 2'h2 (x4 lane)
		[1] cmd_type = 1'h1 (write command)
		[0] sup_flash_cmd = 1'h0 (generic SPI transaction)
Register Map: Interface Base Address + 0x218	0xA500_0000	Payload (XiP pattern of the target device)
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

To start the XiP read access, send all the transactions listed in Table 5.55 to the AXI4L interface. Then, perform the polling mode as described in the Polling Mode after a Read Transaction is started for FIFO Disabled Configuration section.

Table 5.55. Programming Flow: Read 4-bytes Data, FIFO Disabled, Big Endian

Interface Address	Write Data	Description/Comment
Register Map: Interface Base Address + 0x208	0x0004_0048	[31:16] xfer_len_bytes = 16'h4 (4-byte read data)
		[15:13] num_wait_sck = 3'h0
		[12:8] tgt_cs = 5'h0
		[7] multiple_flash_target = 1'h0
		[6] frm_end = 1'h1 (end of the transaction)
		[5] frm_start = 1'h0 (continuation of previous transaction)
		[4] data_rate = 1'h0 (single data rate)
		[3:2] lane_width = 2'h2 (x4 lane)
		[1] cmd_type = 1'h0 (read command)
		[0] sup_flash_cmd = 1'h0 (generic SPI transaction)
Register Map: Interface Base Address + 0x220	0x0000_0001	Start Transaction

6. If a configuration change is required, set the configuration register 0. Bit [31:16] of this configuration register can be changed before starting direct access to update SPI transaction settings.
7. Set the xip_enable bit of the Start Transaction Register 0x0000_0220 to *1*. This indicates that the next read access will be in XiP mode.
8. Execute a read access on the main interface bus to read a 256-byte data starting from the 0x0800_0000 flash address map. The interface output valid signal asserts whenever a 4-byte read data is available. While the direct access is being performed on the main interface of the IP, AHB-L or AXI4, Polling Mode for Transaction Status can be performed on the AXI4L interface. Since it will take some time for the direct read access on the main interface to be processed and started by the IP, consider a minimum delay of ten SCLK cycles before starting the polling mode. This polling mode will check the transaction status of the IP before requesting a new flash transaction. During the polling mode, the xip_enable bit of the Start Transaction Register 0x0000_0220 should stay at value 1.
9. XiP access can continue until deactivation of the XiP mode of the flash device is executed. To deactivate the XiP of the Macronix device used, send all the transactions in Table 5.52, Table 5.53, Table 5.54, and Table 5.55. In Table 5.54, set the packet data 0 to 32'h0000_0000 to exit the XiP mode of the flash. 8'h00 is the performance enhancement mode deactivation pattern used in this example.
10. Set the xip_enable bit of the Start Transaction Register 0x0000_0220 to *0*. This indicates that the next read access will be on non-XiP mode.

5.5.8. Polling Mode

This section describes the different polling modes to track a flash operation, write to packet data registers, and read from packet data registers.

5.5.8.1. Polling Mode for Transaction Status

After any transaction is initiated, the following are the steps to check and end the transaction:

1. Poll the trans_stat bit of the transaction status register described in the Transaction Status Register section to determine if the transaction has started.
2. Continue polling until an idle state is reached.
3. Once idle, write *0* to the start_trans bit of the start transaction register described in the Start Transaction Register section.

5.5.8.2. Polling Mode after a Write Transaction is started for FIFO Disabled Configuration

After setting the packet header and data registers and transaction is started, the following are the steps to continue an operation with payload greater than 2 dwords:

1. Perform a read on the transaction status register described in the Transaction Status Register section to check if the transaction has started.
2. For xfer_len_bytes greater than 8 (2 dwords), once the transaction has started, poll the interrupt status register bits [1:0] for wr_pkt_data_1_empty_int and wr_pkt_data_0_empty_int. Once a bit is read as high, clear the interrupt status bit, then write the new dword.
3. Continue performing step 2 until all xfer_len_bytes of data are written to the packet data register.
4. To check and end the transaction, perform the polling mode as described in the Polling Mode for Transaction Status section.

5.5.8.3. Polling Mode after a Read Transaction is started for FIFO Disabled Configuration

After setting the packet header and data registers and transaction is started, the following are the steps to continue a read operation:

1. Perform a read on the transaction status register to check if the transaction has started.
2. Once the read transaction has started, poll the interrupt status register bit [4] for rd_pkt_data_0_not_empty_int. Once the bit is read as high, clear the interrupt status bit, then perform a read on the packet data 0 register address.
3. Continue performing step 2 until all xfer_len_bytes of data are read back via the packet data 0 register.
4. To check and end the transaction, perform the polling mode as described in the Polling Mode for Transaction Status section.

5.5.8.4. Polling Mode after a Write Transaction is started for FIFO Enabled Configuration

Perform the steps in the Polling Mode for Transaction Status section to poll the status of the started write transaction and conclude this operation.

5.5.8.5. Polling Mode after a Read Transaction is started for FIFO Enabled Configuration

After setting the packet header and data value format in the RX FIFO Mapping registers and transaction is started, the following are the steps to continue a read operation:

1. Perform a read on the transaction status register to check if the transaction has already started.
2. Continue polling the transaction status register until an idle state is read.
3. Read the return data from flash through continuous reading of the RX FIFO Mapping register address until all target xfer_len_bytes of data are read.

Note: If reading the return data from flash is preferred while the flash read operation is on-going, poll the interrupt status register bit [4] for rx_fifo_empty_int to flag an available dword on the RX FIFO, then perform the read.

6. Example Design

A system design with the QSPI Flash Controller IP can be developed to test this peripheral IP with a RISC-V processor.

The QSPI Flash Controller example design allows you to compile, simulate, and test the QPSI Flash Controller IP on the following Lattice evaluation board:

- CertusPro-NX Evaluation Board

6.1. Example Design Supported Configuration

The following IP configuration was used during IP generation in the Propel project but some IP attributes with equivalent registers were also changed in the C code test sequence during validation.

Note: In the table below, ✓ refers to a checked option in the QSPI Flash Controller IP example design and × refers to an unchecked option or a non-applicable option in the QSPI Flash Controller IP.

Table 6.1. QSPI Flash Controller IP Configuration Used on Example Design

QSPI Flash Controller IP GUI Parameter	QSPI Flash Controller IP configuration
Interface Settings	
Main Interface	AHB-L
Enable AXI4-L Interface (for register access only)	✓
Interface Data Width	32
AXI4 ID Width	×
Data Endianness	Big Endian
IO Primitive	
Enable IO Buffer	✓
SPI Configuration and Transfer Settings	
Supported Protocol	Quad
Number of SPI Target	1
Clock Mode	3
First Transmitted Bit	MSB
Chip Select High Time (ns)	100
Clock Frequency Settings	
System Clock Frequency (MHz)	125
Internal Clock Frequency (MHz)	125
SPI Clock Frequency Divider	2
SPI Clock Frequency (MHz)	62.5
Transmit FIFO Configuration	
Enable Transmit FIFO	×
Receive FIFO Configuration	
Enable Receive FIFO	×
Flash Address Mapping	
Enable Flash Address Mapping	✓
Register Space Size (KB)	1
Register Map Base Address (Hex)	00000000
Minimum Flash Address Alignment (KB)	4
Starting Flash Address (Hex)	00000000
Flash Memory Map Size (4KB-2GB)	4
AXI/AHB-L Address Map (Hex)	10003000 [1]
Write Access Command	Page Program
Read Access Command	Fast Read
Flash Address Width	24

QSPI Flash Controller IP GUI Parameter	QSPI Flash Controller IP configuration
Command Lane Width	x1
Address Lane Width	x1
Data Lane Width	x1
Dummy clock cycles	8
eXecute-in-Place (XiP) Pattern	8'hA5
Supported Flash Commands	
Supported Flash Commands	Opcode default values

Note:

1. This IP attribute value must be the same as the main interface base address of the design's Address Space when using Propel.

6.2. Overview of the Example Design and Features

The example design discussed in this section is created using a RISC-V RX SoC Project template in the Lattice Propel Development Suite. The generated project includes the following components:

- Processor – RISC-V RX with tightly-coupled memory modules (TCMs), platform level interrupt controller (PLIC), core local interrupter (CLINT), and watchdog
- AXI4 system buses
- GPIO
- UART – Serial port
- Oscillator and PLL
- Glue logic
- System memory

Figure 6.1 shows how the QSPI Flash Controller IP is instantiated on the template. The AXI4-L interface is connected directly to the AXI4 system bus while the AHB-L interface is connected to the AXI4 system bus through a AXI4 to AHB-L bridge. The SPI interface ports are ported out as top-level ports. The input clock and reset are connected to the PLL output clock and system reset, respectively.

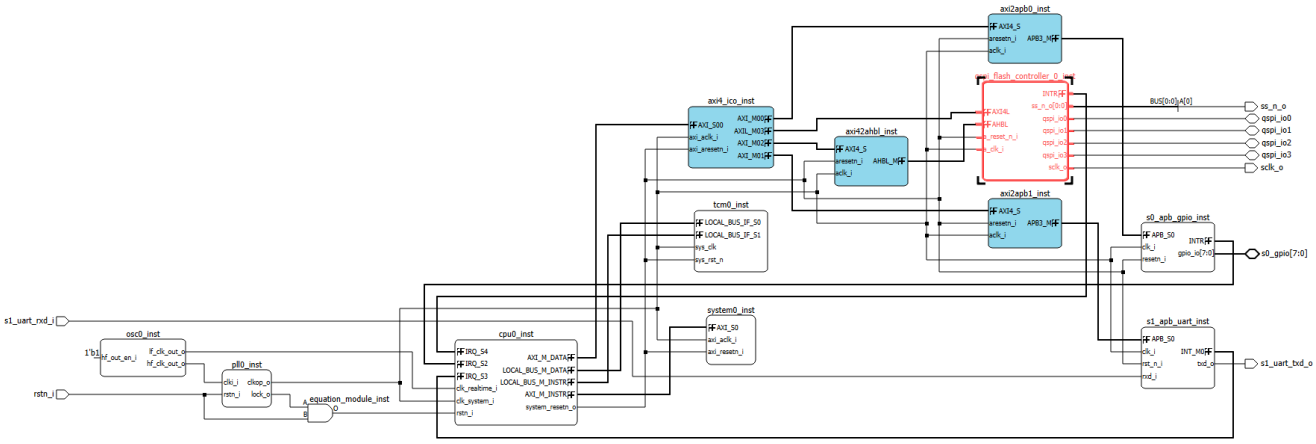

Figure 6.1. QSPI Flash Controller IP Example Design in SoC Project

6.3. Example Design Components

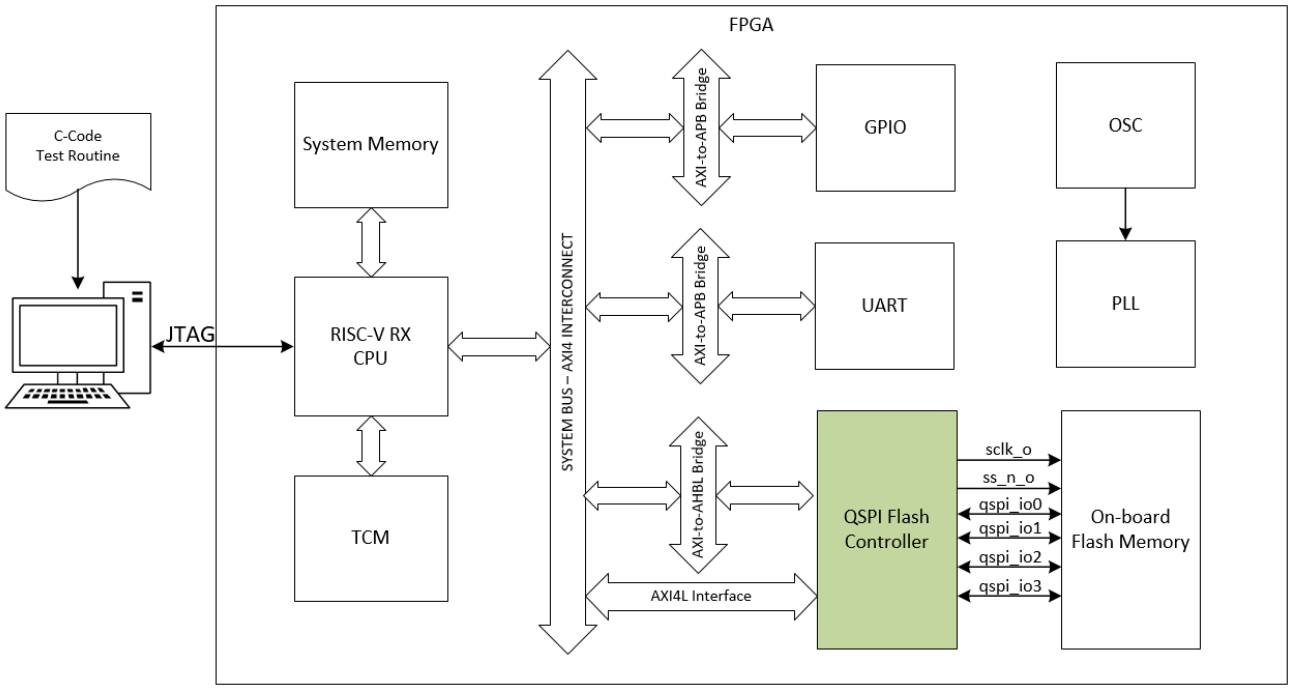

Figure 6.2. QSPI Flash Controller Example Design Block Diagram

The QSPI Flash Controller IP example design includes the following blocks:

- RISC-V RX CPU – Processes data and instructions.
- System memory and TCM – Stores the commands to be used for testing.
- Oscillator and PLL – Clock generator
- System bus – Manages transfers from the processor and memory to the connected IP modules.
- Interface bridge – Translates AXI4 to APB transactions for GPIO and UART; translates AXI4 to AHB-L transactions for QSPI Flash Controller.
- GPIO and UART – Serve as debug modules. UART can be used to display debug messages while GPIO output can be used to indicate successful load of bitstream on the FPGA board.
- QSPI Flash Controller IP – Instance of the IP connected to the flash device available on the FPGA board.
- Flash memory – SPI target. This is not an instance of any submodule. The IP's SPI interface ports connect to the on-board flash device through the physical design constraint (PDC) file.

6.4. Generating the Example Design

The Lattice Propel Builder and Propel SDK can be used to create an embedded system which consists of a system-on-chip (SoC) design with an embedded processor and system software. The following is a procedure for generating an example design for the QSPI Flash Controller IP. A workspace is generated on Propel SDK. The SoC project is then created in Propel Builder. For complete step-by-step instructions, refer to A Step-By-Step Approach to Lattice Propel (FPGA-AN-02052) or the Lattice Propel Builder User Guide (FPGA-UG-02212) and Lattice Propel SDK User Guide (FPGA-UG-02211).

To create an SoC project:

1. Launch Propel SDK. The **Lattice Propel Launcher** opens as shown on Figure 6.3. Browse to a directory for the **Workspace** field and click **Launch**.

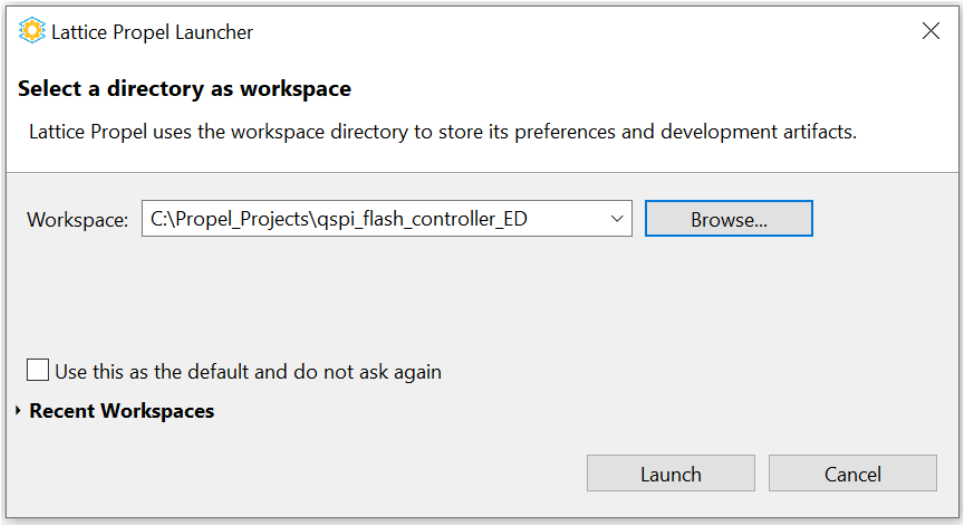

Figure 6.3. Lattice Propel Launcher

2. Create a new SoC design project by selecting **File > New > Lattice SoC Design Project**. The **Create SoC Project** window opens as shown on Figure 6.4. Enter a name for the SoC project in the **Project name** field. Customize the options under **Device Select** using drop-down menus or configure it through the **Board** option. Then, select the **Template Design**. For this example design, RISC-V RX processor, LFCPNX-100-9LFG672C board, and RISC-V RX SoC Project are used. Then, click **Finish**.

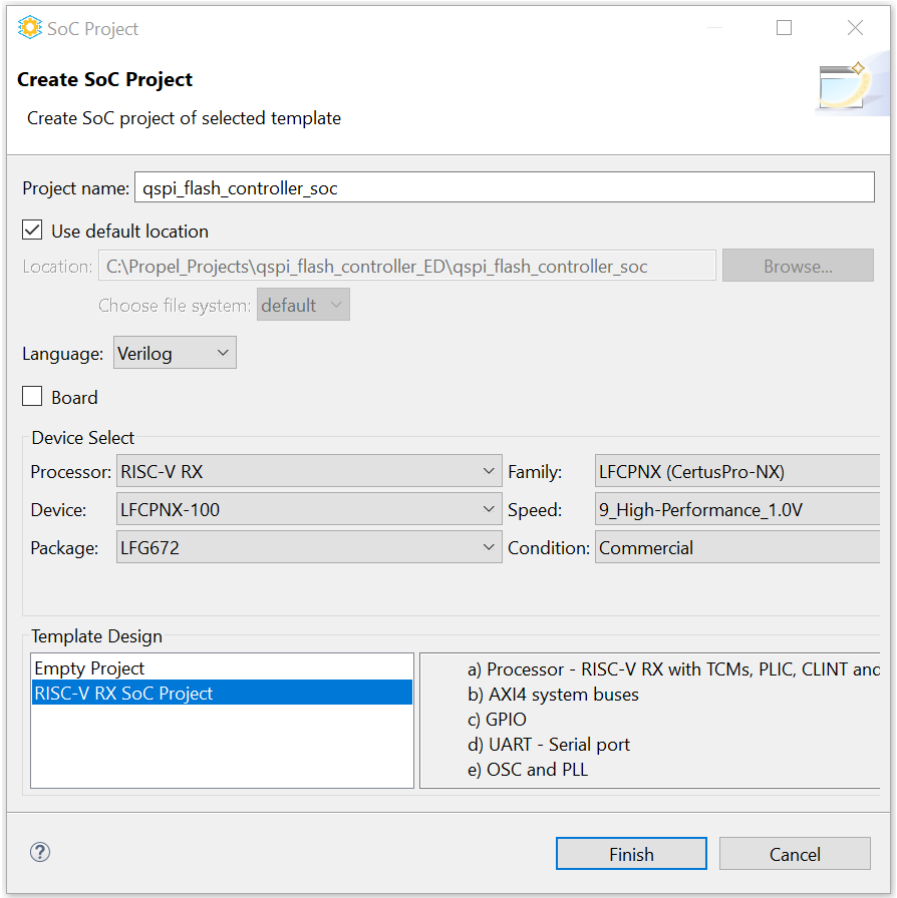

Figure 6.4. Create SoC Project

3. Select the generated project. Then, run Propel Builder by clicking the icon or select **LatticeTools > Open Design in Propel Builder.** The **Propel Builder** opens and loads the design template.
4. In the IP Catalog tab, instantiate the QSPI Flash Controller IP by generating the IP configuration as shown in Table 6.1.
5. After generating the IP, the **Define Instance** window opens as shown in Figure 6.5. Modify the instance name if needed. Then, click **OK**.

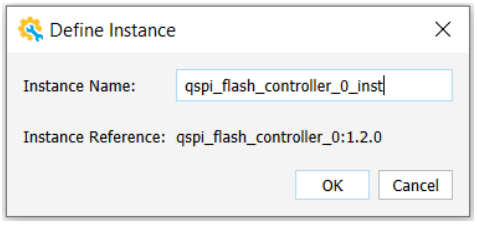

Figure 6.5. Define Instance

6. Connect the instantiated IP to the system (refer to Figure 6.1). Modify the system component settings from default as follows:
   - Oscillator HCLK frequency to 50 MHz.
   - PLL input clock and CLKOP frequency to 50 MHz and 125 MHz, respectively.
   - RISC-V RX CPU Number of User Interrupt Requests to 3.
   - Update the AXI4 interconnect and AXI4 to APB bridge instances. In case an error message is flagged during regeneration, select the instances, then delete and replace these instances with the latest IP versions available on the IP server.
   - AXI4 interconnect with four external subordinates composed of three AXI4 subordinates and one AXI4-L subordinate.
   - Add AXI4 to AHB-L bridge. From the AXI4 interconnect, use this bridge to connect to the QSPI Flash Controller IP AHB-L interface.
   - UART system clock frequency to 125 MHz.
7. View the project's memory space by selecting the **Address** tab as shown in Figure 6.6. Enable the **Lock** of address space settings so that addresses cannot be manually or automatically updated. Click the icon to perform DRC. Save the SoC project.

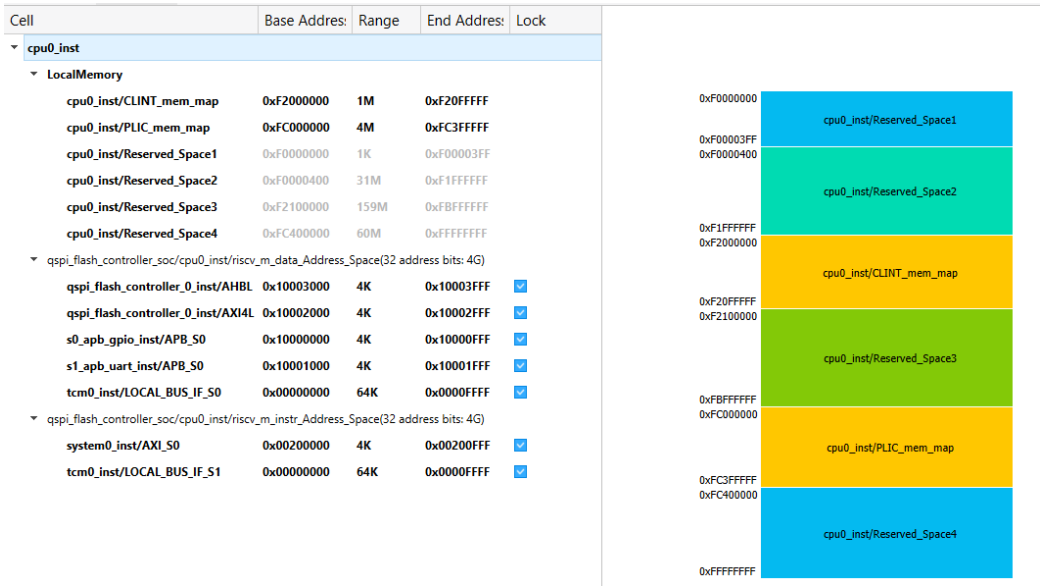

Figure 6.6. Address Tab

To create a Radiant project:

1. Click the icon or select **Design > Run Radiant** to launch the Lattice Radiant Software.
2. Create the IP constraint file. On the pdc file, connect the IP's SPI ports to the on-board flash pins. Set sysCONFIG MASTER_SPI_PORT to DISABLE to use the on-board flash as the target device.
3. Generate the programming file.

To create a C/C++ project:

1. In the Lattice Propel software, build your SoC project to generate the system environment needed for the embedded C/C++ project by selecting the generated SoC project followed by **Project > Build Project**.
2. Check the build result from the **Console** view as shown in Figure 6.7.

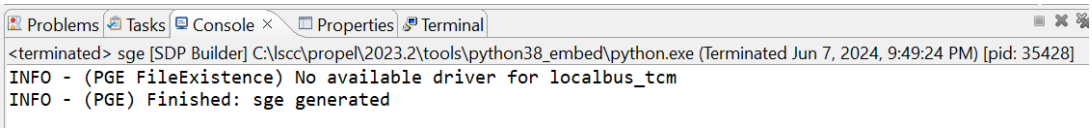

Figure 6.7. Build SoC Project Result

3. Generate a new Lattice C/C++ project by selecting **File > New > Lattice C/C++ Project**. The **C/C++ Project Load System and BSP** window opens as shown in Figure 6.8. Update the **Project name** field. Then, click **Next** followed by **Finish**. For this example design, the FreeRTOS Project template is used.

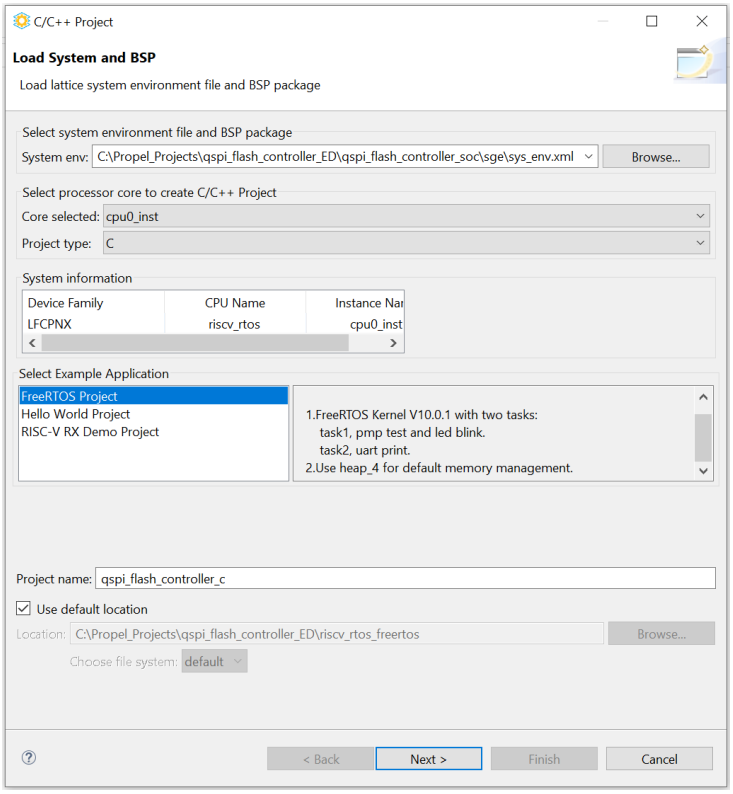

Figure 6.8. C/C++ Project Load System and BSP

4. Build the project by selecting the C/C++ project followed by **Project > Build Project**.
5. Check the build result from **Console** view as shown in Figure 6.9.

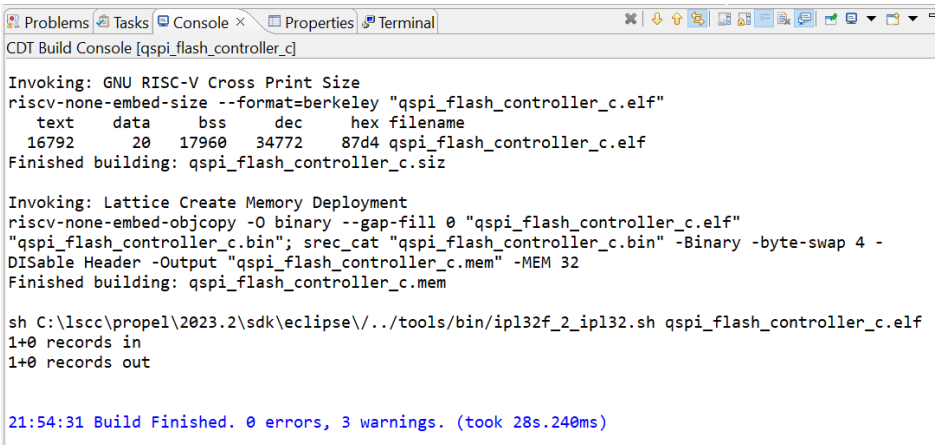

Figure 6.9. Build C/C++ Project Result

6. This environment is now ready for running on the FPGA board. Refer to the On-Chip Debugging section under Propel SDK Flow in A Step-By-Step Approach to Lattice Propel (FPGA-AN-02052) for details on how to run the design on the FPGA board.

6.5. Simulating the Example Design

The Lattice Propel Builder software also has a verification project mode which can be used to generate a simulation environment. The following is a procedure for generating a verification project for the QSPI Flash Controller IP:

1. From the SoC Project, perform the pre-simulation requirements enumerated in the Verification Project Flow section in A Step-By-Step Approach to Lattice Propel (FPGA-AN-02052).
2. Click  to switch to verification project.
3. Reload dut_inst by double clicking on it. The **Reload sbx** window opens as shown in Figure 6.10.

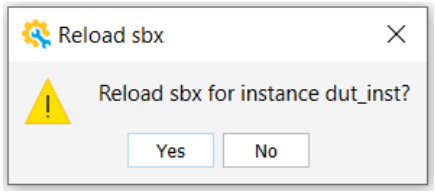

Figure 6.10. Reload sbx

4. Click **Yes**. The verification project schematic is generated as shown in Figure 6.11.

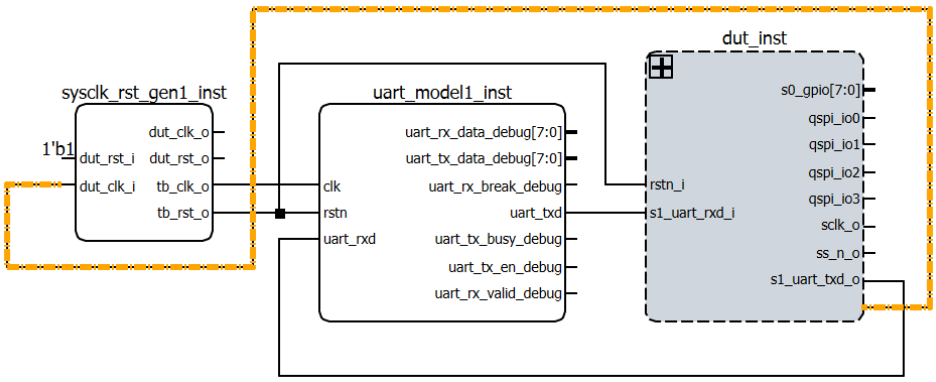

Figure 6.11. Verification Project Schematic

5. Click the **Generate** icon to generate the simulation environment. The testbench file including scripts for the chosen simulator, file lists, and other files are generated in the verification folder of the SoC project.
6. To see the full IP simulation behavior for this example design, connect a flash simulation model to the IP instance in the generated testbench file as shown in Figure 6.12 and add the flash simulation model in the generated file list.

qspi_flash_controller_soc_v.sv

```
85    /* This section is reserved for user to create stimulus using BFM */
86    /* APIs.                                                          */
87    /*----------------------------------------------------------------*/
88      wire DQ3;
89      assign DQ3 = 1'hZ;
90
91      flash_model flash_model_inst (
92          .S              (ss_n_o),
93          .C_             (sclk_o),
94          .RESET_DQ3      (qspi_io3),
95          .DQ0            (qspi_io0),
96          .DQ1            (qspi_io1),
97          .Vcc            ('d1800),
98          .Vpp_W_DQ2      (qspi_io2)
99      );
100
101   initial begin
107
108     /*----------------------Instantiation Blocks----------------------*/
109     /* This section cover the instantiation of DUT, VIPs, Simulation  */
110     /* Models, and other IP/Components.                               */
111     /*----------------------------------------------------------------*/
112     qspi_flash_controller_soc
113     dut_inst
114     (
115       .qspi_io0(qspi_io0),
116       .qspi_io1(qspi_io1),
117       .qspi_io2(qspi_io2),
118       .qspi_io3(qspi_io3),
119       .rstn_i(sysclk_rst_gen1_inst_tb_rst_o_net),
120       .s1_uart_rxd_i(uart_model1_inst_uart_txd_net),
121       .s1_uart_txd_o(uart_model1_inst_uart_rxd_net),
122       .sclk_o(sclk_o),
123       .ss_n_o(ss_n_o),
124       .s0_gpio()
125     );
```

Figure 6.12. Edited Testbench File

7. Click the **Launch Simulation** icon to run the simulation.
   Note: The simulation waveform shown in Figure 6.13 is a result where the generated main.c file in the C/C++ project is updated to perform an IP's operation.

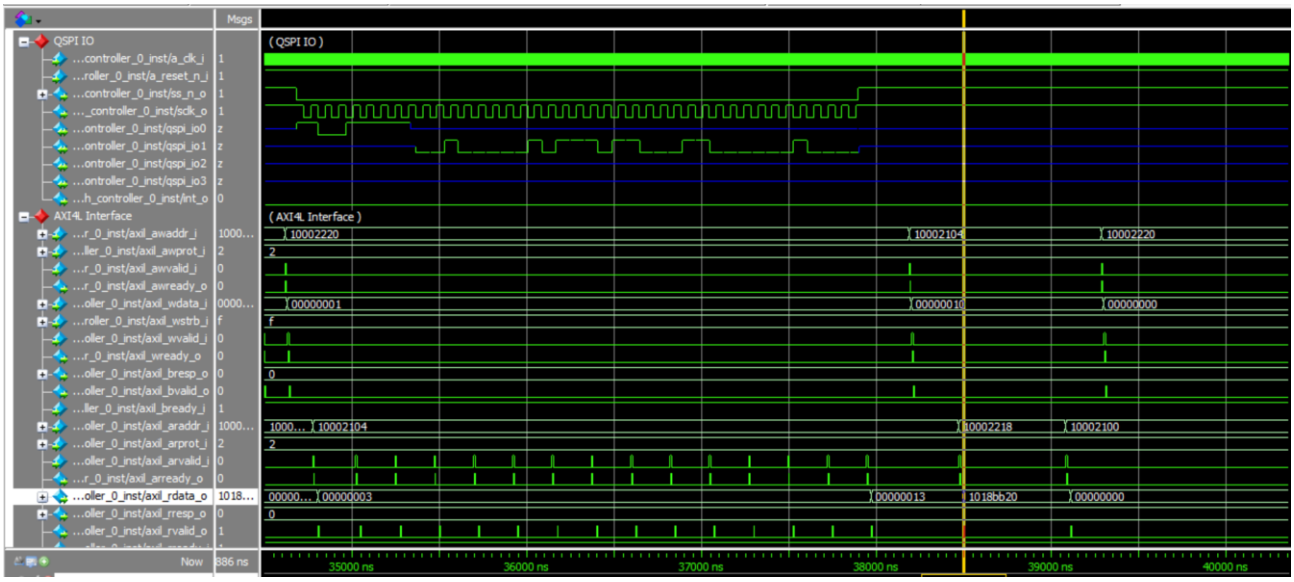

Figure 6.13. Simulation Waveform

6.6. Hardware Testing

The generated bitstream file from the procedure in the Generating the Example Design section was downloaded to the Certus-Pro NX Evaluation Board via the Radiant programmer. The Reveal inserter/analyzer was added to the Radiant project to check the output behavior of the IP.

7. Designing with the IP

This section provides information on how to generate the IP 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's architecture. The steps below describe how to generate the QSPI Flash Controller IP in the Lattice Radiant software.

To generate the QSPI Flash Controller IP:

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

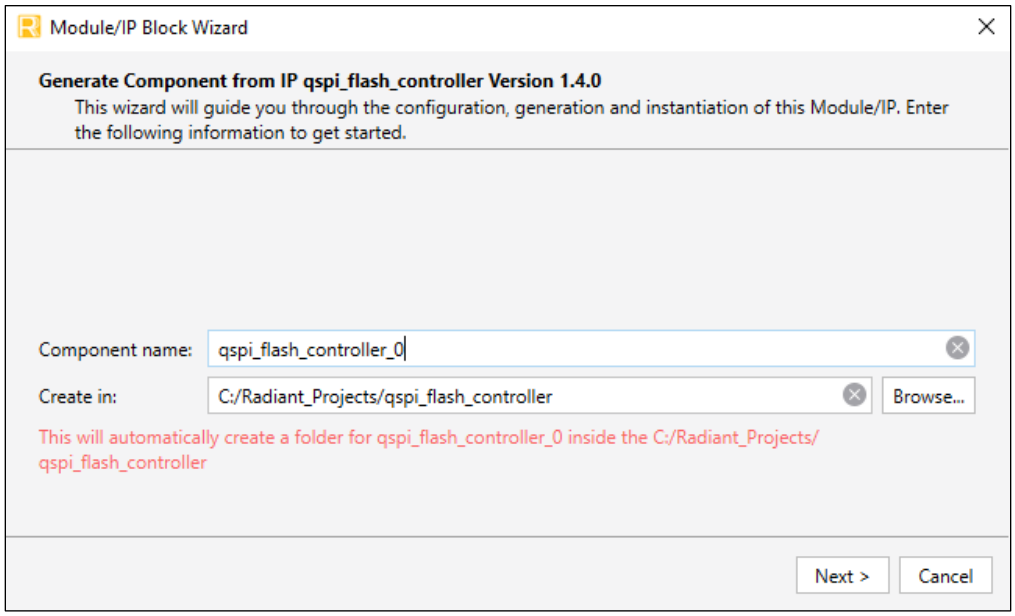

Figure 7.1. Module/IP Block Wizard

3. In the next **Module/IP Block Wizard** window, customize the selected QSPI Flash Controller IP using drop-down lists and check boxes. Figure 7.2 and Figure 7.3 show an example configuration of the QSPI Flash Controller IP. For details on the configuration options, refer to the IP Parameter Description section.

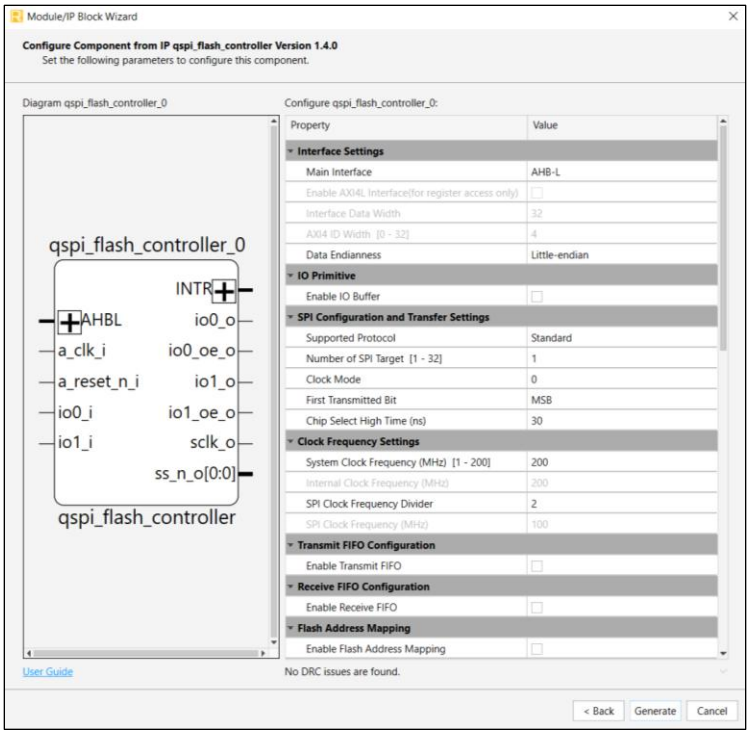

Figure 7.2. IP Configuration – View 1

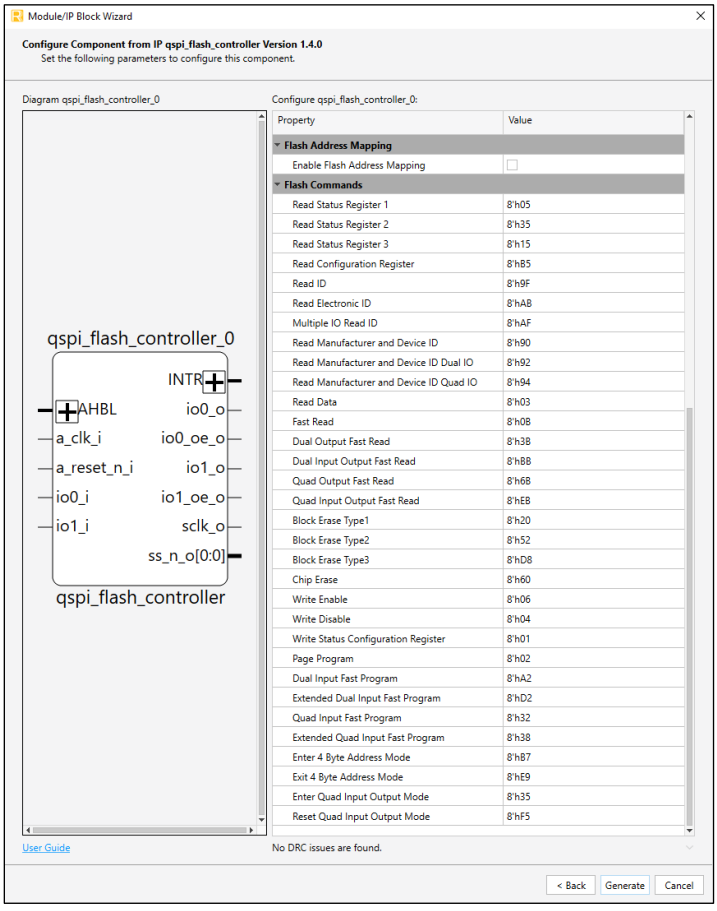

Figure 7.3. IP Configuration – View 2

4. Click **Generate**. The **Check Generating Result** dialog box opens, showing design block messages and results as shown in Figure 7.4.

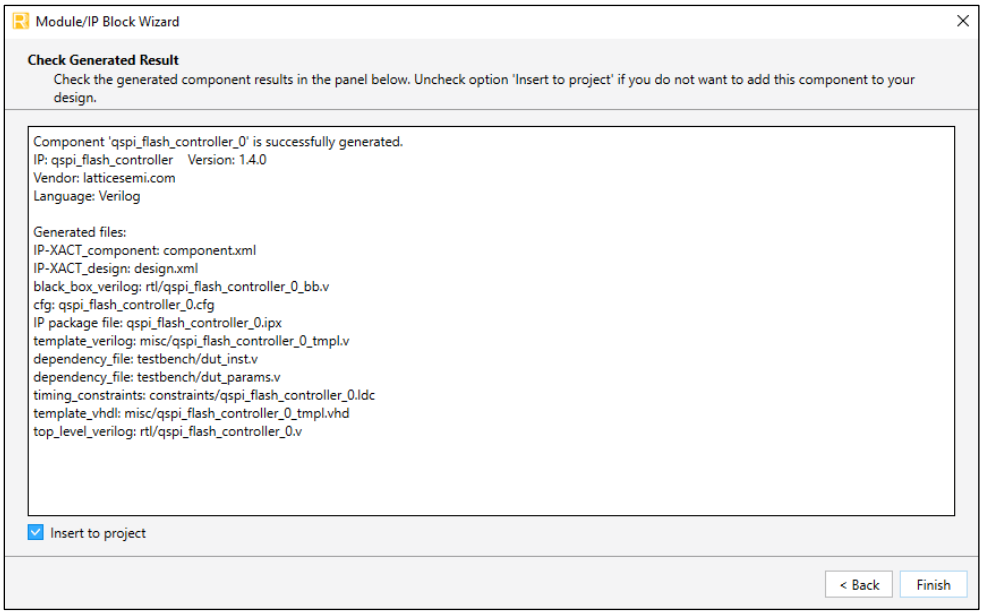

Figure 7.4. Check Generating Results

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 QSPI Flash Controller 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 example as the starting template for your top-level design. The generated files are listed in Table 7.1.

Table 7.1. Generated File List

Attribute	Description
constraints/<Component name>.ldc	IP constraint file.
driver/qspi_flash_cntl.c	This file is the driver source file for the IP written in C.
driver/qspi_flash.cntl.h	This file contains the header information for the IP driver.
misc/<Component name>_tmpl.v misc /<Component name>_tmpl.vhd	These files provide instance templates for the module.
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.
testbench/dut_inst.v	This file contains the IP instance instantiation.
testbench/dut_params.v	This file contains the parameter values set during IP configuration which is used by the tb_top.v for simulation.
testbench/readme.txt	This file contains information and steps on how to run simulation for the IP.
testbench/tb_top.v	Top-level testbench file of this IP.
component.xml	Contains the ipxact:component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format.
<Component name>.ipx	This file contains the information on the files associated to the generated IP.
<Component name>.cfg	This file contains the parameter values used in IP configuration.

7.2. Running Functional Simulation

You can run functional simulation after the IP is generated. The default simulator already has pre-compiled libraries ready for simulation. Choosing a non-default simulator, however, may require additional steps.

For this IP simulation testing, a flash simulation model should be instantiated and added on the testbench/tb_top.v file. Follow the procedures available on the testbench/readme.txt file.

To run functional simulation using the default simulator:

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

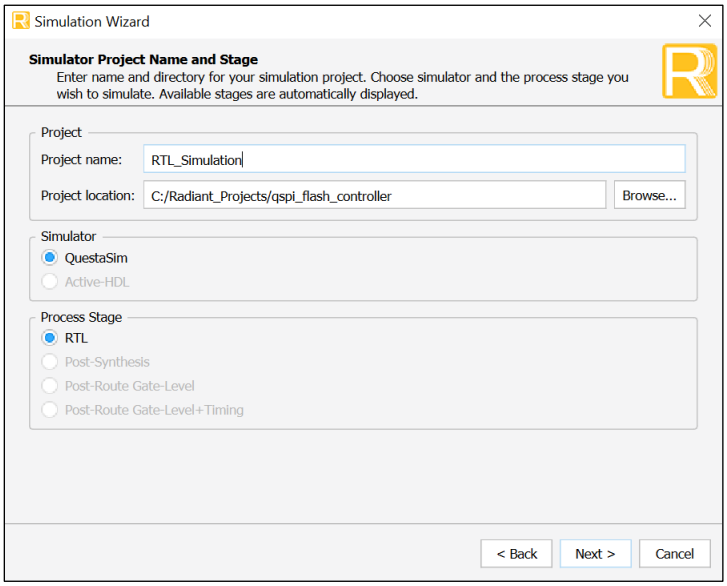

Figure 7.5. Simulation Wizard

2. Enter a project name and click **Next** to open the **Add and Reorder Source** window as shown in Figure 7.6.

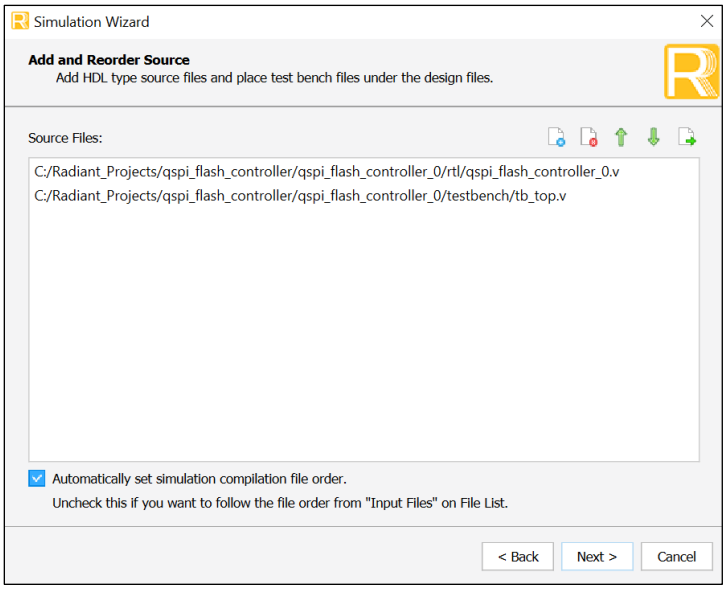

Figure 7.6: Adding Re-Ordering Source

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

4. Click **Finish** to run the simulation.

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

The results of the simulation in our example are provided in Figure 7.7.

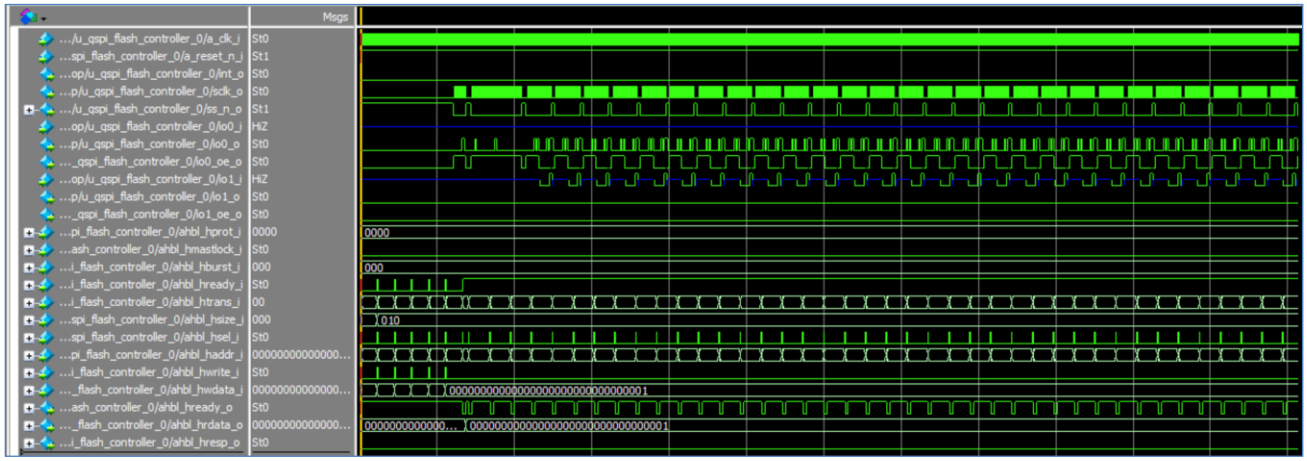

Figure 7.7. Simulation Waveform

7.3. Constraining the IP

It is your responsibility to provide proper timing and physical design constraints to ensure that the design meets the desired performance goals on the FPGA. The content of the following IP constraint file can be added to the user design constraints:

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

The above constraint file has been verified during IP evaluation with the IP instantiated directly in the top-level module. The constraint in this file can be modified given a complete understanding of the effect of the constraint.

To use this constraint file, copy the content of constraint.pdc to the top-level design constraint for post-synthesis.

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

Appendix A. Resource Utilization

Table A.1 through Table A.6 show the resource utilization of the QSPI Flash Controller IP core using different devices and Synplify Pro of the Lattice Radiant software. The default configuration is used, and some attributes are changed from the default value to show the effect on resource utilization.

Table A.1. Resource Utilization using LAV-AT-E70-3LFG1156I

Configuration	f_{MAX} (MHz)[1]	Registers	LUTs[2]	EBRs
AHB-L main interface, other settings on default	239.063	2063	4010	0
AXI4 main interface, other settings on default	224.517	2148	4139	0
AHB-L main interface, transmit and receive FIFOs enabled, dual SPI protocol, other settings on default	231.374	2042	3717	2
AXI4 main interface, transmit and receive FIFOs enabled, dual SPI protocol, other settings on default	240.038	2140	3853	2
AHB-L main interface with AXI4-L for register access, flash address mapping enabled, quad SPI protocol, other settings on default	224.215	2637	5191	0
AXI4 main interface with AXI4-L for register access, flash address mapping enabled, quad SPI protocol, other settings on default	224.669	2729	5218	0
AHB-L main interface with AXI4-L for register access, flash address mapping enabled, transmit and receive FIFOs enabled, quad SPI protocol, other settings on default	213.812	2634	4895	2
AXI4 main interface with AXI4-L for register access, flash address mapping enabled, transmit and receive FIFOs enabled, quad SPI protocol, other settings on default	224.618	2723	5065	2

Notes:

1. f_{MAX} is generated when the FPGA design only contains the QSPI Flash Controller IP core and the target frequency is 200 MHz. These values may be reduced when user logic is added to the FPGA design.
2. The distributed RAM utilization is accounted for in the total LUT4 utilization. The actual LUT4 utilization is distributed among logic, distributed RAM, and ripple logic.

Table A.2. Resource Utilization using LAV-AT-E70-1LFG1156I

Configuration	f_{MAX} (MHz)[1]	Registers	LUTs[2]	EBRs
AHB-L main interface, other settings on default	208.768	2063	4010	0
AXI4 main interface, other settings on default	198.491	2148	4139	0
AHB-L main interface, transmit and receive FIFOs enabled, dual SPI protocol, other settings on default	192.567	2042	3717	2
AXI4 main Interface, transmit and receive FIFOs enabled, dual SPI protocol, other settings on default	196.386	2140	3853	2
AHB-L main interface with AXI4-L for register access, flash address mapping enabled, quad SPI protocol, other	166.223	2637	5191	0

Configuration	f_{MAX} (MHz)[1]	Registers	LUTs[2]	EBRs
settings on default				
AXI4 main interface with AXI4-L for register access, flash address mapping enabled, quad SPI protocol, other settings on default	174.186	2729	5218	0
AHB-L main interface with AXI4-L for register access, flash address mapping enabled, transmit and receive FIFOs enabled, quad SPI protocol, other settings on default	166.389	2634	4895	2
AXI4 main interface with AXI4-L for register access, flash address mapping enabled, transmit and receive FIFOs enabled, quad SPI protocol, other settings on default	174.064	2723	5065	2

Notes:

1. f_{MAX} is generated when the FPGA design only contains the QSPI Flash Controller IP core and the target frequency is 200 MHz. These values may be reduced when user logic is added to the FPGA design.
2. The distributed RAM utilization is accounted for in the total LUT4 utilization. The actual LUT4 utilization is distributed among logic, distributed RAM, and ripple logic.

Table A.3. Resource Utilization using LFCPNX-100-9LFG672I

Configuration	f_{MAX} (MHz)[1]	Registers	LUTs[2]	EBRs
AHB-L main interface, other settings on default	137.080	2016	4007	0
AXI4 main interface, other settings on default	144.425	2160	4288	0
AHB-L main interface, transmit and receive FIFOs enabled, dual SPI protocol, other settings on default	136.091	2059	3710	2
AXI4 main interface, transmit and receive FIFOs enabled, dual SPI protocol, other settings on default	137.798	2124	3891	2
AHB-L main interface with AXI4-L for register access, flash address mapping enabled, quad SPI protocol, other settings on default	123.335	2684	5275	0
AXI4 main interface with AXI4-L for register access, flash address mapping enabled, quad SPI protocol, other settings on default	113.520	2802	5400	0
AHB-L main interface with AXI4-L for register access, flash address mapping enabled, transmit and receive FIFOs enabled, quad SPI protocol, other settings on default	102.312	2631	4942	2
AXI4 main interface with AXI4-L for register access, flash address mapping enabled, transmit and receive FIFOs enabled, quad SPI protocol, other settings on default	99.413	2816	5041	2

Notes:

1. f_{MAX} is generated when the FPGA design only contains the QSPI Flash Controller IP core and the target frequency is 125 MHz. These values may be reduced when user logic is added to the FPGA design.
2. The distributed RAM utilization is accounted for in the total LUT4 utilization. The actual LUT4 utilization is distributed among logic, distributed RAM, and ripple logic.

Table A.4. Resource Utilization using LFCPNX-100-7LFG672I

Configuration	f_{MAX} (MHz)[1]	Registers	LUTs[2]	EBRs
AHB-L main interface, other settings on default	101.75	2021	4033	0
AXI4 main interface, other settings on default	113.804	2148	4270	0
AHB-L main interface, transmit and receive FIFOs enabled, dual SPI protocol, other settings on default	105.708	2025	3737	2
AXI4 main interface, transmit and receive FIFOs enabled, dual SPI protocol, other settings on default	111.247	2127	3883	2
AHB-L main interface with AXI4-L for register access, flash address mapping enabled, quad SPI protocol, other settings on default	96.283	2682	5344	0
AXI4 main interface with AXI4-L for register access, flash address mapping enabled, quad SPI protocol, other settings on default	95.274	2753	5352	0
AHB-L main interface with AXI4-L for register access, flash address mapping enabled, transmit and receive FIFOs enabled, quad SPI protocol, other settings on default	100.321	2628	4940	2
AXI4 main interface with AXI4-L for register access, flash address mapping enabled, transmit and receive FIFOs enabled, quad SPI protocol, other settings on default	87.512	2732	5082	2

Notes:

1. f_{MAX} is generated when the FPGA design only contains the QSPI Flash Controller IP core and the target frequency is 125 MHz. These values may be reduced when user logic is added to the FPGA design.
2. The distributed RAM utilization is accounted for in the total LUT4 utilization. The actual LUT4 utilization is distributed among logic, distributed RAM, and ripple logic.

Table A.5. Resource Utilization using LFD2NX-40-9BG256I

Configuration	f_{MAX} (MHz)[1]	Registers	LUTs[2]	EBRs
AHB-L main interface, other settings on default	128.849	2016	4007	0
AXI4 main interface, other settings on default	138.850	2160	4288	0
AHB-L main interface, transmit and receive FIFOs enabled, dual SPI protocol, other settings on default	120.091	2059	3710	2
AXI4 main interface, transmit and receive FIFOs enabled, dual SPI protocol, other settings on default	127.372	2124	3891	2
AHB-L main interface with AXI4-L for register access, flash address mapping enabled, quad SPI protocol, other settings on default	111.445	2684	5275	0
AXI4 main interface with AXI4-L for register access, flash address mapping enabled, quad SPI protocol, other settings on default	126.534	2802	5400	0

Configuration	f_{MAX} (MHz)[1]	Registers	LUTs[2]	EBRs
AHB-L main interface with AXI4-L for register access, flash address mapping enabled, transmit and receive FIFOs enabled, quad SPI protocol, other settings on default	117.440	2631	4942	2
AXI4 main interface with AXI4-L for register access, flash address mapping enabled, transmit and receive FIFOs enabled, quad SPI protocol, other settings on default	111.000	2816	5041	2

Notes:

1. f_{MAX} is generated when the FPGA design only contains the QSPI Flash Controller IP core and the target frequency is 125 MHz. These values may be reduced when user logic is added to the FPGA design.
2. The distributed RAM utilization is accounted for in the total LUT4 utilization. The actual LUT4 utilization is distributed among logic, distributed RAM, and ripple logic.

Table A.6. Resource Utilization using LFD2NX-40-7BG256I

Configuration	f_{MAX} (MHz)[1]	Registers	LUTs[2]	EBRs
AHB-L main interface, other settings on default	78.284	2021	4033	0
AXI4 main interface, other settings on default	80.418	2148	4270	0
AHB-L main interface, transmit and receive FIFOs enabled, dual SPI protocol, other settings on default	79.911	2025	3737	2
AXI4 main interface, transmit and receive FIFOs enabled, dual SPI protocol, other settings on default	75.746	2127	3883	2
AHB-L main interface with AXI4-L for register access, flash address mapping enabled, quad SPI protocol, other settings on default	72.129	2682	5344	0
AXI4 main interface with AXI4-L for register access, flash address mapping enabled, quad SPI protocol, other settings on default	70.200	2753	5352	0
AHB-L main interface with AXI4-L for register access, flash address mapping enabled, transmit and receive FIFOs enabled, quad SPI protocol, other settings on default	63.955	2628	4940	2
AXI4 main interface with AXI4-L for register access, flash address mapping enabled, transmit and receive FIFOs enabled, quad SPI protocol, other settings on default	65.432	2732	5082	2

Notes:

1. f_{MAX} is generated when the FPGA design only contains the QSPI Flash Controller IP core and the target frequency is 125 MHz. These values may be reduced when user logic is added to the FPGA design.
2. The distributed RAM utilization is accounted for in the total LUT4 utilization. The actual LUT4 utilization is distributed among logic, distributed RAM, and ripple logic.

References

- QSPI Flash Controller IP Release Notes (FPGA-RN-02016)
- QSPI Driver API Reference (FPGA-TN-02339)
- RISC-V RX CPU IP User Guide (FPGA-IPUG-02254)
- A Step-By-Step Approach to Lattice Propel (FPGA-AN-02052)
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- Avant-E web page
- Avant-G web page
- Avant-X web page
- Certus-N2 web page
- Certus-NX web page
- CertusPro-NX web page
- CrossLink-NX web page
- MachXO5-NX web page
- CertusPro-NX Evaluation Board web page
- QSPI Flash Controller IP Core web page
- Lattice Radiant Software web page
- Lattice Propel Design Environment web page
- Lattice Solutions IP Cores web page
- Lattice Solutions Reference Designs web page
- 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, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.3, IP v1.5.0, June 2025

Section	Change Summary

Introduction	• In Table 1.1. Summary of the QSPI Flash Controller IP:
	  • Renamed *Supported FPGA Family* to *Supported Devices* and incorporated *Targeted Devices* information into row.
	  • Removed *Targeted Devices* row.
	  • Added IP core version to *Lattice Implementation*.
	  • Made minor editorial changes.

References	• Added QSPI Driver API Reference (FPGA-TN-02339). • Added QSPI Flash Controller IP Core web page. • Updated listing name to Lattice Radiant Software web page.

Revision 1.2, IP v1.4.0, December 2024

Section	Change Summary
All	Minor editorial changes.
Cover	Added IP version.

Introduction	• In Table 1.1. Summary of the QSPI Flash Controller IP:
	  • Added CrossLink-NX, MachXO5-NX, and Certus-N2 to *Supported FPGA Family*.
	  • Removed IP Version row.
	  • Added IP Changes row.
	  • Added LIFCL-17, LIFCL-33, LIFCL-40, LFMXO5-25, LFMXO5-55T, LFMXO5-100T, and LN2-CT-20 to *Targeted Devices*.
	  • Added Resources row.
	  • Added IP core v1.4.0 to *Lattice Implementation* and removed older entries.
	  • Added note on *Lattice Implementation*.
	• Added the IP Support Summary section.
	• In the Features section:
	  • Updated list of features for the different bus interfaces.
	  • Updated the programmable SPI clock description.
	  • Added generic command packet support.
	  • Removed reference to Macronix target device in relation to XiP for read access.
	• Removed the IP Validation Summary section.
	• Added the Hardware Support section.
	• Added CrossLink-NX, MachXO5-NX, and Certus-N2 devices in Table 1.3. Minimum Device Requirements for QSPI Flash Controller IP.

Functional Description	• Updated description for AXI4-Lite in Table 2.1. User Interfaces and Supported Protocols.
	• In the IP Core Operation section:
	  • Updated description.
	  • Added the Generic Commands Operation section.
	  • Updated description in the Non-XiP Access and eXecute-in-Place (XiP) Access sections.
	• In the User Packet Frame Structure section:
	  • Added Figure 2.9. Generic Command Packet with 1DW Header.
	  • Updated description to include reference to generic SPI operations.

IP Parameter Description	• In Table 3.1. General Attributes:
	  • Removed *SPI Clock Polarity* and *SPI Clock Phase* attributes.
	  • Added *Clock Mode* and *eXecute-in-Place (XiP) Pattern* attributes.
	  • Updated selectable values for *SPI Clock Frequency* under Clock Frequency Settings.
	  • Updated selectable values for *Address Depth* under Transmit FIFO Configuration and Receive FIFO Configuration.

Section	Change Summary
	• Updated description for *Enable Transmit FIFO*, *Enable Receive FIFO*, *Flash Memory Map Size*, and *AXI/AHB-L Address Map*. • Updated attribute name to *Flash Memory Map Size (KB)*. • Modified descriptions to indicate attributes that are visible only when a certain attribute selection is made. • In the SPI Clock Phase and Polarity section: • Updated description. • Removed clocking modes 1 and 2 in Table 3.2. Clocking Modes. • Removed the Clocking Mode 1 (SPI Clock Polarity=0, SPI Clock Phase=1) Waveforms and Clocking Mode 2 (SPI Clock Polarity=1, SPI Clock Phase=0) Waveforms figures. • In the IP Parameter Settings for Example Use Cases section: • Updated references to packet header registers in the Multiple SPI Targets section. • Updated description in the Transmit and Receive FIFO section.
Register Description	• In Table 5.2. Summary of QSPI Flash Controller IP Core Registers: • Added Generic SPI Transactions Packet Header and Data Transfer Status register. • Updated reserved address for Status and Interrupt Registers. • Updated description for Packet Header 0, Packet Data 0, and Packet Data 1. • In Table 5.3. QSPI Configuration Register 0: • Added en_frame_end_done_cntr bit. • Updated reserved to bit[14]. • Updated description for cpha and cpol. • Updated default value of dummy_clock_cycles, sck_rate, cpha, and cpol. • In Table 5.4. QSPI Configuration Register 1: • Updated description and default value of tgt_rd_trans_cnt and tgt_wr_trans_cnt. • Added statement on Lattice Interrupt Interface definition in the Status and Interrupt Registers section. • Updated description in the Interrupt Status Register when FIFO is Enabled section under the Interrupt Status Register section. • In Table 5.18. Interrupt Status Register when FIFO is Enabled: • Updated bit/field names by removing "_stat". • Updated descriptions of rx_fifo_full_int, rx_fifo_almost_full_int, rx_fifo_almost_empty_int, rx_fifo_empty_int, tx_fifo_full_int, tx_fifo_almost_full_int, tx_fifo_almost_empty_int, and tx_fifo_empty_int. • Updated default value for rx_fifo_empty_int and tx_fifo_empty_int. • In Table 5.19. Interrupt Status Register when FIFO is Disabled: • Updated bit/field names by removing "_stat". • Removed rd_pkt_data_1_not_empty_int_stat. • Updated reserved to bit[7:5]. • Updated description for rd_pkt_data_0_not_empty_int, wr_pkt_data_1_empty_int, and wr_pkt_data_0_empty_int. • Updated description in the Interrupt Enable Register when FIFO is Enabled section under the Interrupt Enable Register section. • In Table 5.20. Interrupt Enable Register when FIFO is Enabled: • Updated description for rx_fifo_full_int_en through tx_fifo_empty_int_en. • In Table 5.21. Interrupt Enable Register when FIFO is Disabled: • Updated bit/field names to tgt_rd_trns_cnt_hit_int_en and tgt_wr_trns_cnt_hit_int_en. • Removed rd_pkt_data_1_not_empty_int_en. • Updated reserved to bit[7:5]. • Updated description for rd_pkt_data_0_not_empty_int_en, wr_pkt_data_1_empty_int_en, and wr_pkt_data_0_empty_int_en. • Updated description in the Interrupt Set Register when FIFO is Enabled section under the Interrupt Set Register section.

Section	Change Summary

	• In Table 5.22. Interrupt Set Register when FIFO is Enabled:
	  • Updated bit/field names to tgt_rd_trns_cnt_hit_int_set and tgt_wr_trns_cnt_hit_int_set.
	  • Updated description for rx_fifo_full_int_set through tx_fifo_empty_int_set.
	• In Table 5.23. Interrupt Set Register when FIFO is Disabled:
	  • Updated bit/field names to tgt_rd_trns_cnt_hit_int_set and tgt_wr_trns_cnt_hit_int_set.
	  • Removed rd_pkt_data_1_not_empty_int_set.
	  • Updated reserved to bit[7:5].
	  • Updated description for rd_pkt_data_i_not_empty_int_set, wr_pkt_data_1_empty_int_set, and wr_pkt_data_0_empty_int_set.
	• Added the Generic SPI Transaction Packet Header and Data Transfer Status Register section under the Status and Interrupt Registers section.
	• Updated description in the Control Registers section.
	• In Table 5.28. Packet Header 0 Register – Supported Flash Commands:
	  • Updated description for xfer_len_bytes and sup_flash_cmd.
	• Added Packet Header 0 Register for Generic SPI Transactions section under the Packet Header 0 Register section.
	• In Table 5.32. Packet Header 2 Register and Table 5.33. Packet Header 3 Register:
	  • Updated description of register.
	• Added Generic Flash Write Operation and Generic Flash Read Operation sections.
	• In the Non-XiP Access (Direct Write/Read Access) through Flash Memory Map section:
	  • Updated description and rearranged Flash Memory Map Size item in list.
	  • Updated steps 4b and 5b.
	• In the XiP Access (Direct Read Access) through Flash Memory Map section:
	  • Updated description and sample sequence.
	  • Removed the Programming Flow: 1-Byte Quad Input/Output Fast Read, FIFO Disabled, Big Endian, Quad SPI table.
	  • Added Table 5.52. Programming Flow: 1-Byte Quad Input/Output Fast Read, FIFO Disabled, Big Endian, Table 5.53. Programming Flow: 4-Byte Address, FIFO Disabled, Big Endian, Table 5.54. Programming Flow: XiP Pattern and Dummy, FIFO Disabled, Big Endian, and Table 5.55. Programming Flow: Read 4-bytes Data, FIFO Disabled, Big Endian.
	• In the Polling Mode after a Write Transaction is started for FIFO Disabled Configuration section under the Polling Mode section:
	  • Updated bit/field names to wr_pkt_data_1_empty_int and wr_pkt_data_0_empty_int.
	• In the Polling Mode after a Read Transaction is started for FIFO Disabled Configuration section under the Polling Mode section:
	  • Updated bit/field name to rd_pkt_data_0_not_empty_int.
	• In the Polling Mode after a Read Transaction is started for FIFO Enabled Configuration section under the Polling Mode section:
	  • Updated bit/field name to rx_fifo_empty_int.

Example Design	• Added introductory paragraph to list the evaluation board used for the example design.
	• In Table 6.1. QSPI Flash Controller IP Configuration Used on Example Design:
	  • Removed *SPI Clock Polarity* and *SPI Clock Phase* attributes.
	  • Added *Clock Mode* and *eXecute-in-Place (XiP) Pattern* attributes.

Designing with the IP	• Updated Figure 7.1. Module/IP Block Wizard, Figure 7.2. IP Configuration – View 1, Figure 7.3. IP Configuration – View 2, and Figure 7.4. Check Generating Results.
	• In Table 7.1. Generated File List:
	  • Rearranged items.
	  • Added *testbench/readme.txt*, *driver/qspi_flash_cntl.c*, and *driver/qspi_flash.cntl.h*.
	  • Updated name to *constraints/<Component name>.ldc*.
	• Removed IP Evaluation section.

Section	Change Summary
Appendix A. Resource Utilization	Updated resource utilization values for f_{MAX}, registers, and LUTs in Table A.1. Resource Utilization using LAV-AT-E70-3LFG1156I through Table A.6. Resource Utilization using LFD2NX-40-7BG256I.
References	• Added QSPI Flash Controller IP Release Notes, RISC-V RX CPU IP User Guide, Certus-N2, CrossLink-NX, MachXO5-NX, and CertusPro-NX Evaluation Board web pages. • Removed Lattice Propel Builder User Guide and Lattice Propel SDK User Guide. • Added Lattice Propel Design Environment web page. • Updated and added links for IP and Reference Designs. • Removed Lattice Radiant Software User Guide. • Removed redundant item for Lattice Radiant Timing Constraints Methodology.

Revision 1.1, July 2024

Section	Change Summary
All	Minor editorial changes.
Acronyms and Abbreviation in This Document	Added an rearranged items.

Introduction	• Updated Overview of the IP and Features sections.
	• Updated supported FPGA families, IP version, targeted devices, Lattice implementation, synthesis, and simulation information in Table 1.1. Summary of the QSPI Flash Controller IP in Quick Facts section.
	• Updated the Licensing and Ordering Information section.
	• In IP Validation Summary section:
	  • Specified synthesis tool used for IP validation.
	  • In Table 1.2. IP Validation Level:
	  • Updated IP validation information.
	  • Removed validation in progress note.
	• Table 1.3. Minimum Device Requirements for QSPI Flash Controller IP in Minimum Device Requirements section:
	  • Removed Link Width column and added Device column.
	  • Added information by devices.
	• Removed Host section in Naming Conventions section.
	• Added Attribute Names section in Naming Conventions section.

Functional Description	• Updated Figure 2.1. QSPI Flash Controller IP Block Diagram in IP Architecture Overview section. • Removed Programming Flow section. • Added Clocking, Reset, User Interfaces, and Other IP Specific Blocks/Layers/Interfaces sections.

IP Parameter Description	• Table 3.1. General Attributes in General section:
	  • Renamed and moved table into General section.
	  • Incorporated default information into Selectable Values column and removed Default column.
	  • Removed Attribute Description table and merged attribute descriptions into table.
	  • Added, updated, or removed dependency information for various attributes.
	  • Updated various attribute names.
	  • Updated description for various attributes.
	  • Updated selectable and default values for various attributes.
	  • Removed attributes Enable Back-to-Back Transfer and Minimum Idle Time between Transfer (number of SCK).
	  • Added attributes Register Space Size, Register Map Base Address, Write Access Command, Read Access Command, Flash Address Width, Command Lane Width, Address Lane Width, Data Lane Width, and Dummy clock cycles under Flash Address Mapping.

Section	Change Summary
	• Renamed table segment rows to *Flash Address Mapping* and *Supported Flash Commands*. • Split *Transmit and Receiver FIFO Configuration* into *Transmit FIFO Configuration* and *Receive FIFO Configuration*. • Added IP Parameter Settings for Example Use Cases section.
Signal Description	• Table 4.1. Top-Level Ports: • Renamed title to *Top-Level Ports*. • Renamed column to *Type*. • Updated various attribute names. • Removed attributes sck_i and ss_n_i. • Updated width, output reset value, and description for various attributes. • Renamed table segment rows to *Disabled IO Buffer*, *Enabled IO Buffer*, *AHB-Lite Interface*, *AXI4-Lite Interface*, and *AXI Interface*. • Rearranged attributes under Disabled IO Buffer.
Register Description	• Removed offset address from section titles. • Renamed Bits and Reset columns in register tables to Field and Default columns, respectively. • Added Table 5.1. Register Access Types. • Table 5.2. Summary of QSPI Flash Controller IP Core Registers in Overview section: • Updated description for registers Flash Command Code 0 through 7, Minimum Flash Address Alignment, AXI/AHB-L Address Map, Packet Header 0, Packet Data 0, and Packet Data 1. • Updated register name to AXI/AHB-L Address Map. • Changed offset address 0x0000_0114 and 0x0000_0224 to reserved. • Updated Configuration Registers section. • In QSPI Configuration Register 0 section: • Removed descriptions for registers en_flash_address_space_map, chip_select_behavior, min_idle_time, and en_back_to_back_trans. • Table 5.3. QSPI Configuration Register 0: • Added configuration register 0 bit names for bits 16 through 31 and changed bits 3 through 7 and 14 to both reserved and read only. • Merged register descriptions into table. • Added descriptions for registers dat_lane_width, addr_lane_width, cmd_lane_width, xip_addr_width, dummy_clock_cycles, flash_map_en_rd_access_cmd, and flash_map_en_wr_access_cmd. • Changed bit 15 to reserved and read only. • Updated descriptions for registers sck_rate, cpha, and cpol. • Table 5.4. QSPI Configuration Register 1: • Corrected table title. • Merged register descriptions into table. • Updated Flash Command Code 0 Register through Flash Command Code 7 Register sections. • In Minimum Flash Address Alignment Register section: • Table 5.13. Minimum Flash Address Alignment Register: • Updated and added description for min_flash_addr_alignment into table. • Updated default value. • In Starting Flash Address Space Mapping Register section: • Table 5.14. Starting Flash Address Space Mapping Register: • Updated and added description for start_flash_address_space_map into table. • In Flash Memory Map Size Register section: • Updated section title. • Table 5.15. Flash Memory Map Size Register: • Updated table title.

Section	Change Summary
	• Updated and added description for flash_memory_map_size into table. • Updated register name and default value. • In AXI/AHB-L Address Map Register section: • Updated section title. • Table 5.16. AXI/AHB-L Address Map Register: • Updated table title and bit field name. • Added description for axi_ahbl_address_map and merged description into table. • In Status and Interrupt Registers section: • Reorganized information in Interrupt Status Register section into subsections and merged register descriptions into tables. • Reorganized information in Interrupt Enable Register section into subsections and merged register descriptions into tables. • Reorganized information in Interrupt Set Register section into subsections and merged register descriptions into tables. • In Supported Flash Command Packet Header and Data Transfer Status Register section: • Merged description for registers into table. • Updated descriptions of registers in Table 5.24. Supported Flash Command Transaction Status Register. • Removed Unsupported Flash Command Packet Header and Data Transfer Status Register section. • In Control Registers section: • Updated description. • Updated TX FIFO Mapping and RX FIFO Mapping sections. • Reorganized information in Packet Header 0 Register into subsections. • In Packet Header 0 Register section: • Updated description. • In Table 5.27. Packet Header 0 Register – Supported Flash Commands: • Updated bit field name to flash_command_code. • Merged register descriptions into table. • Updated description for flash_command_code and sup_flash_cmd. • Updated num_wait_sck setting and description. • Removed Packet Header 0 Register for Unsupported Flash Commands section. • In Packet Header 1 Register section: • Updated description. • In Table 5.29. Packet Header 1 Register: • Merged register descriptions into table. • Updated reserved bit fields to read-only access and tgt_cs to read-write access. • Updated description for tgt_cs. • Updated Packet Header 2 Register section. • Updated Packet Header 3 Register section. • Updated Packet Data 0 Register section. • Updated Packet Data 1 Register section. • In Start Transaction Register section: • Updated description. • In Table 5.34. Start Transaction Register: • Added xip_enable and updated reserved bits information. • Added description for xip_enable. • Merged register description into table. • Removed Enable Loopback Testing Register section. • Added Operation Details/Programming Flow section.
Example Design	Added Example Design section.
Designing with the IP	• In Generating and Instantiating the IP section:

Section	Change Summary
	• Renamed section. • Updated Figure 7.1. Module/IP Block Wizard, Figure 7.2. IP Configuration – View 1, and Figure 7.4. Check Generating Results. • Added reference to Figure 7.3. IP Configuration – View 2. • Added Figure 7.3. IP Configuration – View 2. • In Generated Files and File Structure section: • Updated description. • Updated Table 7.1. Generated File List. • In Running Functional Simulation section: • Updated description. • Updated Figure 7.5. Simulation Wizard and Figure 7.6: Adding Re-Ordering Source. • Removed Parse HDL Files for Simulation step and figure. • Removed Summary Wizard figure and reference to figure in step 3. • Removed reference to ModelSim in step 4. • Added Figure 7.7. Simulation Waveform and note referencing figure. • Moved and updated Constraining the IP section. • Added IP Evaluation section.
Appendix	• Removed Resource Utilization table. • Added Table A.1. Resource Utilization using LAV-AT-E70-3LFG1156I through Table A.6. Resource Utilization using LFD2NX-40-7BG256I.
References	Added references.

Revision 1.0, December 2023

Section	Change Summary
All	Initial release.

www.latticesemi.com