Lattice Sentry QSPI Controller Streamer IP – Lattice Propel Builder

IP Version: 1.2.0

User Guide

FPGA-IPUG-02109-1.1

December 2025

Disclaimers

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

Inclusive Language

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

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
CPU	Central Processing Unit
EBR	Embedded Block RAM
ESB	Embedded Security Block
FIFO	First In, First Out
FPGA	Field-Programmable Gate Array
HDL	Hardware Description Language
IP	Intellectual Property
LUT	Look-Up Table
PP	Page Program
QREAD4B	Quad Read with 4-Byte Addressing
QSPI	Quad Serial Peripheral Interface
RDID	Read Identification
RO	Read Only
RW	Read and Write
RW1C	Read and Write 1 to Clear
Rx	Receiver
SHA	Secure Hash Algorithm
SPI	Serial Peripheral Interface
Tx	Transmitter
WO	Write Only

1. Introduction

A Quad-Serial Peripheral Interface (QSPI) is a serial interface, where four data lines are used to read, write, and erase flash chips. It is faster than the traditional Serial Peripheral Interface (SPI) and is specifically designed to communicate with flash chips that support this interface. Unlike the traditional SPI that uses separate data lines for input and output, MISO and MOSI, the QSPI interface configures the data lines on the fly so that they act as outputs to send some information to the flash memory and act as inputs to read some memory contents.

The Lattice Semiconductor Sentry™ QSPI Controller Streamer IP supports SPI and QSPI transactions. The design is implemented in Verilog HDL. It can be configured and generated using Lattice Propel™ Builder and implemented using the Lattice Diamond™ or Lattice Radiant™ software.

1.1. Features

The key features of the QSPI Controller Streamer IP include:

- Generation of SPI and QSPI transactions
- Support for long SPI transactions: up to 256-byte write and 4 Gb read with no CPU interactions
- Programmable transaction type and length
- Provision of external 8-bit FIFO interface for connecting to other blocks
- Support for Advanced Microcontroller Bus Architecture (AMBA 3) Advanced Peripheral Bus (APB) Protocol v1.0

1.2. Conventions

1.2.1. Nomenclature

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

1.2.2. 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

1.2.3. Host

The logic unit inside the FPGA interacts with the QSPI Controller Streamer IP through APB.

1.2.4. Attribute Names

Attribute names in this document are formatted in title case and italicized.

2. Functional Description

The QSPI Controller Streamer is a configurable SPI controller, which can support SPI and QSPI targets. It contains FIFOs for Tx and Rx data, which support page read and page program of 256 bytes. It also provides an 8-bit external Rx FIFO output interface which can be connected to other IP blocks to stream data in long bursts.

The QSPI Controller Streamer provides significant performance improvement by supporting data read and write transactions of programmable length, allowing an entire SPI flash device to be read in one SPI transaction. The 8-bit external Rx FIFO output interface also enables direct transmission of input data from the SPI target to another block, without tying up the CPU or system bus.

2.1. Block Diagram

QSPI Controller Streamer Block Diagram is shown in Figure 2.1. There are Tx and Rx FIFOs with each having a 32-bit access port for the APB system bus and an 8-bit access port for the SPI Controller state machine. 8-bit data is packed or unpacked into 32-bit chunks as it enters or leaves the FIFOs. The endianness of the 32-bit data is determined by the *Tx FIFO Endianness* and *Rx FIFO Endianness* in the attributes table (Table 2.2).

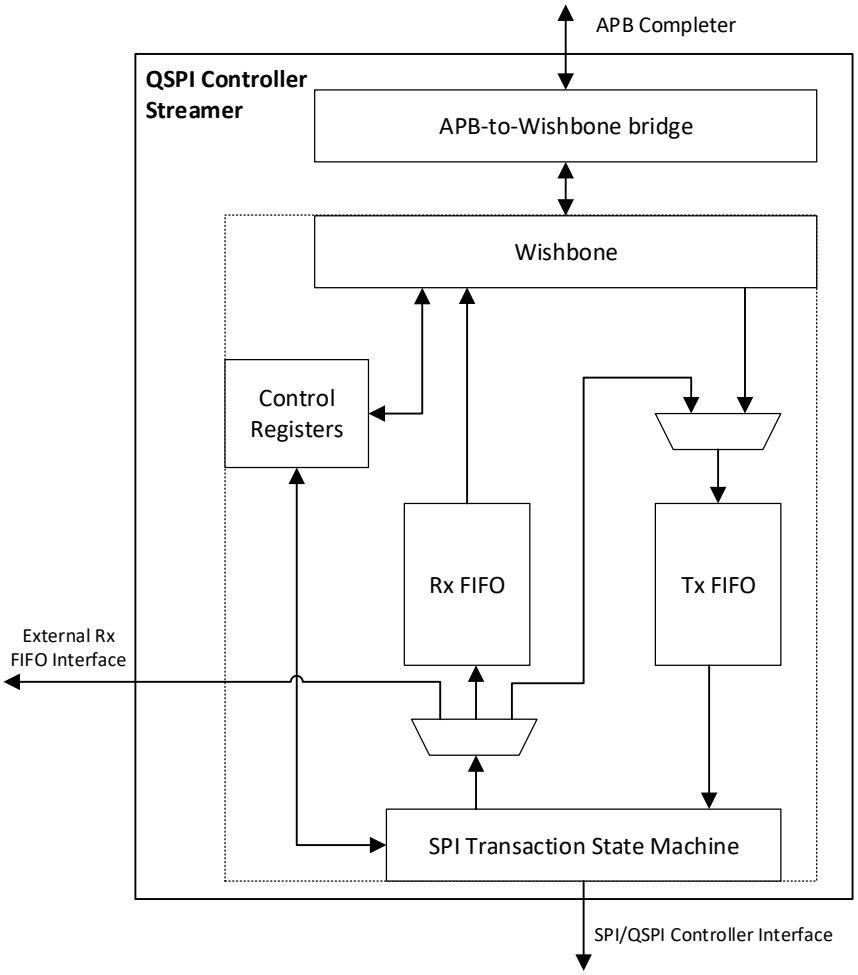

Figure 2.1. QSPI Controller Streamer Block Diagram

2.2. Signal Description

Table 2.1. QSPI Controller Streamer Signal Description

Port	Width	Direction	Description
System			
clk_i	1	Input	SPI Controller clock input
reset_i	1	Input	Asynchronous reset active high
int_o	1	Output	Interrupt request
APB			
apb_psel_i	In	1	Select signal Indicates that the completer device is selected and a data transfer is required.
apb_paddr_i	In	32	Address signal
apb_pwdata_i	In	32	Write data signal
apb_pwrite_i	In	1	Direction signal Write = 1, Read = 0
apb_penable_i	In	1	Enable signal Indicates the second and subsequent cycles of an APB transfer.
apb_pready_o	Out	1	Ready signal Indicates transfer completion. The completer uses this signal to extend an APB transfer.
apb_prdata_o	Out	32	Read data signal
QSPI Controller			
spi_mst_csn_o	1	Output	Chip select
spi_mst_sck_o	1	Output	SPI/QSPI clock
spi_mst_si_i	4	Input	SPI: spi_mst_si_i[1]=MISO, spi_mst_si_i[3:2]=unused, spi_mst_si_i[0]=unused QSPI: spi_mst_si_i[3:0] = serial data input
spi_mst_so_o	4	Output	SPI: spi_mst_so_o[0]=MOSI, spi_mst_so_o[3:1]=unused QSPI: spi_mst_so_o[3:0] = serial data output
spi_mst_oe_o	3	Output	spi_mst_oe_o[0]: direction control for qpi_sio0 I/O pad (1=output, 0=input) spi_mst_oe_o[1]: direction control for qpi_sio1 I/O pad (1=output, 0=input) spi_mst_oe_o[2]: direction control for qpi_sio2 and qpi_sio3 I/O pads (1=output, 0=input)
External Rx FIFO (Optional)*			
rxfifo_clk_o	1	Output	Clock output for external Rx FIFO
rxfifo_valid_o	1	Output	Output data is valid.
rxfifo_data_o	8	Output	SPI input data to write to external FIFO
rxfifo_last_o	1	Output	Indicates that the current data output is the last received byte of the SPI transaction. Example usage: Connect this signal to bit 31 on Embedded Security Block (ESB)'s High Speed Port to use as the last_byte_indicator for SHA calculations.
rxfifo_full_i	1	Input	FIFO full indicator Any SPI or QSPI transactions in progress stall until FIFO is no longer full.

***Note**: This interface is only present when *No. of External Rx FIFO Interfaces* > 0.

2.3. Attribute Summary

The QSPI Controller Streamer IP's configurable attributes are shown in Table 2.1 and are described in Table 2.3.

Table 2.2. Attributes Table

Attribute	Selectable Values	Default	Dependency on Other Attributes
General			
SPI Mode	0, 3	0	—
Configuration			
SPI Clock Divider	0, 1, 2, 3, 4, 5	2	—
Tx FIFO Size	4–512	512	—
Tx FIFO Almost Full Flag	4–Tx FIFO Size	256	—
Tx FIFO Almost Empty Flag	4–Tx FIFO Size	4	—
Tx FIFO Endianness	big, little	big	—
Rx FIFO Size	4–1024	256	—
Rx FIFO Almost Full Flag	4–Rx FIFO Size	252	—
Rx FIFO Almost Empty Flag	4–Rx FIFO Size	4	—
Rx FIFO Endianness	big, little	big	—
No. of External Rx FIFO Interfaces	0, 1	1	—

Table 2.3. Attribute Description

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>General</td><td></td></tr>
<tr><td>SPI Mode</td><td>Default value for the spi_mode bit field in the qspi_ctrl register</td></tr>
<tr><td>Configuration</td><td></td></tr>
<tr><td>SPI Clock Divider</td><td>Default value for the clock divisor sck_div bit field in the qspi_ctrl register
0: Fqpi_sck_o = Fclk_i
1: Fqpi_sck_o = Fclk_i/2
2: Fqpi_sck_o = Fclk_i/4
3: Fqpi_sck_o = Fclk_i/8
4: Fqpi_sck_o = Fclk_i/16
5: Fqpi_sck_o = Fclk_i/32</td></tr>
<tr><td>Tx FIFO Size</td><td>Size of the transmit FIFO in bytes
It must be a multiple of 4 to ensure it is 32-bit aligned.</td></tr>
<tr><td>Tx FIFO Almost Full Flag</td><td>Threshold value for signaling that the FIFO is almost full</td></tr>
<tr><td>Tx FIFO Almost Empty Flag</td><td>Threshold value for signaling that the FIFO is almost empty</td></tr>
<tr><td>Tx FIFO Endianness</td><td>Specifies the order of Tx FIFO bytes at the 32-bit APB interface.
Transmit bytes over SPI in this order, from 0–3:
<table>
<tr><th>APB Tx FIFO Data</th><th>31:24</th><th>23:16</th><th>15:8</th><th>7:0</th></tr>
<tr><td>Big endian</td><td>0</td><td>1</td><td>2</td><td>3</td></tr>
<tr><td>Little endian</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table></td></tr>
<tr><td>Rx FIFO Size</td><td>Size of the receive FIFO in bytes
It must be a multiple of 4 to ensure it is 32-bit aligned.</td></tr>
<tr><td>Rx FIFO Almost Full Flag</td><td>Threshold value for signaling that the FIFO is almost full</td></tr>
<tr><td>Rx FIFO Almost Empty Flag</td><td>Threshold value for signaling that the FIFO is almost empty</td></tr>
<tr><td>Rx FIFO Endianness</td><td>Specifies the order of Rx FIFO bytes at the 32-bit APB interface.
Received bytes from SPI are packed in this order, from 0–3:
<table>
<tr><th>APB Rx FIFO Data</th><th>31:24</th><th>23:16</th><th>15:8</th><th>7:0</th></tr>
<tr><td>Big endian</td><td>0</td><td>1</td><td>2</td><td>3</td></tr>
<tr><td>Little endian</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table></td></tr>
</table>

Parameter	Description
No. of External Rx FIFO Interfaces	Number of external 8-bit Rx FIFO interfaces The default number is 1.

2.4. Register Description

The QSPI Controller Streamer IP register map is shown in the Table 2-4.

Table 2.4. Summary of QSPI Controller Streamer IP Registers

Offset	Name	Access	Description

0x00	QSPI_CTRL	RW	• spi_mode[1:0]
			  • 00: SPI mode 0
			  • 01: reserved
			  • 10: reserved
			  • 11: SPI mode 3
			• sck_div[4:2]
			  • 0: Fqpi_sck_o = Fclk_i
			  • 1: Fqpi_sck_o = Fclk_i/2
			  • 2: Fqpi_sck_o = Fclk_i/4
			  • 3: Fqpi_sck_o = Fclk_i/8
			  • 4: Fqpi_sck_o = Fclk_i/16
			  • 5: Fqpi_sck_o = Fclk_i/32
			• reserved[30:5]
			• soft_reset[31]
			Writing 1 to this bit resets all of the internal logic, flushes the FIFOs by reseting the read and write pointers, and restores all registers to their default settings. Reads return 0. This is intended for error recovery.

0x04	CMD_DATA	RW	Command data to transmit in transaction phase 1. It is always big endian.
0x08	TX_FIFO_DATA	WO	Data to transmit in transaction phase 2 When the Tx FIFO is full, register writes to this address is blocked until the FIFO is no longer full. Tx FIFO status is available in the fifo_ctrl and int_status registers. The endianness depends on the *Tx FIFO Endianness* attribute.
0x0C	RX_FIFO_DATA	RO	Data received in transaction phase 4 If the Rx FIFO contains less than four bytes when a 32-bit read is received on the system bus and there is an SPI transaction currently in progress, the read is blocked until 4 bytes are received or the SPI transaction completes. The endianness depends on the *Rx FIFO Endianness* attribute.

0x10	TRANSACTION_CTRL1	RW	• ph1_num_bytes[2:0] – Number of bytes from cmd_data to transmit in transaction phase 1. The legal values are 0–4.
			• ph2_num_bytes[11:3] – Number of bytes from Tx FIFO to transmit in transaction phase 2. The legal values are 0–*Tx FIFO Size*.
			• ph3_dummy_cycles[16:12] – Number of dummy cycles to transmit in transaction phase 3
			• ph1_mode[18:17] – Transmit phase 1 data in:
			  • 0: SPI mode
			  • 1: reserved
			  • 2: QSPI mode
			  • 3: reserved
			• ph2_mode[20:19] – Transmit phase 2 data in:
			  • 0: SPI mode
			  • 1: reserved
			  • 2: QSPI mode
			  • 3: reserved

Offset	Name	Access	Description

			• ph3_mode[22:21] – Transmit phase 3 dummy cycles in:
			  • 0: SPI mode
			  • 1: reserved
			  • 2: QSPI mode
			  • 3: reserved
			• ph4_mode[24:23] – Receive phase 4 data in:
			  • 0: SPI mode
			  • 1: reserved
			  • 2: QSPI mode
			  • 3: reserved
			• rxfifo_last_en[25] – Enable(1)/Disable(0) assertion of rxfifo_last_o for the last received byte of the SPI transaction
			• reserved[30:26]
			• start[31] – Write 1 to start an SPI transaction. Reading of this bit returns 0.

0x14	TRANSACTION_CTRL2	RW	ph4_num_bytes[31:0] – Number of bytes to receive in transaction phase 4
0x18	STATUS	RO	• tx_fifo_empty[0] – Tx FIFO is empty. • tx_fifo_almost_empty[1] – Tx FIFO is not empty and has bytes less than *Tx FIFO Almost Empty Flag*. • tx_fifo_almost_full[2] – Tx FIFO is not full and has bytes more than *Tx FIFO Almost Full Flag*. • tx_fifo_full[3] – Tx FIFO is full. • rx_fifo_empty[4] – Rx FIFO is empty. • rx_fifo_almost_empty[5] – Rx FIFO is not empty and has bytes less than *Rx FIFO Almost Empty Flag*. • rx_fifo_almost_full[6] – Rx FIFO is not full and has bytes more than *Rx FIFO Almost Full Flag*. • reserved[30:8] • busy[31] – SPI transaction is in progress.

0x1C	FIFO_CTRL	RW	• reserved[6:0]
			• tx_fifo_flush[7] – Flush contents of Tx FIFO by resetting read and write pointers.
			• rx_fifo_dest[9:8]:
			  • 0: internal Rx FIFO
			  • 1: external Rx FIFO interface
			  • 2: reserved
			  • 3: internal Tx FIFO
			• reserved[14:10]
			• rx_fifo_flush[15]: Flush contents of Rx FIFO by resetting read and write pointers.
			• reserved[31:16]

0x20	INT_STATUS	RW	Interrupt status: • done_int[0] – Done interrupt. The SPI transaction is completed. • tx_fifo_empty_int[1] – Tx FIFO Empty interrupt • tx_fifo_almost_empty_int[2] – Tx FIFO Almost Empty interrupt • tx_fifo_almost_full_int[3] – Tx FIFO Almost Full interrupt • tx_fifo_full_int[4] – Tx FIFO Full interrupt • rx_fifo_empty_int[5] – Rx FIFO Empty interrupt • rx_fifo_almost_empty_int[6] – Rx FIFO Almost Empty interrupt • rx_fifo_almost_full_int[7] – Rx FIFO Almost Full interrupt • rx_fifo_full_int[8] – Rx FIFO Full interrupt • reserved[31:9] Writing 1 to a bit clears that interrupt.

Offset	Name	Access	Description
			FIFO interrupts are triggered on the rising edge of the corresponding FIFO condition, such as empty or full and stay asserted until cleared by writing a 1 to this register to clear the interrupt. Current status of the FIFO conditions is always available in the status register.
0x24	INT_ENABLE	RW	Interrupt enable: • done_en[0] – Enable Done interrupt. The SPI transaction is completed. • tx_fifo_empty_en[1] – Enable Tx FIFO Empty interrupt • tx_fifo_almost_empty_en[2] – Enable Tx FIFO Almost Empty interrupt • tx_fifo_almost_full_en[3] – Enable Tx FIFO Almost Full interrupt • tx_fifo_full_en[4] – Enable Tx FIFO Full interrupt • rx_fifo_empty_en[5] – Enable Rx FIFO Empty interrupt • rx_fifo_almost_empty_en[6] – Enable Rx FIFO Almost Empty interrupt • rx_fifo_almost_full_en[7] – Enable Rx FIFO Almost Full interrupt • rx_fifo_full_en[8] – Enable Rx FIFO Full interrupt • reserved[31:9]
0x28	INT_SET	RW	Interrupt set: • done_set[0] – Set Done interrupt. SPI transaction is completed. • tx_fifo_empty_set[1] – Set Tx FIFO Empty interrupt. • tx_fifo_almost_empty_set[2] – Set Tx FIFO Almost Empty interrupt. • tx_fifo_almost_full_set[3] – Set Tx FIFO Almost Full interrupt. • tx_fifo_full_set[4] – Set Tx FIFO Full interrupt. • rx_fifo_empty_set[5] – Set Rx FIFO Empty interrupt. • rx_fifo_almost_empty_set[6] – Set Rx FIFO Almost Empty interrupt. • rx_fifo_almost_full_set[7] – Set Rx FIFO Almost Full interrupt. • rx_fifo_full_set[8] – Set Rx FIFO Full interrupt. • reserved[31:9]

The behavior of registers to write and read access is defined by its access type, which is defined in Table 2.5.

Table 2.5. Access Type Definition

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

2.5. APB Completer Interface

The APB Completer interface provides an APB completer interface for CPU access of the register set.

2.6. External FIFO Interface

The external FIFO interface supports the transfer of large streams of data from one SPI flash to another SPI flash. An example of this is boot image recovery between two separate SPI flash devices. Enabling this feature adds an external 8-bit interface into the Tx FIFO so that the Rx FIFO output of one QSPI Controller Streamer can be connected to the Tx FIFO input of a separate QSPI Controller Streamer. In this configuration, the CPU can set up an SPI flash read transaction for the source flash device and an SPI page program transaction for the destination flash device. The data is streamed directly between the two without further CPU interaction.

2.7. Operation

2.7.1. Transaction Phases

The QSPI Controller Streamer generates an SPI or a QSPI transaction in multiple phases, as shown in Figure 2.2. Each phase is controlled by separate register settings. In the typical usage model, the CPU programs all of the transaction phase registers with the settings for the desired transaction, then programs the Start register to start the transaction. For transactions which use data, the CPU must write data to the FIFO before starting the transaction. See example sequences below for details.

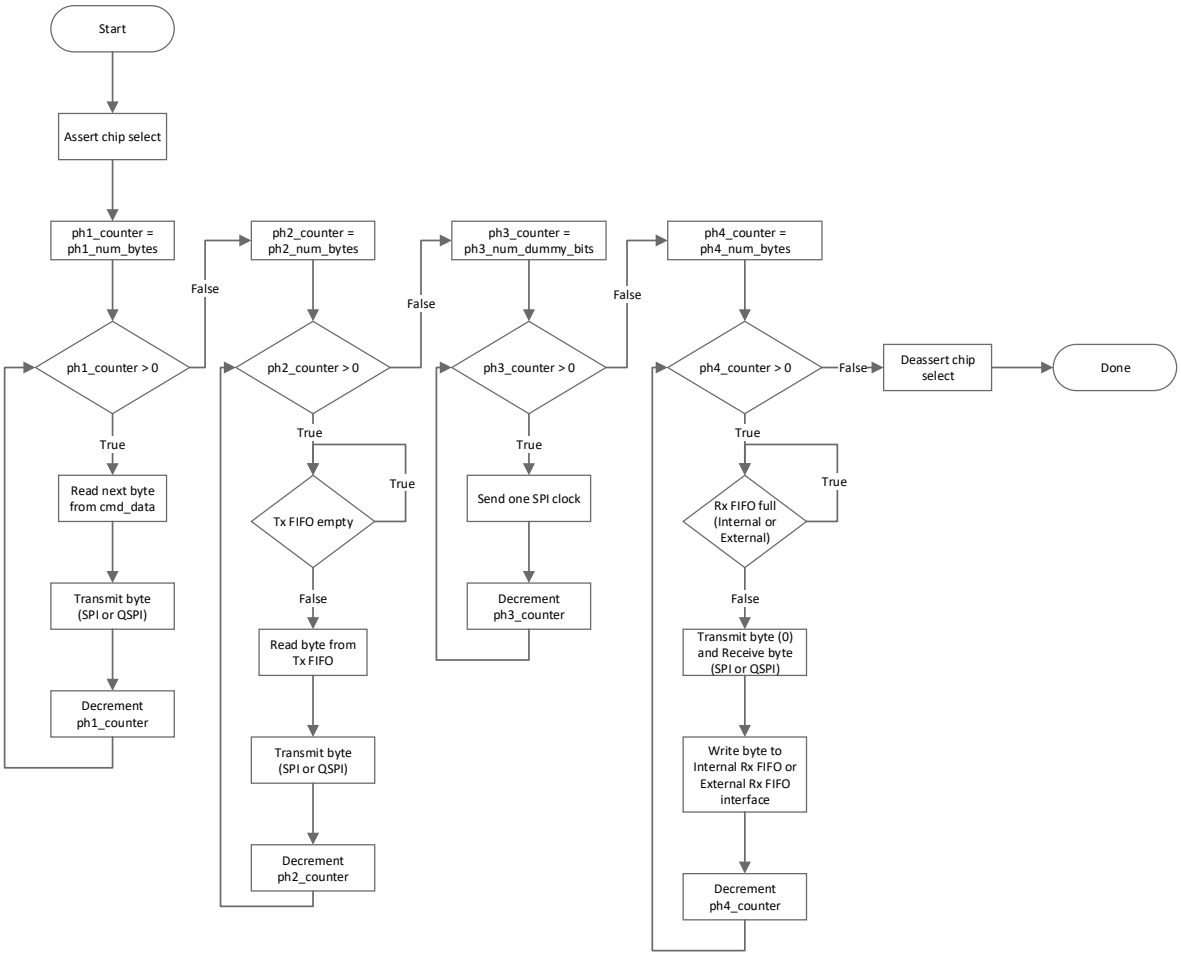

Figure 2.2. QSPI Controller Streamer Programmable Phases

Phase 1: Transmit ph1_num_bytes ranging from 0–4 bytes from the cmd_data register.

- For SPI flash devices, this normally includes one command byte and zero or three address bytes.
- Data is transmitted in the SPI mode or QSPI mode depending on the ph1_mode setting in transaction_ctrl1.
- Serial data input is ignored.

Phase 2: Transmit ph2_num_bytes ranging from 0–1028 bytes from Tx FIFO.

- For SPI flash devices, this is normally used for page program data and/or 4-byte addressing.
- Data is transmitted in the SPI mode or QSPI mode depending on the ph2_mode setting in transaction_ctrl1.
- Serial data input is ignored.

Phase 3: Transmit ph3_num_dummy_bits ranging from 0–15 bits.

- For SPI flash devices, this is normally used to generate dummy cycles for read data commands.
- Dummy data (0) is transmitted in the SPI mode or QSPI mode depending on the ph3_mode setting.
- Serial data input is ignored.

Phase 4: Receive ph4_num_bytes ranging from 0–4GB bytes and send to Rx FIFO.

- For SPI flash devices, this is normally used for read commands.
- Data is received in SPI mode or QSPI mode depending on the ph4_mode setting.
- Received data is stored in Rx FIFO or sent out the External Rx FIFO interface depending on the rx_fifo_dest.
- Serial data output is 0 for SPI or high impedance for QSPI.
- The SPI target ignores the data.

SPI Flash Page Program (PP) example:

cmd_data = 0x02xxxxxx (where xxxxxx = 24-bit Flash address).

Tx FIFO contains DataByte1…DataByte16 values

ph1_num_bytes = 4, ph1_mode = 0

ph2_num_bytes = N, ph2_mode = 0 (N=16 in this example)

ph3_num_dummy_bits = 0, ph3_mode = 0

ph4_num_bytes = 0, ph4_mode = 0

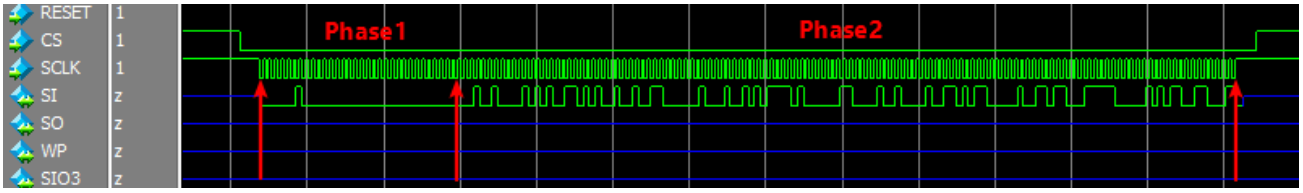

Figure 2.3. Example for PP Program Sequence

SPI Flash FAST_READ example:

cmd_data = 0x0Bxxxxxx (where xxxxxx = 24-bit address).

ph1_num_bytes = 4, ph1_mode = 0

ph2_num_bytes = 0, ph2_mode = 0

ph3_num_dummy_bits = N, ph3_mode = 0 (N=8 in this example)

ph4_num_bytes = M, ph4_mode = 0 (M=16 in this example)

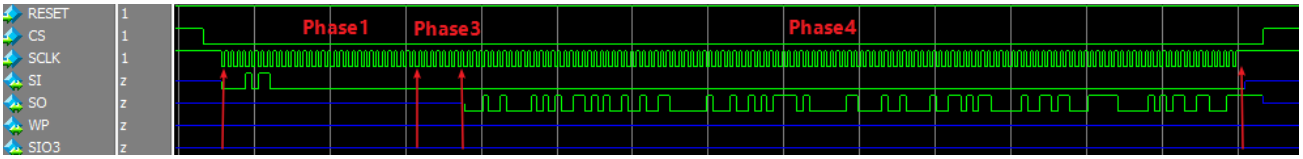

Figure 2.4. Example for FAST_READ Sequence

SPI Read Identification (RDID) example:

cmd_data = 0x9F000000

ph1_num_bytes = 1, ph1_mode = 0

ph2_num_bytes = 0, ph2_mode = 0

ph3_num_dummy_bits = 0, ph3_mode = 0

ph4_num_bytes = 3, ph4_mode = 0

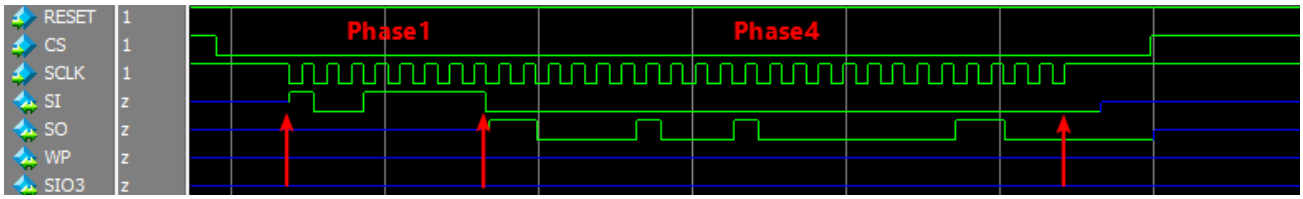

Figure 2.5. Example for RDID Sequence

SPI Flash QREAD4B example:

cmd_data = 0x6C000000

Tx FIFO contains 4-byte Read Address

ph1_num_bytes = 1, ph1_mode = 0

ph2_num_bytes = 4, ph2_mode = 0

ph3_num_dummy_bits = N, ph3_mode = 0 (N=8 in this example)

ph4_num_bytes = M, ph4_mode = 2 (M=64 in this example)

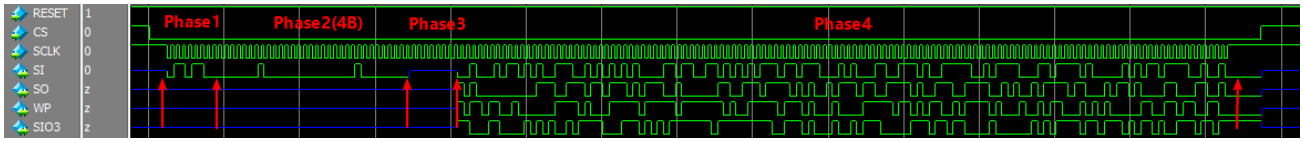

Figure 2.6. Example for QREAD4B Sequence

2.7.2. Width Conversion

Each Tx and Rx FIFO has a 32-bit access port for the APB system bus and an 8-bit access port for the SPI Controller state machine. The 8-bit data is packed or unpacked into 32-bit chunks as it enters or leaves the FIFOs. The endianness of the 32-bit data is determined by the Tx FIFO Endianness and Rx FIFO Endianness attributes. See Table 2.2.

Wherever possible, the implementation must avoid stalling the system bus while doing width conversions. For example, on the Tx FIFO, the 32-bit write value must be stored in a local register and the system bus write cycle must be terminated before doing the four 8-bit writes to the Tx FIFO. On the Rx FIFO, the logic must read bytes from the Rx FIFO into a local 32-bit register whenever the Rx FIFO is not empty, so that the 32-bit value can be returned immediately whenever a system bus read is received. This avoids tying up the system bus and stalling the CPU while the width conversions are being performed.

2.7.3. FIFO Empty or Full Behavior

The recommended usage model is for the CPU to write all of the data for a transaction to the Tx FIFO, for example, a full 256 byte page, before starting the transaction so that the Tx FIFO does not become empty in the middle of a transaction.

If the Rx FIFO indicates that it is full before the transaction is completed, then the SPI or QSPI state machine stalls until the Rx FIFO is no longer full. When this stall occurs, qpi_csn_o is held asserted but the SPI or QSPI clock is gated off, that is, held in the inactive state. When the Rx FIFO is not full, the clock is gated back on and data is received over SPI or QSPI.

2.8. User Interface Timing Diagram

2.8.1. APB Completer interface Timing

APB write operation from APB Requestor is shown in Figure 2.7.

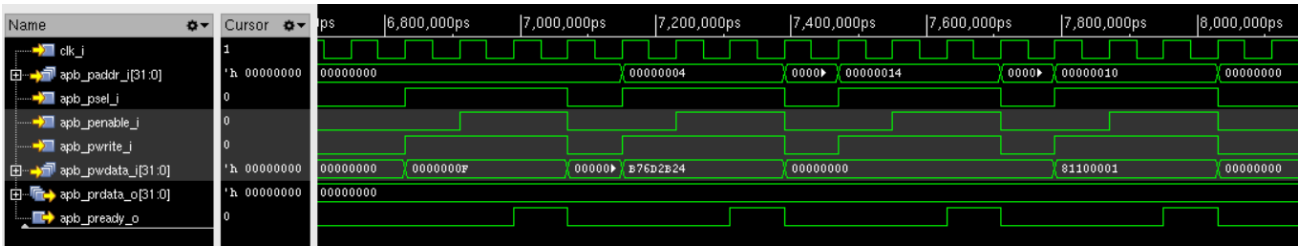

Figure 2.7. APB Writing Timing

APB read operation from APB Requestor is shown in Figure 2.8.

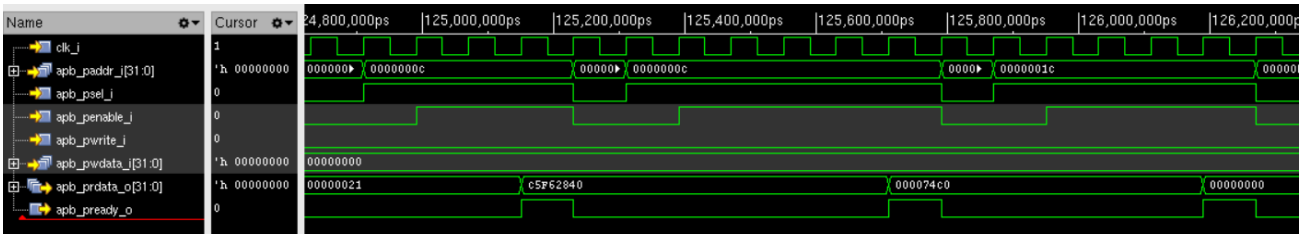

Figure 2.8. APB Reading Timing

Interrupt generation and acknowledge between APB Requestor and Completer is shown in Figure 2.9.

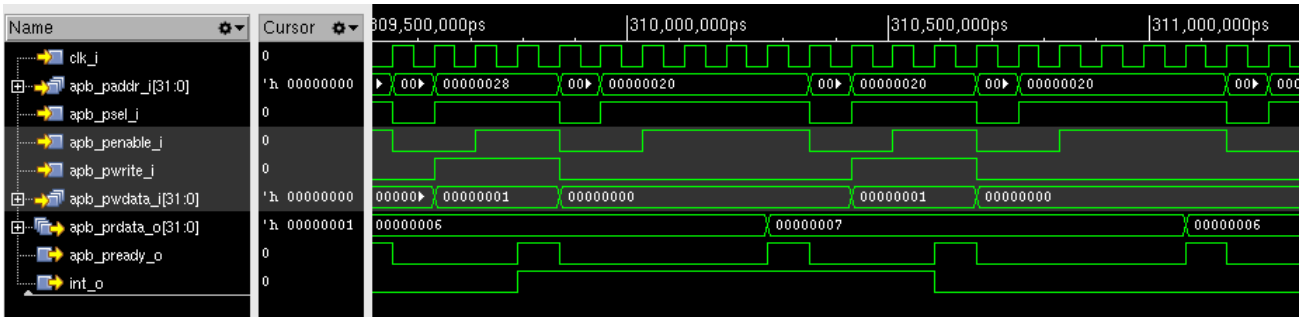

Figure 2.9. Interrupt Generation and Acknowledge Timing

2.8.2. External Rx FIFO Interface to ESB Timing

You can connect QSPI Controller Streamer to ESB for image authentication. The timing diagram is shown in Figure 2.10.

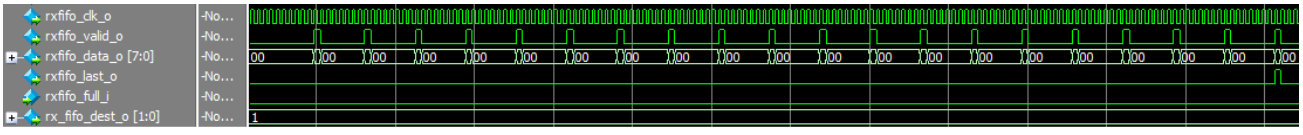

Figure 2.10. External Rx FIFO Interface Timing

2.8.3. Typical Flash Read or Program Flow

The typical flash read or program flow is shown in Figure 2.11. The flash used is MX25L12845G from MACRONIX INTERNATIONAL CO., LTD.

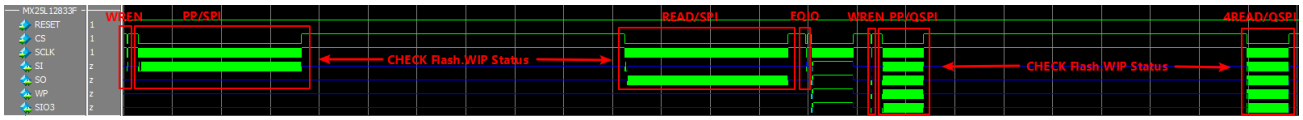

Figure 2.11. Typical Flash Read or Program Flow

3. Licensing and Ordering Information

The Lattice Sentry QSPI Controller Streamer IP is provided at no additional cost with the Lattice Propel Builder software.

Appendix A. Resource Utilization

Table A.1. Resource Utilization using LCMXO3D-9400HC-6BG484C

Configuration	Registers	LUTs	EBRs	Tools
Default	975	1348	2	Lattice Diamond Version 3.14 Synplify Pro (R) V-2023.09L-2, Build 349R

Table A.2. Resource Utilization using LFMXO4-110HE-5BBG256C

Configuration	Registers	LUTs	EBRs	Tools
Default	1181	1810	2	Lattice Radiant Version 2025.1 SP1 Synplify Pro (R) W-2024.09LR-SP1-1, Build 155R

References

- MachXO3D Devices web page
- MachXO4 Devices web page
- Lattice Sentry QSPI Streamer IP web page
- Lattice Sentry QSPI Controller Streamer IP – Lattice Propel Builder Release Notes (FPGA-RN-02111)
- Lattice Propel 2025.2 Builder User Guide (FPGA-UG-02243)
- Lattice Diamond Software 3.11 User Guide
- Lattice Propel Design Environment web page
- Lattice Radiant Software web page
- Lattice Diamond Software 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

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

Revision 1.1, IP v 1.2.0, December 2025

Section	Change Summary
All	• Updated the document title from *Lattice Sentry QSPI Master Streamer IP Core for MachXO3D - Lattice Propel Builder* to *Lattice Sentry QSPI Controller Streamer IP – Lattice Propel Builder*. • Changed *QSPI master* to *QSPI controller*. • Changed *SPI slave* to *SPI target*. • Changed *APB master* to *APB requestor*. • Changed *APB slave* to *APB completer*.
Disclaimers	Updated this section.
Inclusive Language	Added this section.
Abbreviations in This Document	• Updated the section title to its current. • Updated the abbreviation table.
Introduction	Updated the following paragraph: *The Lattice Semiconductor Sentry™ QSPI Master Streamer IP core for MachXO3D™ supports SPI and QSPI transactions. The design is implemented in Verilog HDL. It can be configured and generated using Lattice Propel™ Builder. It can be targeted to MachXO3D™ FPGA devices and implemented using the Lattice Diamond® software Place and Route tool integrated with the Synplify Pro® synthesis tool.* to *The Lattice Semiconductor Sentry™ QSPI Controller Streamer IP supports SPI and QSPI transactions. The design is implemented in Verilog HDL. It can be configured and generated using Lattice Propel™ Builder and implemented using the Lattice Diamond™ or Lattice Radiant software.*
Licensing and Ordering Information	• Updated the section title from *Ordering Part Number* to *Licensing and Ordering Information*. • Removed the old OPNs. • Added the following: *The Lattice Sentry QSPI Controller Streamer IP is provided at no additional cost with the Lattice Propel Builder software*.
References	Updated this section.
Technical Support Assistance	Added the link to Lattice Answer Database.
Appendix A. Resource Utilization	Updated Table A.1. Resource Utilization using LCMXO3D-9400HC-6BG484C. Added Table A.2. Resource Utilization using LFMXO4-110HE-5BBG256C.
Revision History	Added the following note: *In some instances, the IP may be updated without changes to the user guide. The user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates*.

Revision 1.0, May 2020

Section	Change Summary
All	Initial release.

www.latticesemi.com