



Lattice Sentry QSPI Master Streamer IP Core for MachXO3D - Lattice Propel Builder

User Guide

FPGA-IPUG-02109-1.0

May 2020

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 and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. 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. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. 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.

Contents

| | |
|---|----|
| Acronyms in This Document | 5 |
| 1. Introduction | 6 |
| 1.1. Features | 6 |
| 1.2. Conventions | 7 |
| 1.2.1. Nomenclature..... | 7 |
| 1.2.2. Signal Names | 7 |
| 1.2.3. Host | 7 |
| 1.2.4. Attribute Names..... | 7 |
| 2. Functional Description..... | 8 |
| 2.1. Block Diagram | 8 |
| 2.2. Signal Description..... | 9 |
| 2.3. Attribute Summary..... | 10 |
| 2.4. Register Description | 11 |
| 2.5. APB Slave Interface | 13 |
| 2.6. External FIFO Interface..... | 13 |
| 2.7. Operation | 14 |
| 2.7.1. Transaction Phases..... | 14 |
| 2.7.2. Width Conversion..... | 16 |
| 2.7.3. FIFO Empty/Full Behavior..... | 16 |
| 2.8. User Interface Timing Diagram | 17 |
| 2.8.1. APB Slave interface Timing..... | 17 |
| 2.8.2. External Rx FIFO Interface to ESB Timing | 17 |
| 2.8.3. Typical Flash Read/Program Flow | 18 |
| 3. Ordering Part Number | 19 |
| Appendix A. Resource Utilization | 20 |
| References..... | 21 |
| Technical Support Assistance | 22 |
| Revision History | 23 |

Figures

| | |
|---|----|
| Figure 2.1. QSPI Master Streamer Block Diagram | 8 |
| Figure 2.2. QSPI Master Streamer programmable phases..... | 14 |
| Figure 2-3. Example for PP Program Sequence | 15 |
| Figure 2-4. Example for FAST_READ Sequence..... | 15 |
| Figure 2-5. Example for RDID Sequence | 16 |
| Figure 2.6. Example for QREAD4B Sequence..... | 16 |
| Figure 2.7. APB Writing Timing | 17 |
| Figure 2.8. APB Reading Timing | 17 |
| Figure 2.9. Interrupt Generation and Acknowledge Timing | 17 |
| Figure 2.10. External Rx FIFO Interface Timing..... | 17 |
| Figure 2.11. Typical Flash Read/Program Flow..... | 18 |

Tables

| | |
|--|----|
| Table 2.1. QSPI Master Streamer Signal Description | 9 |
| Table 2.2. Attributes Table | 10 |
| Table 2.3. Attribute Description | 10 |
| Table 2-4. Summary of QSPI Master Streamer IP Core Registers | 11 |
| Table 2-5. Access Type Definition | 13 |
| Table A.1. Resource Utilization..... | 20 |

Acronyms in This Document

A list of acronyms used in this document.

| Acronym | Definition |
|---------|----------------------------------|
| APB | Advanced Peripheral Bus |
| CPU | Central Processing Unit |
| FIFO | First In, First Out |
| FPGA | Field-Programmable Gate Array |
| PFR | Platform Firmware Resiliency |
| QSPI | Quad Serial Peripheral Interface |
| SPI | Serial Peripheral Interface |

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 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.

1.1. Features

The key features of the QSPI Master 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 AMBA 3 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 Master Streamer IP through APB.

1.2.4. Attribute Names

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

2. Functional Description

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

The QSPI Master 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 external Rx FIFO output interface (8-bit) also enables direct transmission of input data from the SPI slave to another block, without tying up the CPU or system bus.

2.1. Block Diagram

QSPI Master 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 system bus (APB) and an 8-bit access port for the SPI Master 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 (see [Table 2-2](#)).

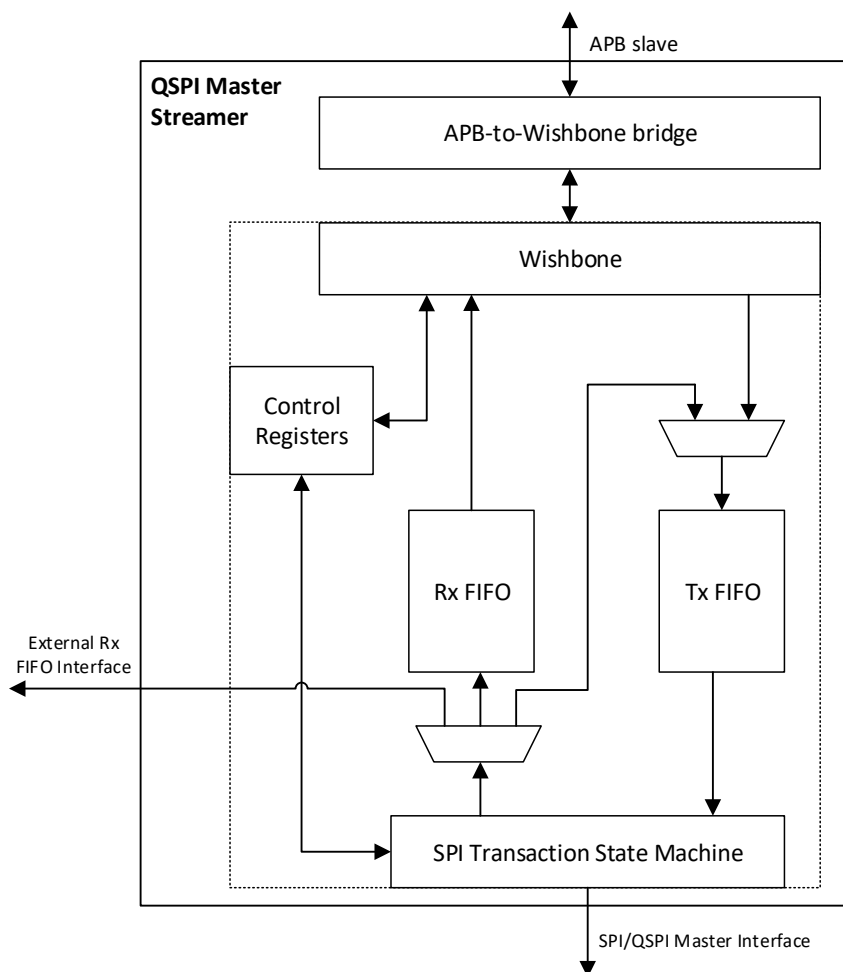


Figure 2.1. QSPI Master Streamer Block Diagram

2.2. Signal Description

Table 2.1. QSPI Master Streamer Signal Description

| Port | Width | Direction | Description |
|-------------------------------------|-------|-----------|---|
| System | | | |
| clk_i | 1 | Input | Master 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 slave 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. Slave uses this signal to extend an APB transfer. |
| apb_prdata_o | Out | 32 | Read data signal |
| QSPI Master | | | |
| 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 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/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 Master 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 | <i>big, little</i> | <i>big</i> | — |
| 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 | <i>big, little</i> | <i>big</i> | — |
| No. of External Rx FIFO Interfaces | 0, 1 | 1 | — |

Table 2.3. Attribute Description

| Parameter | Description | | | | | | | | | | | | | | | |
|---------------------------|--|------------------|-------|-------|------|-----|------------|---|---|---|---|---------------|---|---|---|---|
| General | | | | | | | | | | | | | | | | |
| SPI Mode | Default value for the <code>spi_mode</code> bit field in the <code>qspi_ctrl</code> register | | | | | | | | | | | | | | | |
| Configuration | | | | | | | | | | | | | | | | |
| SPI Clock Divider | Default value for the clock divisor <code>sck_div</code> bit field in the <code>qspi_ctrl</code> register <ul style="list-style-type: none"> 0: <code>Fqpi_sck_o = Fclk_i</code> 1: <code>Fqpi_sck_o = Fclk_i/2</code> 2: <code>Fqpi_sck_o = Fclk_i/4</code> 3: <code>Fqpi_sck_o = Fclk_i/8</code> 4: <code>Fqpi_sck_o = Fclk_i/16</code> 5: <code>Fqpi_sck_o = Fclk_i/32</code> | | | | | | | | | | | | | | | |
| Tx FIFO Size | <ul style="list-style-type: none"> Size of the transmit FIFO, in bytes Must be a multiple of 4 (that is, 32-bit aligned) | | | | | | | | | | | | | | | |
| Tx FIFO Almost Full Flag | Threshold value for signaling that the FIFO is almost full | | | | | | | | | | | | | | | |
| Tx FIFO Almost Empty Flag | Threshold value for signaling that the FIFO is almost empty | | | | | | | | | | | | | | | |
| Tx FIFO Endianness | Specifies the order of Tx FIFO bytes at the 32-bit APB interface. Transmit bytes over SPI in this order (from 0-3): <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>APB Tx FIFO Data</th> <th>31:24</th> <th>23:16</th> <th>15:8</th> <th>7:0</th> </tr> </thead> <tbody> <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> </tbody> </table> | APB Tx FIFO Data | 31:24 | 23:16 | 15:8 | 7:0 | Big endian | 0 | 1 | 2 | 3 | Little endian | 3 | 2 | 1 | 0 |
| APB Tx FIFO Data | 31:24 | 23:16 | 15:8 | 7:0 | | | | | | | | | | | | |
| Big endian | 0 | 1 | 2 | 3 | | | | | | | | | | | | |
| Little endian | 3 | 2 | 1 | 0 | | | | | | | | | | | | |
| Rx FIFO Size | <ul style="list-style-type: none"> Size of the receive FIFO, in bytes Must be a multiple of 4 (that is, 32-bit aligned) | | | | | | | | | | | | | | | |
| Rx FIFO Almost Full Flag | Threshold value for signaling that the FIFO is almost full | | | | | | | | | | | | | | | |
| Rx FIFO Almost Empty Flag | Threshold value for signaling that the FIFO is almost empty | | | | | | | | | | | | | | | |

| Parameter | Description | | | | | | | | | | | | | | | |
|------------------------------------|--|------------------|-------|-------|------|-----|------------|---|---|---|---|---------------|---|---|---|---|
| Rx FIFO Endianness | <p>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):</p> <table border="1"> <thead> <tr> <th>APB Rx FIFO Data</th> <th>31:24</th> <th>23:16</th> <th>15:8</th> <th>7:0</th> </tr> </thead> <tbody> <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> </tbody> </table> | APB Rx FIFO Data | 31:24 | 23:16 | 15:8 | 7:0 | Big endian | 0 | 1 | 2 | 3 | Little endian | 3 | 2 | 1 | 0 |
| APB Rx FIFO Data | 31:24 | 23:16 | 15:8 | 7:0 | | | | | | | | | | | | |
| Big endian | 0 | 1 | 2 | 3 | | | | | | | | | | | | |
| Little endian | 3 | 2 | 1 | 0 | | | | | | | | | | | | |
| No. of External Rx FIFO Interfaces | <ul style="list-style-type: none"> Number of external Rx FIFO interfaces (8-bit) Default is 1 | | | | | | | | | | | | | | | |

2.4. Register Description

The QSPI Master Streamer IP core register map is shown in the [Table 2-4](#).

Table 2-4. Summary of QSPI Master Streamer IP Core Registers

| Offset | Name | Access | Description |
|--------|-------------------|--------|--|
| 0x00 | QSPI_CTRL | RW | <p>spi_mode[1:0]</p> <ul style="list-style-type: none"> 00: SPI mode 0 01: reserved 10: reserved 11: SPI mode 3 <p>sck_div[4:2]:</p> <ul style="list-style-type: none"> 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 <p>reserved[30:5] soft_reset[31]</p> <p>Writing 1 to this bit resets all of the internal logic, flushes the FIFOs (resets the read/write pointers), and restores all registers to their default settings. Reads return 0. Intended for error recovery.</p> |
| 0x04 | CMD_DATA | RW | Command data to transmit in transaction phase 1 (always big endian) |
| 0x08 | TX_FIFO_DATA | WO | <p>Data to transmit in transaction phase 2</p> <p>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. Endianness depends on <i>Tx FIFO Endianness</i> attribute.</p> |
| 0x0C | RX_FIFO_DATA | RO | <p>Data received in transaction phase 4</p> <p>If the Rx FIFO contains less than four bytes when a 32-bit read is received on the system bus and there is a SPI transaction currently in progress, the read is blocked until 4 bytes are received or the SPI transaction completes. Endianness depends on <i>Rx FIFO Endianness</i> attribute.</p> |
| 0x10 | TRANSACTION_CTRL1 | RW | <p>ph1_num_bytes[2:0] – Number of bytes from cmd_data to transmit in transaction phase 1 (legal values: 0-4)</p> <p>ph2_num_bytes[11:3] – Number of bytes from Tx FIFO to transmit in transaction phase 2 (legal values: 0- <i>Tx FIFO Size</i>)</p> <p>ph3_dummy_cycles[16:12] – Number of dummy cycles to transmit in transaction phase 3</p> <p>ph1_mode[18:17] – Transmit phase 1 data in:</p> <ul style="list-style-type: none"> 0: SPI mode 1: reserved 2: QSPI mode 3: reserved |

| Offset | Name | Access | Description |
|--------|-------------------|--------|--|
| | | | <p>ph2_mode[20:19] – Transmit phase 2 data in:</p> <ul style="list-style-type: none"> • 0: SPI mode • 1: reserved • 2: QSPI mode • 3: reserved <p>ph3_mode[22:21] – Transmit phase 3 dummy cycles in:</p> <ul style="list-style-type: none"> • 0: SPI mode • 1: reserved • 2: QSPI mode • 3: reserved <p>ph4_mode[24:23] – Receive phase 4 data in:</p> <ul style="list-style-type: none"> • 0: SPI mode • 1: reserved • 2: QSPI mode • 3: reserved <p>rxfifo_last_en[25] – Enable(1)/Disable(0) assertion of rxfifo_last_o for the last received byte of the SPI transaction</p> <p>reserved[30:26]</p> <p>start[31] – Write 1 to start an SPI transaction (reads return 0)</p> |
| 0x14 | TRANSACTION_CTRL2 | RW | ph4_num_bytes[31:0] – Number of bytes to receive in transaction phase 4 |
| 0x18 | STATUS | RO | <p>tx_fifo_empty[0] – Tx FIFO is empty</p> <p>tx_fifo_almost_empty[1] – Tx FIFO is not empty and has less than <i>Tx FIFO Almost Empty Flag</i> bytes</p> <p>tx_fifo_almost_full[2] – Tx FIFO is not full and has more than <i>Tx FIFO Almost Full Flag</i> bytes</p> <p>tx_fifo_full[3] – Tx FIFO is full</p> <p>rx_fifo_empty[4] – Rx FIFO is empty</p> <p>rx_fifo_almost_empty[5] – Rx FIFO is not empty and has less than <i>Rx FIFO Almost Empty Flag</i> bytes</p> <p>rx_fifo_almost_full[6] – Rx FIFO is not full and has more than <i>Rx FIFO Almost Full Flag</i> bytes</p> <p>reserved[30:8]</p> <p>busy[31] – SPI transaction is in progress</p> |
| 0x1C | FIFO_CTRL | RW | <p>reserved[6:0]</p> <p>tx_fifo_flush[7] – Flush contents of Tx FIFO (reset read and write pointers)</p> <p>rx_fifo_dest[9:8]:</p> <ul style="list-style-type: none"> • 0: internal Rx FIFO • 1: external Rx FIFO interface • 2: reserved • 3: internal Tx FIFO <p>reserved[14:10]</p> <p>rx_fifo_flush[15]: flush contents of Rx FIFO (reset read and write pointers)</p> <p>reserved[31:16]</p> |
| 0x20 | INT_STATUS | RW | <p>Interrupt status:</p> <p>done_int[0] – Done interrupt (SPI transaction completed)</p> <p>tx_fifo_empty_int[1] – Tx FIFO Empty interrupt</p> <p>tx_fifo_almost_empty_int[2] – Tx FIFO Almost Empty interrupt</p> <p>tx_fifo_almost_full_int[3] – Tx FIFO Almost Full interrupt</p> <p>tx_fifo_full_int[4] – Tx FIFO Full interrupt</p> <p>rx_fifo_empty_int[5] – Rx FIFO Empty interrupt</p> <p>rx_fifo_almost_empty_int[6] – Rx FIFO Almost Empty interrupt</p> <p>rx_fifo_almost_full_int[7] – Rx FIFO Almost Full interrupt</p> |

| Offset | Name | Access | Description |
|--------|------------|--------|---|
| | | | rx_fifo_full_int[8] – Rx FIFO Full interrupt reserved[31:9] Writing 1 to a bit clears that interrupt FIFO interrupts are triggered on the rising edge of the corresponding FIFO condition (empty, full, etc.) 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 (SPI transaction 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 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 register value | Ignores write access |
| WO | Returns 0 | Updates register value |
| RW | Returns register value | Updates register value |
| RW1C | Returns register value | Writing 1'b1 on register bit clears the bit to 1'b0. Writing 1'b0 on register bit is ignored. |
| RSVD | Returns 0 | Ignores write access |

2.5. APB Slave Interface

The APB slave interface provides an APB slave 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 Master Streamer can be connected to the Tx FIFO input of a separate QSPI Master 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 Master 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 should write data to the FIFO before starting the transaction (see examples sequence below for details).

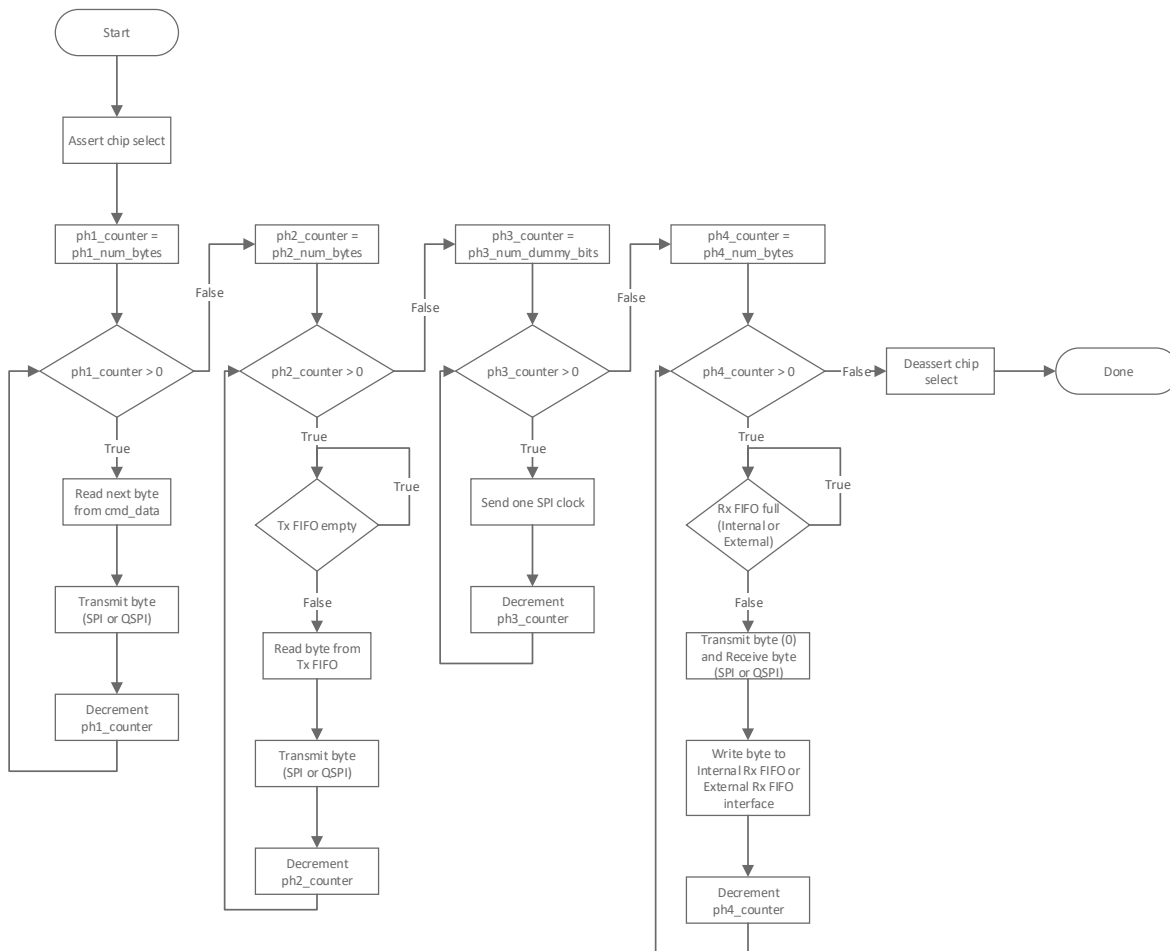


Figure 2.2. QSPI Master Streamer programmable phases

Phase 1: Transmit `ph1_num_bytes` (0-4) bytes from `cmd_data` register

- For SPI flash devices, this normally includes 1 command byte and 0 or 3 address bytes.
- Data is transmitted in 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` (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 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 (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 SPI mode or QSPI mode depending on the ph3_mode setting.
- Serial data input is ignored.

Phase 4: Receive ph4_num_bytes (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.
- SPI slave ignores the data.

SPI Flash Page Program 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 example)

ph3_num_dummy_bits = 0, ph3_mode = 0

ph4_num_bytes = 0, ph4_mode = 0

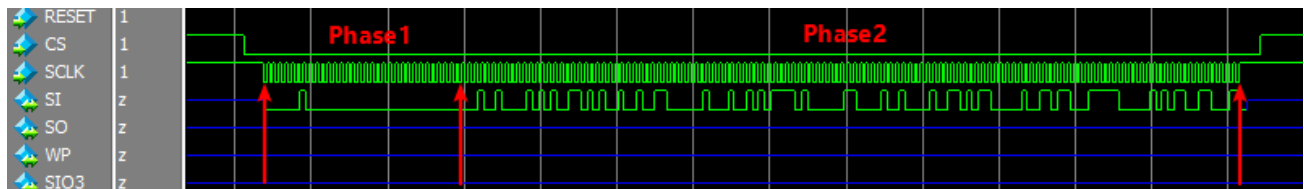


Figure 2-3. Example for PP Program Sequence

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 example)

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

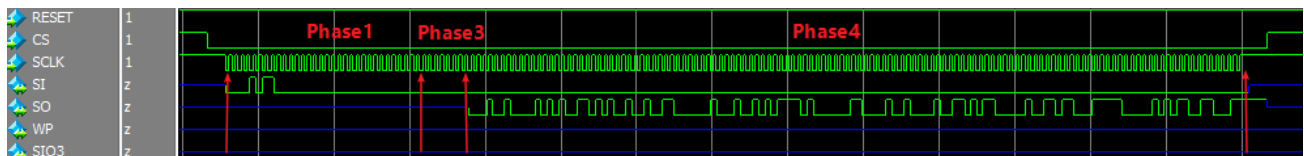


Figure 2-4. Example for FAST_READ Sequence

SPI 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

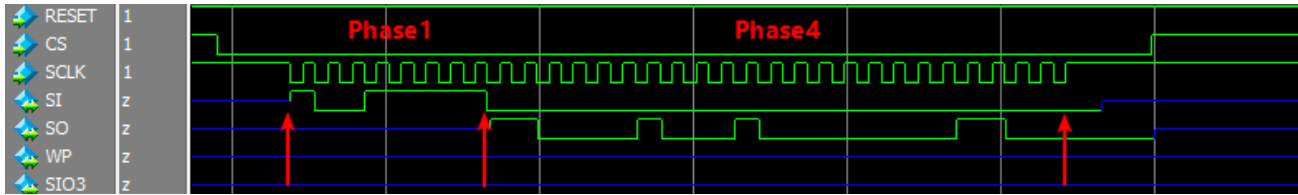


Figure 2-5. Example for RDID Sequence

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

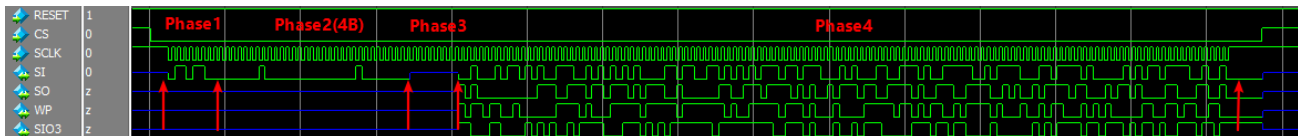


Figure 2.6. Example for QREAD4B Sequence

2.7.2. Width Conversion

Each Tx and Rx FIFO has a 32-bit access port for the system bus (APB) and an 8-bit access port for the SPI Master 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 should avoid stalling the system bus while doing width conversions. For example, on the Tx FIFO, the 32-bit write value should be stored in a local register and the system bus write cycle should be terminated before doing the four 8-bit writes to the Tx FIFO. On the Rx FIFO, the logic should 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/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/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/QSPI clock is gated off (held in the inactive state). When the Rx FIFO is not full, the clock is gated back on and data is received over SPI/QSPI.

2.8. User Interface Timing Diagram

2.8.1. APB Slave interface Timing

APB write operation from APB master is shown in [Figure 2.7](#).

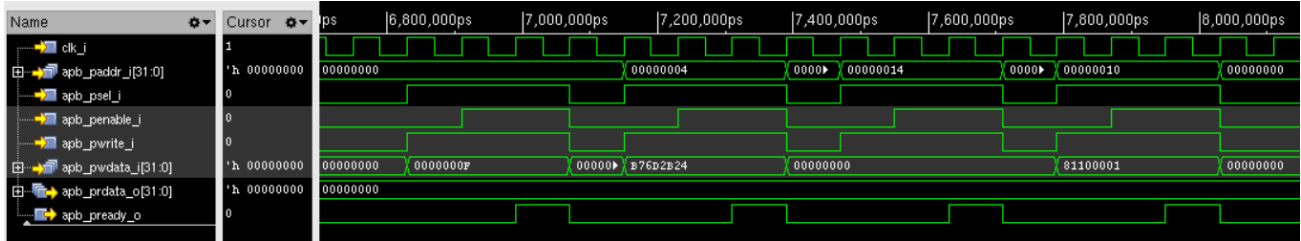


Figure 2.7. APB Writing Timing

APB read operation from APB master is shown in [Figure 2.8](#).

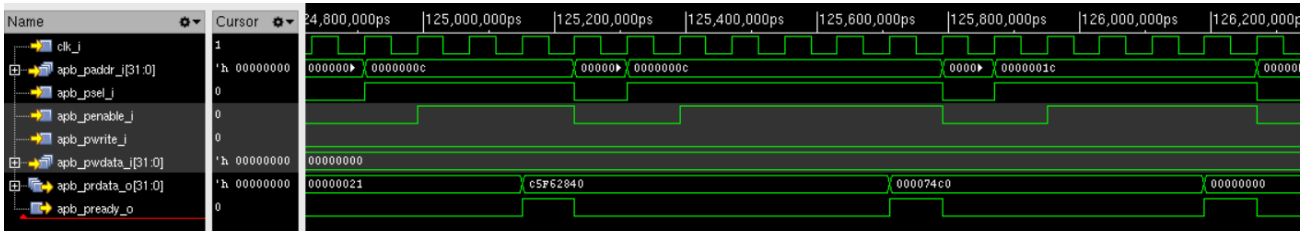


Figure 2.8. APB Reading Timing

Interrupt generation and acknowledge between APB master and slave is in [Figure 2-9](#).

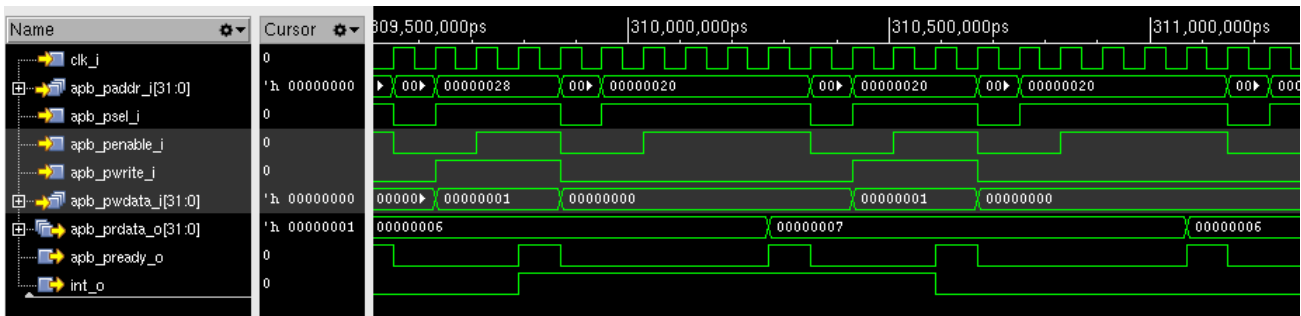


Figure 2.9. Interrupt Generation and Acknowledge Timing

2.8.2. External Rx FIFO Interface to ESB Timing

You can connect QSPI Master Streamer to ESB (for image authentication). The timing diagram is described in [Figure 2-10](#).

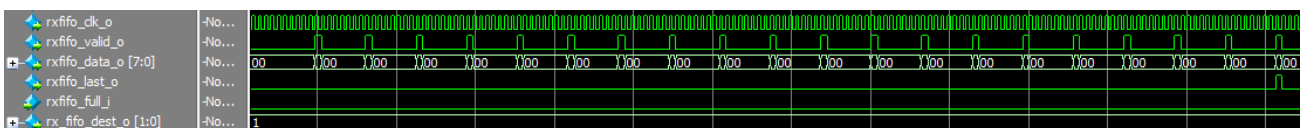


Figure 2.10. External Rx FIFO Interface Timing

2.8.3. Typical Flash Read/Program Flow

The typical flash (MX25L12845G, MACRONIX, CO, Ltd) read/program flow is shown in Figure 2.11.



Figure 2.11. Typical Flash Read/Program Flow

3. Ordering Part Number

The Ordering Part Number (OPN) for the QSPI Master Streamer IP Core targeting MachXO3D FPGA devices are the following:

- QSPIMS-M3D-U – Project License
- QSPIMS-M3D-UT – Site License

Appendix A. Resource Utilization

Table A.1. Resource Utilization

| FPGA Resource Utilization | Registers | LUTs | EBRs | Target Device | Synthesis Tools |
|---------------------------|-----------|------|------|---------------|--|
| Number of resources | 945 | 1335 | 2 | MachXO3D | Synopsys® Synplify Pro N-2018.03L-SP1-1 |

References

- [MachXO3D FPGA Web Page in latticesemi.com](#)
- [Lattice Propel 1.0 User Guide](#)
- [Lattice Diamond Software 3.11 User Guide](#)

Technical Support Assistance

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

Revision History

Revision 1.0, May 2020

| Section | Change Summary |
|---------|------------------|
| All | Initial release. |



www.latticesemi.com